Intro and Image Capture/Display

This project will familiarize you with the basics of MATLAB for image processing and will stimulate you to think about some issues concerning image capture.

NOTE: This document contains hyperlinks that are highlighted with a light blue box. These links lead to online help as well as image files you will need for each project.

Getting Started

1. To start MATLAB on a PC, simply go to the start menu and select the MATLAB group and then MATLAB from the dropdown menu.
2. To become familiar with some of the "gee-whiz" capabilities of MATLAB, try running `openExample('matlab_featured/xfourier')`. and then clicking the Run and Advance button at the top of the MATLAB editor. Pay attention to the text that is generated in the window. The text shows the MATLAB commands that were used to generate the examples. You may also wish to explore some of the other examples by visiting `https://www.mathworks.com/help/matlab/examples.html`.
3. Run `openExample('matlab_featured/intro')` to get an idea of the basic capabilities of MATLAB.
4. You can type `doc` at any time to get a hypertext version of the MATLAB Reference Guide or `help <cmd>` to get help on specific commands (assuming this is installed on the machine where you are).
5. If you need help on a MATLAB function and the hyperlink doesn't work, simply type `help <function>` at the MATLAB prompt.
6. Read "MATLAB for DSP" and "MATLAB for Image Processing". The techniques you learn in these short overviews will be very useful throughout the semester.

Exercises

1. Type "help images" to see a list of all functions in the Image Processing Toolbox or see the "Image Processing Toolbox User's Guide". You will note under "Image file I/O" the function `imread`. Note that MATLAB allows you to read in a number of different image types. This function can be called so that it returns two arguments — a `colormap` matrix and an image matrix.
   (a) Read in the image `sedona.png`. Display the image using `image`. (Be sure to set the colormap to the one that is read in with the image.)
   (b) Examine the colormap and determine how colormaps work to produce what you see on the screen.
2. Set the colormap to gray(256).
   (a) What happens if the index values are displayed directly using this colormap?
   (b) Why does the wrong colormap make it look so bad in this case?
3. Create a ramp image:

(a) Create a 256x256 image whose index is 1 in the first column and ramps to 256 in the last column. If possible, create this image without using any for loops. Display this image using a gray(256) colormap.

(b) Experiment with other colormaps.

4. Create a tinted image:

   (a) Determine how to convert the scenic image to a "parula" colormap so that the scene appears to have been viewed through a tinted filter. You may need to use `ind2gray` and `gray2ind`. (If it looks unusually noisy compared to the original color image, you did it wrong.)

   (b) Figure out how to save the image as a PNG image using `imwrite`.

5. The files `paper1.tif` and `paper2.tif` are two scanned images of the same piece of paper taken a few seconds apart without moving the paper. (You may read them in as grayscale.)

   (a) Are the images solid white? How can you tell? Is this what you expect?

   (b) Are the two images the same? How can you tell? If there are differences, can you describe them and speculate as to what accounts for the difference?

Write a **brief** memo that summarizes your findings. (You may use two pages if absolutely necessary, but one page is best. Images can be small.) You will have to be selective in your discussion to meet the space requirements, but answer all questions at least briefly. (That means you may need to leave some things out that you did in the exercise.) Include at least one image or image comparison and one code snippet. If the project specifically asks that an image or code snippet be included, then you must include that. The memo should be written so that it makes sense without reference to the project instructions.

Submit a PDF of your project memo on Canvas by class time on the due date.

NOTE: All out-of-class work is to be done **independently** and should represent your work alone. Sharing of programming tips and discussing general concepts is ok. Collaborating on experiments or code-writing is not. **Any** such collaboration on these assignments will be considered an act of dishonesty and will be treated accordingly. Memos may be checked using Turnitin (TM) for excessive similarity to one another and to documents available online.

For further help:

- MATLAB Primer
- MATLAB Help Desk

Sampling and Aliasing

This project will give you some hands-on experience with sampling 2-D signals and will demonstrate the effects of aliasing.

You should review your notes on sampling and aliasing before beginning this project. **NOTE: It is essential that you understand aliasing thoroughly and that you do each step accurately. Otherwise, you may do the entire project incorrectly and fail to learn anything from it.**

Exercises

1. Create a 512x512 image containing a vertical white line three pixels wide on a black background. Use `imrotate` with bicubic interpolation to rotate the image 15 degrees. Extract a 256x256 section out of the original image and the rotated image so that the line in the extracted image has ends on the top and bottom edges rather than cutting across a corner. Compare the two 256x256 images visually, and comment on the similarities or differences, if any. Use `truesize` so that each pixel in the display corresponds to a pixel in the image. (Make sure this is the case for every display in this exercise. See the help entry.) Examine the pixel values closely, and describe how the line at an angle is represented in a discrete array.

2. The Fourier magnitude of an image can be viewed by setting the colormap using `colormap(gray(256))`. You can view the the Fourier magnitude to see the stronger features using `imagesc(abs(fftshift(fft2(IMAGE))))` and then view the log-magnitude to see some of the finer details. To view a Fourier magnitude, use a dynamic-range compression mapping such as `log(abs(fftshift(fft2(IMAGE)))+0.01)`, which can then be viewed as an image using `imagesc`. The small constant is there in case any of the Fourier components are nearly zero and can be varied to change the "floor" of the mapping. Note that the origin is in the middle of the image in this display method. What you are viewing here is one period of the periodic frequency spectrum of the discrete image. To get a sense of what is happening with adjacent periods, you may want to view a 3x3 replication of the displayed image using `repmat(IMAGESC_ARG,[3 3])` substituted for the argument to imagesc in the display command above. That is, the goal is to replicate the spectrum, not the spatial-domain image. This display method may also be helpful in understanding some of the following exercises.

   NOTE: When you view the log-magnitude, the darker components have very little significance to the issues in this project. Some lower-magnitude features occur as the result of using an FFT, which assumes that the image is periodic. Such features may appear as a strange texture background over the image. Those features will be considered artifacts for this project and not features of interest. You may want to try increasing the small constant to reduce the visibility of the background artifacts (try 1 or 10).

   (a) View the rotated and unrotated 256x256 images.

   (b) Explain using comparisons with simple Fourier transform pairs and properties why the strongest features in the Fourier magnitudes appear as they do in the two images. (Consider the 1-D Fourier transform of a constant, an impulse, and a pulse; the Fourier transform property of separable signals; and the Fourier transform property of a rotated image.)

   (c) Explain why the differences occur.

3. Consider the 256x256 images generated in Step 1 to represent continuous resolution.

(a) Create new 128x128 images that consist of every other sample of each of the 256x256 images. Taking every other sample will represent the effect of sampling a continuous image. View the images, and explain what, if anything, has changed. Examine the pixel values, and describe what you find.

(b) View the Fourier magnitudes and explain in terms of aliasing. Remember to display the replicated spectral plots as described in #2. What happens to the strong features of the Fourier magnitude of the rotated line?

4. Nonideal sampling can be simulated by averaging each 2x2 block in a "continuous" image to create a new image whose pixels are the block averages of the input image and whose size is one-half the original in each dimension. (This averaging approximates integrating over a continuous pixel region.)

   (a) Write a **general** m-file `function` that will implement this operation on an input image of any size and return the downsampled output image. Please include your m-file in your report.

   (b) Compare the 128×128 rotated-line image resulting from this process to the 128×128 rotated-line image resulting from the previous step, and explain the differences in terms of our in-class analysis of nonideal sampling.

   (c) Look at the periodically replicated Fourier log-magnitudes and explain the differences compared to the previous step.

5. Consider sampling a sinusoid.

   (a) Create a 256x256 image $f(m, n) = \cos(0.2\pi m + 0.6\pi n)$. Create a new image from every other sample of this image.

   (b) View the two images and their Fourier magnitudes (clearer than log-magnitudes in this case). Explain the differences.

   (c) What apparent frequencies are found in the downsampled image? (You should be able to calculate specific numerical frequencies in radians/sample. Use the original equation for $f(m, n)$ to determine this rather than the Fourier plot. The plot should correspond to what you find mathematically.)

Write a **brief** memo that summarizes your findings. (You may use two pages if absolutely necessary, but one page is best. Images can be small.) You will have to be selective in your discussion to meet the space requirements, but answer all questions at least briefly. (That means you may need to leave some things out that you did in the exercise.) Include at least one image or image comparison and one code snippet. If the project specifically asks that an image or code snippet be included, then you must include that. The memo should be written so that it makes sense without reference to the project instructions.

Submit a PDF of your project memo on Canvas by class time on the due date.

NOTE: All out-of-class work is to be done **independently** and should represent your work alone. Sharing of programming tips and discussing general concepts is ok. Collaborating on experiments or code-writing is not. **Any** such collaboration on these assignments will be considered an act of dishonesty and will be treated accordingly. Memos may be checked using Turnitin (TM) for excessive similarity to one another and to documents available online.

For further help:

- Matlab Primer
- Matlab Documentation

Fourier Transforms and Convolution

This exercise is intended to familiarize you with 2-D Fourier transforms, convolution using Fourier transforms, and display of Fourier transforms.

Exercises

1. Form a convolution kernel with `ker1 = ones(5,11)/55;`. Take the 2-D FFT via `fft2(ker1,256,256);`.
   (a) Display the magnitude of the 2-D FFT. (Use `abs`.) You may want to use `fftshift` to shift the origin in the Fourier domain to the center of the plot. Include the Fourier magnitude image in your report.
   (b) Give an explanation for why the magnitude plot appears as it does (in both coordinates) in terms of simple Fourier transform pairs and properties.
   (c) Use the FFT plot to explain what would happen to an image if it were convolved with this 2-D signal `ker1`.
2. Read in `camera.tif` and convert the integer values to a floating-point image with `double`.
   (a) Take the 2-D FFT, and plot the magnitude. What does the plot look like? (Look **very** closely. It isn't all black.)
   (b) Use the dynamic range compression formula (also used in Project 2) to transform the magnitude coefficients. Scale so that the maximum value is 255 (or use `imagesc`), and plot. Describe the resulting plot.
   (c) What strong features in the original image account for the most noticeable features in the Fourier transform? (Think in terms of simple Fourier transform pairs and properties.)
3. Multiply the FFT coefficients of the image and the kernel, and take the inverse FFT (`ifft2`). (You may need to take the real part because the finite precision creates a very small imaginary component.)
   (a) Display the result, and describe it.
   (b) What operation does this represent on the original cameraman image?
   (c) Compare this to a linear convolution of the image with the kernel using `conv2`.

   Include these images your report.
4. Use `cputime` to determine the following:
   (a) How long does a 1024×1024 FFT take on your machine? Create a random image with `rand` to time it. (It is so fast that you'll need to do it 100 times and average.)
   (b) How many floating point operations per second does your machine do? Use the formula from class for the number of multiplies as the number of floating point operations.
   (c) Determine how long a direct DFT would take for a 1024×1024 image on your machine, using the formula M^2N^2 as the total number of floating point operations.

Write a **brief** memo that summarizes your findings. (You may use two pages if absolutely necessary, but one page is best. Images can be small.) You will have to be selective in your discussion to meet the space requirements, but answer all questions at least briefly. (That means you may need to leave some things out that you did in the exercise.) Include at least one image or image comparison and one code snippet. If the project specifically asks that an image or code snippet be included,

then you must include that. The memo should be written so that it makes sense without reference to the project instructions.
Submit a PDF of your project memo on Canvas by class time on the due date.
NOTE: All out-of-class work is to be done **independently** and should represent your work alone. Sharing of programming tips and discussing general concepts is ok. Collaborating on experiments or code-writing is not. **Any** such collaboration on these assignments will be considered an act of dishonesty and will be treated accordingly. Memos may be checked using Turnitin (TM) for excessive similarity to one another and to documents available online.

For further help:

- MATLAB Primer
- MATLAB Documentation

Image Enhancement

This exercise is intended to expose you to some basic image enhancement operations.

Exercises

1. Download `imadjdemo.m` from the Canvas course home page, and run the `imadjdemo` intensity adjustment demo in MATLAB. (Ignore the warning messages.)

   (a) Select the Quarter image. Select Histogram Equalization where the Intensity Adjustment box is. Does this improve the look of the image? Explain.

   (b) Select Intensity Adjustment again. Now adjust the intensity transformation curve with the mouse until you think the image looks best. Include the intensity transformation in your report (either a plot or a sketch). What kind of intensity transformation does this represent? Describe the effect on the histogram.

2. Write a MATLAB `function` that will do the following (be sure to submit the code in your report):

   - Calculate the vertical and horizontal edge strength. Convolve the input image $x(m, n)$ once with `[ 1 2 1; 0 0 0; -1 -2 -1]` and once with this kernel rotated 90 degrees to obtain two directional gradient images $s_m(m, n)$ and $s_n(m, n)$. Then calculate the edge strength image as $s(m, n) = \sqrt{s_m^2(m, n) + s_n^2(m, n)}$. (This is the first part of the edge detection algorithm discussed in class.)
   - Filter the input image with an $M \times M$ averaging filter to obtain $x_a(m, n)$.
   - Filter the gradient image $s(m, n)$ with an $M \times M$ averaging filter to obtain $s_a(m, n)$.
   - Calculate a weight image as $w(m, n) = \exp(-s_a(m, n)/S)$, where S is an input parameter.
   - Form an output image in which each output point is a weighted sum of the input and smoothed images. The output should be defined as

   $$y(m, n) = (1 - w(m, n))x(m, n) + w(m, n)x_a(m, n)$$

   Inputs to this function should be x, M, and S.

   Do your best to implement this function without using loops. (It will probably take longer to process one image with loops than it takes to figure out how to implement it without loops!)

   The idea behind this function is to smooth the image in regions where the edge strength is low but to avoid smoothing where edge strength is high so that edges are not blurred but the overall image is smoothed.

3. Read `camera.tif` into MATLAB and convert the integer values to a floating-point image with `double`. Add noise generated by `randn` and scaled by 10.

   (a) Using the function you wrote, filter the image with smoothing filters of length $M = 3$, 5, and 7 for $S = 500$. Which filter appears to do the best job?

   (b) Describe the effect of the filters on the image.

   (c) What happens as the filter size increases?

   (d) Try changing S and M in the algorithm to get better results. Report on what you discover.

   (e) Include in your report the filtered image that appears best to you.

(f) Evaluate the output images for $M = 3$, 5, and 7 and $S = 500$ in terms of the sum of squared differences with respect to the original noiseless image:
`ssd = sum((filtered(:)-original(:)).^2).`
Do the same with sum of absolute differences:
`sad = sum(abs(filtered(:)-original(:))).`
Do these error measures correspond to your visual assessment of image quality?

4. Examine the following images:
`rob1-1.TIFF`
`rob1-2.TIFF`
`rob1-3.TIFF`
`rob1-4.TIFF`
`rob2-1.TIFF`
`rob2-2.TIFF`
`rob2-3.TIFF`
`rob2-4.TIFF`
The following information explains the images:

> The files in this directory are of a bankrobber. They were captured in 8 bit grayscale (TIFF) and are somewhat fuzzy. The names are rob#-#.TIFF, where the first # is the frame number and the second # is the copy number. There are 4 copies of the first and second frames acquired from separate frame grabs.
>
> The police would like to be able to identify this guy (the rightmost customer in frames 1 and 2), but the resolution is too poor.

Discuss how you might improve these images.

Write a **one-page** memo that summarizes your findings. You will have to be selective in your discussion to meet the space requirements, but answer all questions at least briefly. (That means you may need to leave some things out that you did in the exercise.) Include at least one image or image comparison and one code snippet. The memo should be written so that it makes sense without reference to the project instructions.
Submit a PDF of your project memo on Canvas by class time on the due date.
NOTE: All out-of-class work is to be done **independently** and should represent your work alone. Sharing of programming tips and discussing general concepts is ok. Collaborating on experiments or code-writing is not. **Any** such collaboration on these assignments will be considered an act of dishonesty and will be treated accordingly. Memos may be checked using Turnitin (TM) for excessive similarity to one another and to documents available online.

For further help:

- Matlab Primer
- Matlab Documentation