ELEE 4920/5920 Image Processing and Computer Vision: Project 1

Due Date: 24 February 2021

Objectives: This exercise will help you develop further familiarity with some of the basic features of the Matlab Image Processing Toolbox and the Matlab programming environment. You will explore the handling of image arrays inside Matlab developing an understanding of the correspondence between the two dimensional picture and its mathematical representation. You will also experimentally evaluate the effects of power-law based contrast adjustment (point-processing) as well as gray-level and spatial quantization effects. **This is an individual project (no teamwork allowed). You may discuss problems, but you must develop and submit your own work.**

Text Coverage: Chapters 1, 2, and chapter 3 (partial).

Course Learning Outcomes Addressed:

1. Demonstrate an understanding of the engineering science which underlies the field of image processing. (ECE/ABET 1).
2. Analyze and/or Implement basic spatial and/or frequency domain image enhancement algorithms (ECE/ABET 2,6);

In the following exercises you must prepare your final solutions in the form of matlab mlx files. You may of course develop your solutions using the interactive matlab environment, but you will need to structure your submitted solutions as matlab mlx and possibly **function** m-files as you deem appropriate (to look up functions – type 'help function'). Function m-files are the matlab equivalent of subroutines. Also, be careful to place a semi-colon ';' at the end of lines which produce large output files. Failure to do so will present you with thousands of lines of scrolling text (which I do not want to see when I review your submissions). You must prepare a well commented PDF electronic submission as well, using the matlab editor's publishing features ('save-as menu pull-down'). Be sure to include a Table of Contents after the Title and use multiple sections with good descriptive text to divide the work. Use ample text descriptions, equations, and/or imported images to introduce methods and explain results. Also, use Titles and other font adjustments to make the published result more appealing.

1. This exercise will help us become familiar with handling of image arrays inside Matlab. Be sure that the images red_cow.jpg and snapper.jpg from this projects archive are in your working directory, and answer the following questions being sure *to explain your approach and your answers:*

(a) Load the images red_cow.jpg and snapper.jpg into Matlab. What are the sizes of the two images?. What method did you use to determine the size?

(b) What are the color values of pixels at row 29, column 86 for red_cow and row 198, column 201 for snapper? Please pay attention to image and matrix indexing in matlab.

(c) **(5 points)** Plot the 103rd row of the green plane for red_cow {Note: red = A(:,:,1), green = A(:,:,2), blue = A(:,:,3)}. Plot the 69th column of the blue plane of snapper. [Use the *plot()* command]. Each plot should place the pixel magnitude along the y-axis and the position along the row or column along the x-axis. Be sure to properly label the plots (using *label, title* etc). **Next** -- use the matlab intensity profile commands to select a 2-segment line to profile [Using matlab help -- search for information on creating an intensity profile of an image]. Plot the image showing the profile and the 3D plot rotated to make the results easy to interpret.

(d) **(5 points)** Create and draw the image shown below in Figure 1. The image sub-block (of size 128x128) on the top-left corner comes from snapper.jpg beginning at coordinates (128,192). The rest of the image is from red_cow.jpg. Use *imshow()* to draw the image. Hint (remember this is a color image).

(e) (**5 points)** What are the maximum and minimum values of the pixels in **grayscale** versions of images red_cow and snapper.

(f) (**5 points)** Execute the following commands in Matlab:

```
f = imread('pout.tif');
```

Calculate and plot the images corresponding to $g^{0.5}(x,y)$ and $g^{1.5}(x,y)$. Comment on the brightness level and quality of the two images. Note that you may have to convert the image to the double data type (use *im2double())* depending on how you choose to solve this problem. Another thing to note is the maximum and minimum pixel values of the image g. What are min/max values of g before scaling?

Graduate students (5 pts.) Alternative Technique- You must study the intrans.m function that I provided for you on the class website with this project. I would like you to pay attention to the use of argument checking (use of nargchk, error, strcmp, and isfloat). Explain the use of "revertclass". The intent of this exercise is that you develop an understanding of the coding methods used for the function. **Use** this function as an alternative approach to implement the requested operations for this problem (This is *in addition to the solution* you provided for (f) above).

2. Image quantization and scaling:

(a) **(10 points)** Quantize the grayscale version of the image snapper to 8 levels and plot it. Explain the method you chose and how it works. Thus if you chose to use built in functions you need to explain how they operate.

(b) (**10 points)** Scale down the image red_cow to a maximum size of approximately 64 pixels by 43 pixels by appropriately sampling every nth pixel in both the horizontal and vertical directions (you may add extra rows and columns to the image to make it an easier size to work with). Plot your result.

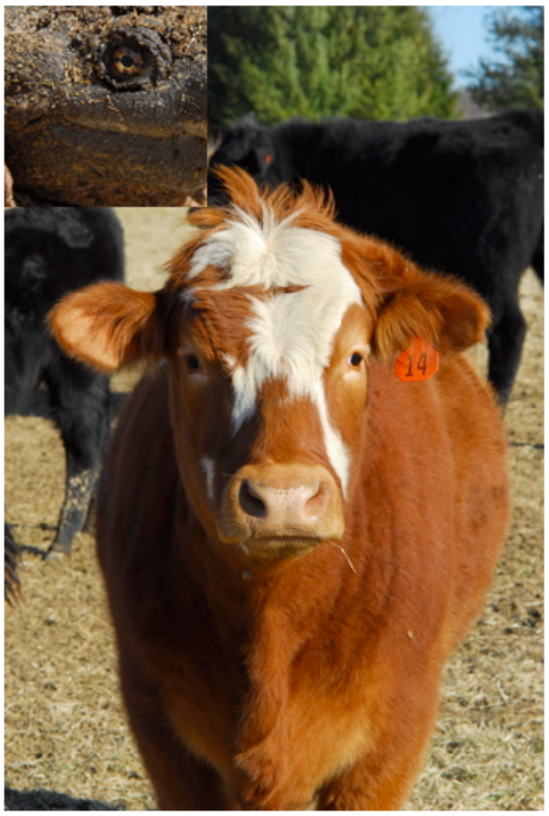


Figure 1. Red Cow with Snapper sub-image

(c) **Graduate students (5 pts.)** Repeat exercise (b) using the **imresize** function in Matlab rather than the sub-sampling approached used earlier. You can experiment with different interpolations to see which gives you the most visually pleasing results. Describe your results and why the latter image looks better. After plotting both reduced images side-by-side, use imresize again on both of the reduced image samples (part b and c) but this time - enlarge both images back to their original size using bicubic interpolation. Once again compare the results and comment on what you learned with this exercise.

3. Geometric Rotations and Affine Homogeneous Transforms

This problem is intended to help you review the materials we explored on 2D and 3D homogeneous transforms and Affine operations and demonstrate your understanding of the principles involved. You should review the multiple demo Matlab mlx files that we reviewed in class and which are posted on Blackboard (Ex3DRotation.mlx, Ex3DRotation2.mlx, Affine_test.mlx).

<u>Overview</u>

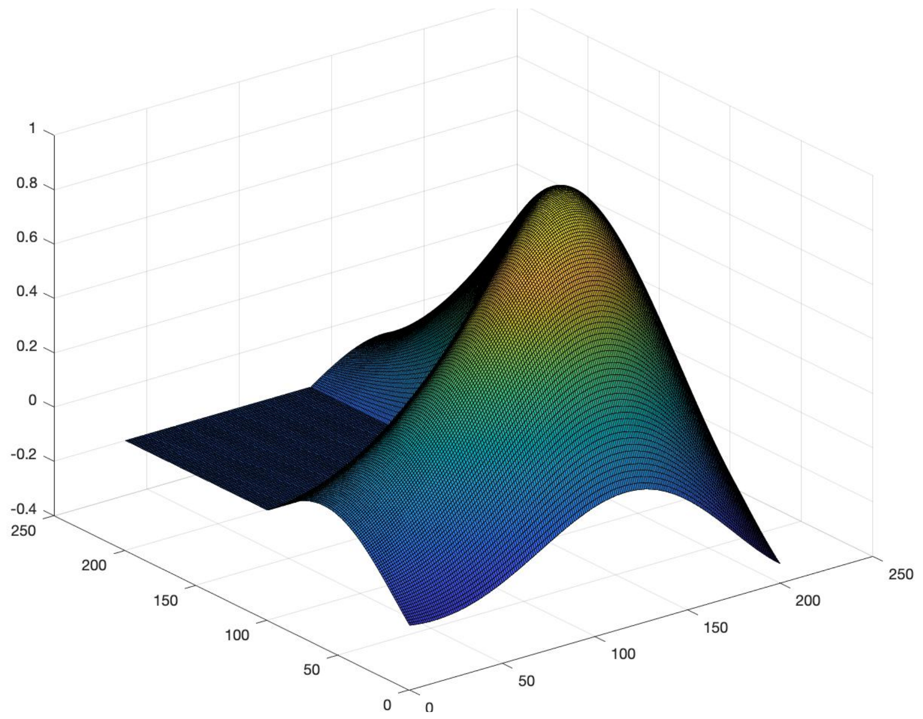
A two-dimensional geometric transformation is a mapping that associates each point in a Euclidean plane with another point in a Euclidean plane. The same is true for 3D transformations except one works with x,y,z in three-dimensional space.

For the first part of this exercise you will implement a series of 3D rotations on a the Matlab Logo function.

(a) **(10 points)**

- Create a surface plot of the Matlab Logo by using the built-in "membrane" command:

```
L=membrane(1,100);
surf(L);
```



Matlab Logo Surface Plot

- Explore the Matlab "rotate" command and use it to rotate this surface by -90 degrees (recall positive angles denote CCW rotations) about the x-axis and then -45 degrees about the z-axis (using the fixed axes approach) at which point one should be peering approximately into the underside of the membrane plot. You should explore the surface(), surf(), meshgrid and similar commands as you work through this.

- **Graduate Students (5 points)**. Repeat this exercise using individual rotation matrices (similar to the demos Ex3DRotation.mlx and Affine_test.mlx) to directly manipulate the xyz coordinates and then plot in 3D. Explain your chosen approach (fixed, moving, etc.) and comment on the results relative to the results you received from the Rotate-command work above.

(b) (5 Points) Now, for the second part of this exercise, you will implement a couple of affine transformations on a Cherry Blossom image (provided with this project). **Undergraduates** need to first convert this image to a grayscale image whereas G**raduate** students should process the image in RGB color.

- Use the imwarp() function and the Matlab transform format: $[u \;\; v] = [x \;\; y \;\; 1]T$ for these operations. Consider a review of the useful Matlab Reference on Transformations: Matlab Ref. But again recall, that the transforms given are Transposes of those provided in our notes and the Kay paper because of the different order of operations implemented by Matlab ([x,y,z,1]*T1 in Matlab vs T2*[x,y,z,1]′ where T1=T2′).
- Load the Cherry.jpg image into the workspace and crop it to a square image (768x768 or 512x512 for example).
- Using the affine2D() function and transform matrices you specify, implement a horizontal shear operation on the Cherry image using a value of 0.1.
- Using the affine2D() function and transform matrices you specify, implement a combined horizontal (-0.1) and vertical shear (0.2).
- Matlab transform format: $[u \;\; v] = [x \;\; y \;\; 1]T$ and the general approach indicated.

(c) Graduate Students (5 points). Create and implement an affine transform that creates a “mirror” image of the input. Explain how you devised your solution.

Instructions for submitting Matlab exercises Recap:
Your report for this project should once again take advantage of the mlx/m-file publishing features of Matlab. You must submit a pdf version of your solution. Compress your final folder (with the published pdf) **and** your final “unpublished” m-file or mlx-file (zip, tar, rar etc.) into one file and upload it to the Blackboard assignment for Project1.

Your report then should thus be written as an m-file with detailed comments, multiple typefaces, bullets and headings etc. to explain each of your results. Be sure to include a header which identifies you along with the date and project number.

When commenting your mlx/m-file please pay careful attention to providing sufficient detail as described below.

Possible Difficulties with creating a PDF report:
Because we are dealing with a lot of images which are essentially very large matrices, you may have some difficulties publishing to a pdf file. First, note that it may take a minute or

so to do all of the computations and picture generation. Also, if you have interactive portions of code (like requesting profile points on an image part 1c) you will have to "interact" to select the points during the export process.

Secondly, if your final pdf file cannot be opened, is of size 0-bytes, or you receive a Java error message during the publish process, you probably need to increase your Java Heap size to handle the large matrices (I had this problem for this project solution).

For now you can try to increase the Java Heap Memory in MATLAB. See the link below. I expect an increase of 50% would be sufficient."
Link to Java-heap resizing instructions

Report Methods:
Briefly describe your algorithms and any other relevant implementation details. This should probably be organized on a problem by problem basis, or grouped for similar problems.

Also, in cases where you have written complex code, present the structure of your code and how to run your programs. Provide pseudo-code or flowcharts as separate documents (or imported as part of and mlx file) when appropriate and/or event sequences to assist in describing the operation of your programs. This should be organized on a problem by problem basis.

Results:
Give detailed observations on all of your results.

Ask yourself the following questions:

- → Are the results what you were expecting they would be?
- → Do you think your results are completely correct?
- → If not, why not?
- → Do you think it's because the algorithm is not the appropriate one, or because your implementation of it has a bug? -- It is important that you realize when your results are wrong, that means that you understood the theory behind the project content.

In some cases, credit is given for incorrect results when the error is pointed out and thorough explanations are given for the incorrect results along with what you think are possible corrections or solutions.

Conclusions:
Write your conclusions for each major problem and/or for your overall project in a clear an concise way. Remember, **conclusions are not observations**. You draw conclusions from what you observe in your results. This is a very important part of your project, so think a lot about what you want to write in here.

NOTE:
In all cases when you are reading or writing an image in your matlab program, do not use a full directory path. Rather, just use the filename (example, IM=imread('lena.tif');). In this case Matlab will attempt to read and write the file from/to the current directory. So, as long as you begin by setting your current directory to your personal working directory -- on

drive Z for the ECE machines (use the current directory/workspace window on the matlab screen), your program will work well. This approach will also allow me to test execute your programs by simply copying them to my working directory and running them.

Please name your final archive file in a distinctive fashion using your name. For example: **Prj1_yourlastname.zip** (e.g. my submission for HW#1 must be hw1_paulik.zip). Please use the Assignment Upload feature on Blackboard to submit your file.