EE141L: Lab 1 – Image Manipulation

August 31, 2018

1 Matlab exercises: image manipulation

In this lab you will answer questions found in this document (look for Q1,Q2...) by running the provided Matlab script and adding your own modifications to the code. You are expected to submit both 1) your matlab code 2) and answers in a lab report. Both are to be uploaded on Blackboard every Friday before 10 AM. Hard copies of your report and or scripts will not be accepted in class. Lab 1 will be due September 7th. Important things to keep in mind:

- MATLAB scripts should run without any errors.
- Figures and results from MATLAB calculations should exactly match those stated in your report (we will check)
- Provide documentation for any function you write (i.e. inputs, outputs, algorithm)
- Lab reports are to be written in Q&A Format
- For all calculations, you will be asked to provide observations as well as justifications for answers
- Figures should have proper labels and titles or captions

Failure to complete any of the above will negatively impact your grade for lab assignments.

1.1 Getting files

Create a working folder, e.g. "C:\Users\YourName\Documents\ee141l\lab2\code". Download a 512×512 color image from `http://sipi.usc.edu/database/database.php?volume=misc` and the Matlab script file **lab2.m** from blackboard. Put both of them in your working folder.

1.2 Loading image

Open Matlab, select the working folder that you created as your current directory, and right click in the image file, and rename it to **picture.tiff**. Then open the **lab2.m** file, you should see

Listing 1: Script 1

```
clear;
filename = 'picture.tiff'; %this is the name of an image in your
    current folder
fmt = 'tiff';                %image format, e.g. jpe, png,  tiff
im = imread(filename, fmt);
imGray = rgb2gray(im);          %convert rgb image into gray scale
```

Run script by typing in command window **run lab2.m**

Q1: Check the sizes of **im, imGray** variables, what makes them different?. What is an uint8, and what is its range of values?. Hint: use functions **size, imshow**, look at the workspace.

1.3 Extracting a rectangular block

To extract a part of the image, we will use indices. Indices must be positive integers. The code **pixel = imGray(i,j)** stores the value at i-th row and j-th column in variable **pixel**. To extract a block of pixels we will use arrays of indices.

Creating arrays of indices in Matlab is easy, test the following code by typing it in the command window

- **indices1 = 1:1:10;** this will create the row vector **[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]**
- **indices2 = 5:3:27;** this will create the vector **[5, 8, 11, 14, 17, 20, 23, 26]**
- **indices3 = 1:10;** this is an abbreviated way of writing **indices1 = 1:1:10;**.
- Type **doc :** in the command window for more details on indexing and creating regularly spaced vectors .

The code **block = imGray(starti:endi, startj:endj)** uses two arrays of indices to extract of elements from a matrix.

Now uncomment (remove % as necessary) so you can run the following code.

Listing 2: Script 2

```
clear;
filename = 'picture.tiff'; %this is the name of an image in your
    current folder
fmt = 'tiff';                %format of the image, e.g. jpe, png,
    tiff,
im = imread(filename, fmt);

```

```
imGray = rgb2gray(im);

[m, n] = size(imGray); %matrix imGray has m rows, and n columns

%extracting an image block using indices
starti = floor(m/2)+1;
endi = floor(starti + m/4)-1;

startj = 1;
endj = floor(startj +n/4)-1;

block = imGray(starti:endi, startj:endj);

zero_padded_block = uint8(zeros(m,n));
zero_padded_block(starti:endi,startj:endj) = block;
```

Q2: What makes **block** and **zero_padded_block** different?. what is the size of the block as a function of **m, n**? Hint: again use functions **size, imshow**

1.4 Extracting multiple rectangular blocks using a for loop

Now I have added a for loop that will extract rectangular blocks and display them as images.

A **for** loop is a way of repeating an operation.

- the syntaxis **for t=1:nI**..... code... **end**, means the code inside the loop will be repeated **nI** times.
- The variable **t** is the loop index/counter, and it will increase its value by one at the end of the iteration.
- the variable **nI** is the number of times the code inside the loop will run. At the last iteration the loop index will take the value **nI**.

Umcomment necessary lines to run the following code

Listing 3: Script 3

```
clear;

filename = 'picture.tiff'; %this is the name of an image in your
   current folder
fmt = 'tiff';                %image format, e.g. jpe, png, tiff,
im = imread(filename, fmt);

imGray = rgb2gray(im);

[m, n] = size(imGray); %matrix imGray has m rows, and n columns

```

```
%extracting an image block using indices
starti = floor(m/2)+1;
endi = floor(starti + m/4)-1;

startj = 1;
endj = floor(startj +n/4)-1;

block = imGray(starti:endi, startj:endj);
zero_padded_block = uint8(zeros(m,n));
zero_padded_block(starti:endi,startj:endj) = block;

%extracting many image blocks using a for loop
block_size = 128;
IndicesI = 1:block_size:m;
nI = length(IndicesI);

for t=1:nI

    starti = IndicesI(t);
    endi   = starti + block_size-1;
    endi   = min(endi,m);
    block  = imGray(starti:endi,1:n);
    %zero_padded_block = uint8(zeros(m,n));
    %zero_padded_block(starti:endi,1:m) = block;

    figure;
    imshow(block);
    %imshow(zero_padded_block);

end
```

Q3 What are the displayed images, what are their dimensions?

Q4 Uncomment (remove % from) lines 33,34, 38, comment line 37, and run code again. Describe displayed images, and compare with images from **Q3**.

Q5 What is the span of the set of obtained zero padded images from Q4?, what do you get if you add all the zero padded images?

1.5 Write your own code to extract multiple blocks

Q6 Write code that displays zero padded blocks of size **m x 128**. Hint: copy/paste the for loop portion and modify.

Q7 Write code that displays zero padded blocks of size [256 x 256] that comprise the whole image (Hint: fill in code below).

Listing 4: Script 5

```
1
2  block_size = 256;
3  IndicesI = 1:block_size:m;
4  IndicesJ =
5  nI = length(IndicesI);
6  nJ =
7  for t=1:nI
8      for s=1:nJ
9          starti = IndicesI(t);
10         endi   = starti + block_size-1;
11         endi   = min(endi,m);
12
13         startj =
14         endj   =
15         endj   =
16         block  =
17         zero_padded_block = uint8(zeros(m,n));
18         zero_padded_block(,) =
19
20         figure;
21         imshow(zero_padded_block);
22
23     end
24 end
```