

EN-74 ECE: Introduction to Image Processing  
Tufts University  
Fall 2007  
Problem Set 2: Solutions  
Due Sept. 20, 2007

1. Write two Matlab functions that convert between row column and lexicographic ordering for indexing into an image. The functions should have the form

```
l = rc2lex(m,n,M,N);  
[m,n] = lex2rc(l,M,N);
```

where

- $l$  = lexicographic index of the pixel of interest
- $m, n$  = row and column indices of the pixel of interest
- $M, N$  = number of rows and columns in the image

Prove with some examples that your functions work correctly

rc2lex.m

```
function l = rc2lex(m,n,M,N)  
  
% Given the row and columns, the lexicographic index of the pixel is n-1  
% times the number of rows in the image (M) plus m. Indeed, (n-1)*M is the  
% total number of pixels "skipped" in the first n columns of the image. To  
% this we must add m which tells us how far down the n-th column we are.  
  
l = (n-1)*M + m;
```

lex2rc.m

```
function [m,n] = lex2rc(l,M,N)  
  
% Let's start with an example that we can do by hand. If M = 4, N = 3, and  
% l = 10 then n = 3 and m = 2.  
  
% How do we know n = 3? Well, there are M=4 rows in the image, so to get  
% to pixel l=10 pixel we have to traverse two full columns but not a third  
% since 2*4=8 which is less than 10 but 3*4 = 12 which would mean we have  
% passed pixel 10. More generally, we get n as the integer part of l/M  
% plus 1. In our case this the integer part of 10/4 is 2 plus one is  
% three. In Matlab, the integer part of l/M is obtained using the "fix"  
% function so we have  
  
n = fix(l/M) + 1;  
  
% Once we know n, we use the result of the rc2lex discussion to find m.  
% Specifically, from rc2lex we know l = (n-1)*M + m so m = l - (n-1)*M  
  
m = l - (n-1)*M;
```

See solution in file ps02.m for an example showing that these functions work properly.

2. Exercises 2.1, 2.2, and 2.3 in the McAndrew text

Exercise 2.1

Typing

```
help imdemos
```

in Matlab produced the following listing of TIFF formatted pictures

```
autumn.tif          board.tif           cameraman.tif  
canoe.tif           cell.tif           circbw.tif
```

circuit.tif	eight.tif	forest.tif
kids.tif	logo.tif	m83.tif
moon.tif	mri.tif	paper1.tif
pout.tif	shadow.tif	spine.tif
tire.tif	trees.tif	

To obtain the required information for these pictures the you should use the `imfinfo` function e.g.

```
>> imfinfo('autumn.tif')
ans =
    Filename:
    '/Applications/MATLAB704/toolbox/images/indemos/autumn.tif'
    FileModDate: '04-Dec-2000 13:57:54'
    FileSize: 213642
    Format: 'tif'
    FormatVersion: []
    Width: 345
    Height: 206
    BitDepth: 24
    ColorType: 'truecolor'
    FormatSignature: [73 73 42 0]
    ByteOrder: 'little-endian'
    NewSubfileType: 0
    BitsPerSample: [8 8 8]
    Compression: 'Uncompressed'
    PhotometricInterpretation: 'RGB'
    StripOffsets: [30x1 double]
    SamplesPerPixel: 3
    RowsPerStrip: 7
    StripByteCounts: [30x1 double]
    XResolution: 72
    YResolution: 72
    ResolutionUnit: 'Inch'
    Colormap: []
    PlanarConfiguration: 'Chunky'
    TileWidth: []
    TileLength: []
    TileOffsets: []
    TileByteCounts: []
    Orientation: 1
    FillOrder: 1
    GrayResponseUnit: 0.0100
    MaxSampleValue: [255 255 255]
    MinSampleValue: 0
    Thresholding: 1
```

From this we see

- The image `autumn.tif` is truecolor
- The image has 206 rows (height = 206) and 345 columns (width = 345)
- Executing the commands `a = imread('autumn.tif');` ; `imshow(a)` shows us that we are looking at the picture of some trees by a pond during the fall.

Analogous steps can be used on the other images to obtain the information requested in the problem.

### Exercise 2.2

The code for this is in the M file. The results are:

```
TIF file size = 65240
JPG file size = 10717
PNG file size = 38267
BMP file size = 66614
```

So, we see that JPG, which is pretty aggressive with the compression it uses results in the smallest files.

### Exercise 2.3

The code for this is in the M file. The results are:

True color image

```
TIF file size = 213642
JPG file size = 12736
PNG file size = 118786
BMP file size = 213470
```

Binary image

```
TIF file size = 3822
JPG file size = 18988
PNG file size = 2068
BMP file size = 10142
```

Color indexed image

```
TIF file size = 68180
JPG file size = 28683
PNG file size = 61530
BMP file size = 67978
```

For all of these, JPEG yields the smallest file sizes. In the case of true color images, the results are really significant.

3. Load the cameraman image into Matlab and, by trial and error extract the sub-image that corresponds to his face. To be a bit more specific, suppose that the variable `cman` holds the full image. The problem here is to find the variable `r_start`, `r_end`, `c_start`, `c_end` such that the following code snippet displays the face

```
face = cman(r_start:r_end, c_start:c_end)
imshow(face)
```

See solution in file `ps02.m`