Previous class:
- Color vectors – RGB
- 1-dimensional array – vector

Now:
- Play with image files
- 2-dimensional array — matrix

Grayness: a value in [0..255]

0 = black
255 = white

Creating a matrix
- Built-in functions: ones, zeros, rand
- E.g., zeros(2,3) gives a 2-by-3 matrix of 0s
- "Build" a matrix using square brackets, [ ], but the dimension must match up:
  - [x y] puts y to the right of x
  - [x; y] puts y below x
  - [4 0 3; 5 1 9] creates the matrix [4 0 3; 5 1 9]
  - [4 0 3; ones(1,3)] gives [4 0 3; 1 1 1]
  - [4 0 3; ones(3,1)] doesn't work

2-d array: matrix
- An array is a named collection of like data organized into rows and columns
- A 2-d array is a table, called a matrix
- Two indices identify the position of a value in a matrix, e.g.,
  \[ \text{mat}(r, c) \]
  refers to component in row \( r \), column \( c \) of matrix \( \text{mat} \)
- Array index starts at 1
- Rectangular: all rows have the same # of columns

% What will \( M \) be?
\[ M = \begin{bmatrix} \text{ones}(1,3) \mid 1:4 \end{bmatrix} \]

A

\[
\begin{bmatrix}
1 & 1 & 1 & 0 \\
1 & 2 & 3 & 4
\end{bmatrix}
\]

B

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 2 & 3
\end{bmatrix}
\]

C  Error – \( M \) not created
What is \( [7 \ 0 \ 5]' \) ?

\[ \begin{align*}
A & : \text{Same as } [5 \ 0 \ 7] \\
B & : \text{Same as } [7; \ 0; \ 5] \\
C & : \text{Same as } [5; \ 0; \ 7]
\end{align*} \] 

What will \( A \) be?

\[ \begin{align*}
A & = [1 \ 1] \\
A & = [A' \ \text{ones}(2,1)] \\
A & = [1 \ 1 \ 1 \ 1; \ A \ A]
\end{align*} \]

\[ \begin{align*}
A & : \text{3-by-4 matrix} \\
B & : \text{4-by-3 matrix} \\
C & : \text{vector of length 12} \\
D & : \text{Error}
\end{align*} \]

Working with a matrix:

- **size** and individual components

Given a matrix \( M \):

\[
\begin{array}{cccc}
2 & -1 & 5 & 0 \\
3 & 8 & 6 & 7 \\
5 & -3 & 8.5 & 9.10 \\
52 & 81 & 5.7 & 2
\end{array}
\]

\[\text{[nr, nc]}= \text{size}(M) \quad \% \text{ nr is # of rows,} \]
\[\% \text{ nc is # of columns}\]

\[\begin{align*}
M(2,4) & = 1; \\
\text{disp}(M(3,1)) \\
M(1,nc) & = 4;
\end{align*} \]

Images can be encoded in different ways

- **Common formats include**
  - JPEG: Joint Photographic Experts Group
  - GIF: Graphics Interchange Format
- **Data are compressed**
- **We will work with jpeg files:**
  - **imread**: read a .jpg file and convert it to a "normal numeric" array that we can work with
  - **imwrite**: write an array into a .jpg file (compressed data)

Let's put a picture in a frame

- **Read a grayscale jpeg file into a matrix \( P \)**
  \[P = \text{imread('}<filename>\text{.jpg});\]
- **See the image represented by \( P \)**
  \[\text{imshow}(P)\]
- **Change the "edge pixels" into the frame color (grayscale) you want**
  \[
  \ldots
  \]

Problem: produce a negative
Problem: produce a negative

- "Negative" is what we say, but all color values are positive numbers!
- Think in terms of the extremes, 0 and 255. Then the "negative" just means the opposite side.
- So 0 is the opposite of 255;
  - 1 ... 254;
  - 5 ... 250;
  - 30 ... 225;
  - x ... 255-x

A color picture is made up of RGB matrices

<table>
<thead>
<tr>
<th>Color image</th>
<th>3-d Array</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="LawSchool.jpg" alt="Color Image" /></td>
<td><img src="LawSchoolMirror.jpg" alt="3-d Array" /></td>
</tr>
</tbody>
</table>

Operations on images amount to operations on matrices—good way to practice matrix manipulation!

Extracting subarrays and tiling

- Accessing a submatrix: $M(\ldots, \ldots, \ldots)$
- Accessing a subarray (3-d): $P(\ldots, \ldots, \ldots)$
- Concatenate horizontally: $[ PL \  PR ]$
- Concatenate vertically: $[ PT; PB ]$

Your multi-media project

- Create a Matlab program that involves image and sound manipulation
- You get to
  - Make your own design
  - Set the level of difficulty
- Finish by 10:00pm Thursday and submit in CMS

Example: Mirror Image

LawSchool.jpg  LawSchoolMirror.jpg

Solution Framework

1. Read LawSchool.jpg from memory and convert it into an array.
2. Manipulate the Array.
3. Convert the array to a jpg file and write it to memory.
Reading and writing jpg files

% Read jpg image and convert to % a 3D array A
A = imread('LawSchool.jpg');

% Write 3D array B to memory as % a jpg image
imwrite(B,'LawSchoolMirror.jpg')

A 3-d array as 3 matrices

\[ [nr, nc, np] = \text{size}(A) \]\n\#rows #columns #layers (pages)

4-by-6 \ M1= A(:,:,1)
4-by-5 \ M2= A(:,:,2)
4-by-6 \ M3= A(:,:,3)

% Make mirror image of A
[nr,nc,np]= size(A);
for r= 1:nr
    for c= 1:nc
        B(r,c )= A(r,nc-c+1 );
    end
end

% Make mirror image of A -- the whole thing
A= imread('LawSchool.jpg');
[nr,nc,np]= size(A);
B= zeros(nr,nc,np);
B= uint8(B); % Type for image color values
for r= 1:nr
    for c= 1:nc
        for p= 1:np
            B(r,c,p)= A(r,nc-c+1,p);
        end
    end
end
image(B) % Show 3-d array data as an image
imwrite(B,'LawSchoolMirror.jpg')

Turn the white duck yellow!
- The duck’s body and the image’s background show some contrast. However, neither the duck’s body nor the background has a uniform color.
- Are the RGB values different enough for us to write a “rule” in the program to tell between the duck and the background?
- Check out the RGB values!