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

Things to do:
1. Read `bwduck.jpg` from memory and convert it into an array
2. Show the original picture
3. Assign a gray value (frame color) to the “edge pixels”
4. Show the manipulated picture

Reading a jpeg file and displaying the image

```matlab
% Read jpg image and convert to an array P
P = imread('bwduck.jpg');

% Show the data in array P as an image
imshow(P)
```
% Frame a grayscale picture
P = imread('bwduck.jpg');
imshow(P)

% Change the "frame" color
width = 50;
frameColor = 200;  % light gray
[nr,nc] = size(P);
for r = 1:nr
    for c = 1:nc
        % At pixel (r,c)
    end
end
imshow(P)

Things to consider…
1. What is the type of the values in P?
2. Can we be more efficient?
See pictureFrame*.m

Accessing a submatrix
M
| 2 3 5 0 3 |
| 3 8 6 7 7 |
| 5 -3 8.5 9 10 |
| 52 81 .5 7 2 |

- M refers to the whole matrix
- M(3,5) refers to one component of M
- M(2:3,3:5) refers to a submatrix of M

A color picture is made up of RGB matrices \( \rightarrow \) 3-d array

E.g., color image data is stored in a 3-d array \( A \):

\[
0 \leq A(i,j,1) \leq 255 \\
0 \leq A(i,j,2) \leq 255 \\
0 \leq A(i,j,3) \leq 255
\]

A color picture is made up of RGB matrices \( \rightarrow \) 3-d array

Operations on images amount to operations on matrices!

Example: Mirror Image

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

\[
\begin{align*}
\text{size}(A) &= [nr, nc, np] \\
A(1:nr,1:nc,1) &= \text{M1} \\
A(1:nr,1:nc,2) &= \text{M2} \\
A(1:nr,1:nc,3) &= \text{M3}
\end{align*}
\]

%Store mirror image of A in array B

\[
\begin{align*}
[nr, nc, np] &= \text{size}(A) \\
\text{for } r &= 1:.nr \\
&\quad \text{for } c &= 1:nc \\
&\quad \quad B(r,c) = A(r,nc-c+1) \\
\end{align*}
\]

Vectorized code simplifies things…

Work with a whole column at a time

\[
\begin{align*}
\text{for } r &= 1:nr \\
&\quad \text{for } c &= 1:nc \\
&\quad \quad \text{for } p &= 1:np \\
&\quad \quad \quad B(r,c,p) = A(r,nc-c+1,p)
\end{align*}
\]
Vectorized code to create a mirror image

\[
A = \text{imread}(\text{\textquote{LawSchool.jpg}})\\
[nr,nc,np] = \text{size}(A);\\
\text{for } c = 1:nc\\
B(:,c,1) = A(:,nc-c+1,1)\\
B(:,c,2) = A(:,nc-c+1,2)\\
B(:,c,3) = A(:,nc-c+1,3)\\
\text{end}\\
\text{imwrite}(B,\text{\textquote{LawSchoolMirror.jpg}})\\
\]

Even more compact vectorized code to create a mirror image...

\[
\text{for } c = 1:nc\\
B(:,c,1) = A(:,nc-c+1,1)\\
B(:,c,2) = A(:,nc-c+1,2)\\
B(:,c,3) = A(:,nc-c+1,3)\\
\text{end}\\
B = A(:,nc:-1:1,:)
\]

Example: color → black and white

Can “average” the three color values to get one gray value.

Averaging the RGB values to get a gray value

\[
\text{for } i = 1:m\\
\text{for } j = 1:n\\
M(i,j) = .3*R(i,j) + .59*G(i,j) + .11*B(i,j)\\
\text{end}\\
\text{end}\\
\]

Scalar operation
Averaging the RGB values to get a gray value

\[ M = 0.3R + 0.59G + 0.11B \]

Here are 2 ways to calculate the average. Are gray value matrices \( g \) and \( h \) the same given image data \( A \)?

```matlab
for r = 1:nr
    for c = 1:nc
        g(r,c) = A(r,c,1)/3 + A(r,c,2)/3 + ... + A(r,c,3)/3;
        h(r,c) = (A(r,c,1) + A(r,c,2) + A(r,c,3))/3;
    end
end
```

A: yes  B: no

showToGrayscale.m

Matlab has a built-in function to convert from color to grayscale, resulting in a 2-d array:

\[ B = \text{rgb2gray}(A) \]