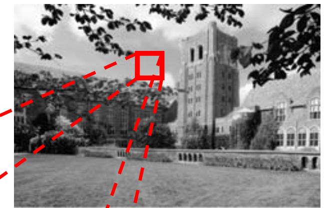
- Previous class:
  - Play with sound files
  - Practice working with vectors
- Now:
  - Play with image files
  - 2-dimensional array—matrix

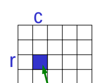
A picture as a matrix—2-dimensional array

1458-by-2084



150	149	152	153	152	155
151	150	153	154	153	156
153	151	155	156	155	158
154	153	156	157	156	159
156	154	158	159	158	161
157	156	159	160	159	162

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.,

  `mat(r,c)`

  refers to component in row r, column c of matrix mat
- Array index starts at 1
- Rectangular: all rows have the same #of columns

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 $\begin{bmatrix}4&0&3\\5&1&9\end{bmatrix}$
  - [4 0 3; ones(1,3)] gives $\begin{bmatrix}4&0&3\\1&1&1\end{bmatrix}$
  - [4 0 3; ones(3,1)] doesn't work


```
% What will M be?
M = [ones(1,3); 1:4]
```

A
```
1 1 1 0
1 2 3 4
```

B
```
1 1 1
1 2 3
```

C *Error – M not created*

What will **A** be?

```
A= [1  1]
A= [A'  ones(2,1)]
A= [1  1  1;  A  A]
```

A 3-by-4 matrix

B 4-by-3 matrix

C vector of length 12

D *Error*

Working with a matrix: `size` and individual components

Given a matrix M

2	-1	.5	0	-3
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

```
[nr, nc]= size(M)   % nr is #of rows,
                    % nc is #of columns
M(2,4)= 1;
disp(M(3,1))
M(1,nc)= 4;
```


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)

Grayness: a value in [0..255]

0 = black
255 = white

These are *integer* values
Type: `uint8`

150	149	152	153	152	155
151	150	153	154	153	156
153	151	155	156	155	158
154	153	156	157	156	159
156	154	158	159	158	161
157	156	159	160	159	162

Let's put a picture in a frame

- Read a grayscale jpeg file into a matrix P
  P = imread('<filename>.jpg');
- See the image represented by P
  imshow(P)
- Change the "edge pixels" into the frame color (grayscale) you want
  …

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;

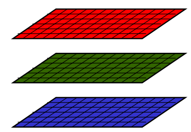
1	…	254;
5	…	250;
30	…	225;
x	…	255-x

```
function newIm = toNegative(im)
% newIm is the negative of image im
% im, newIm are 3-d arrays; each component is uint8

[nr,nc,np]= size(im);    % dimensions of im
newIm= zeros(nr,nc);     % initialize newIm
newIm= uint8(newIm);     % Type for image color values

for r= 1:nr
    for c= 1:nc
        for p= 1:np
            newIm(r,c,p)= _______________;
        end
    end
end
```

15

A color picture is made up of RGB matrices

Color image   3-d Array



0 ≤ A(i,j,1) ≤ 255

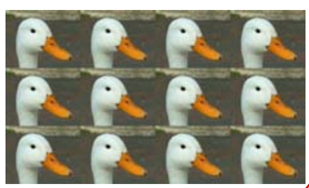
0 ≤ A(i,j,2) ≤ 255

0 ≤ A(i,j,3) ≤ 255

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

18

Extracting subarrays and tiling



rows columns all layers

- Accessing a submatrix: M(_:_ , _:_)
- Accessing a subarray (3-d): new= P(_:_ , _:_ , :)
- Concatenate horizontally: [PL PR]
- Concatenate vertically: [PT; PB]

19

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 11:30am and submit in CMS

Manipulate sound vector and playback

Mirror image

Sub-array

Photo negative

Set color conditionally

tiling

20

Example: Mirror Image

LawSchool.jpg

LawSchoolMirror.jpg

21

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.

22

Reading and writing jpg files

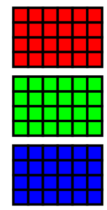
```
% Read jpg image and convert to
% a 3D array A
  A = imread('LawSchool.jpg');

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

23

A 3-d array as 3 matrices

```
[nr, nc, np] = size(A)  % dimensions of 3-d array A
```

#rows #columns #layers (pages)



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

24

```
% Make mirror image of A

[nr,nc,np]= size(A);
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
```

26

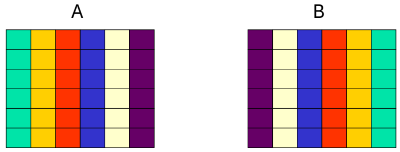
```
% Make mirror image of A -- the whole thing

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')
```

30

Vectorized code simplifies things…

Work with a whole column at a time

A B



1 2 3 4 5 6 1 2 3 4 5 6

Column c in B
is column nc-c+1 in A

41

Vectorized code to create a mirror image

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

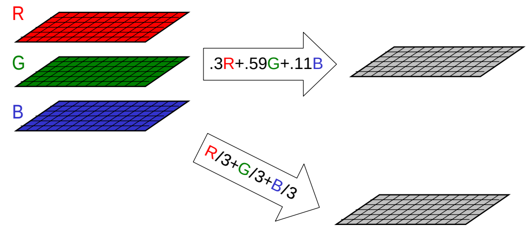
45

Example: color → black and white

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

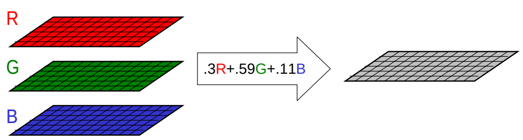
47

Averaging the RGB values to get a gray value



48

Averaging the RGB values to get a gray value

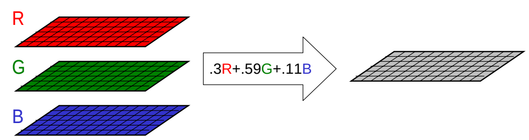



```
for  i= 1:m
   for j= 1:n
      M(i,j)= .3*R(i,j) + .59*G(i,j) + .11*B(i,j)
   end
end
```

scalar operation

49

Averaging the RGB values to get a gray value




```
M= .3*R + .59*G + .11*B
```

vectorized operation

50

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

```
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

52

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!

53