L23. Working with Image Files

```
imread, imwrite,
imshow, uint8,
rgb2gray
```

Pictures as Arrays

A black and white picture can be encoded as a 2D Array

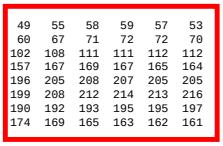
Typical:

0 (black) <= A(i,j) <= 255 (white)

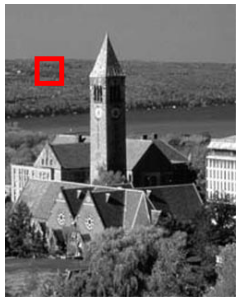
Values in between correspond to different levels of grayness.

Just a Bunch of Numbers

318-by-250



49	55	58	59	57	53
60	67	71	72	72	70
102	108	111	111	112	112
157	167	169	167	165	164
196	205	208	207	205	205
199	208	212	214	213	216
190	192	193	195	195	197
174	169	165	163	162	161



A Color Picture is 3 Arrays

Stack them in a 3D array.

Typical:

```
0 <= A(i,j,1) <= 255   (red)
0 <= A(i,j,2) <= 255   (green)
0 <= A(i,j,3) <= 255   (blue)
```

Note 3rd Subscript

Encoding Images

There are a number of file formats for images. Some common ones:

JPEG
(Joint Photographic Experts Group)

GIF
(Graphics Interchange Format)

Behind the scenes: compressing data

A Compression Idea

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Store the array (81 num's) or the purple vectors (18 num's)?

More Dramatic

Suppose A is a 1000-by 2000 times table.

Do I store A (2,000,000 numbers)

or

Do I store the two 1-dimensional multiplier arrays (3000 numbers) and "reconstruct" A

Images can be written as a sum of a relatively small number of times tables

1000-by-2000 picture might be well approximated by the sum of 100 times tables.

2,000,000 vs (100 x 3000)

Operations on Images

They amount to operations on 2D Arrays.

A good place to practice "array" thinking.

Two Problems

We have:

LawSchool.jpg

Problem 1

Want:

LawSchoolMirror.jpg

Problem 2

Want:

LawSchoolUpDown.jpg

Solution Framework

Read **LawSchool.jpg** from memory and convert it into an array.

Manipulate the Array.

Convert the array to a jpg file and write it to memory.

imread

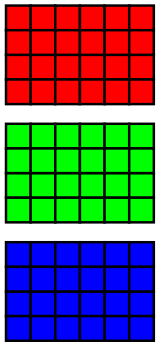
```
% Read image and convert to
% a 3D array…

  A = imread('LawSchool.jpg');
```

The 3D Array

```
>> [m,n,p] = size(A)
m =
        1458          rows
n =
        2084          columns
p =
           3          layers
```

The Layers



1458-by-2084 A(:,:,1)

1458-by-2084 A(:,:,2)

1458-by-2084 A(:,:,3)

Left-Right Mirror Image

```
A = imread('LawSchool.jpg')
[m,n,p] = size(A);
for j=1:n
    B(:,j,1) = A(:,n+1-j,1)
    B(:,j,2) = A(:,n+1-j,2)
    B(:,j,3) = A(:,n+1-j,3)
end
imwrite(B,'LawSchoolMirror.jpg')
```

Equivalent

```
for j=1:n
    B(:,j,1) = A(:,n+1-j,1)
    B(:,j,2) = A(:,n+1-j,2)
    B(:,j,3) = A(:,n+1-j,3)
end
```

```
B = A(:,n:-1:1,:);
```

The Upside Down Image

```
A = imread('LawSchool.jpg')
[m,n,p] = size(A);
for i=1:m
    C(i,:,1) = A(m+1-i,:,1)
    C(i,:,2) = A(m+1-i,:,2)
    C(i,:,3) = A(m+1-I,:,3)
end
imwrite(C,'LawSchoolUpDown.jpg')
```

Equivalent

```
for j=1:n
    C(i,:,1) = A(m+1-i,:,1)
    C(i,:,2) = A(m+1-i,:,2)
    C(i,:,3) = A(m+1-i,:,3)
end
```

```
    C = A(m:-1:1,:,:);
```

New Problem
Color → Black and White

Have:

New Problem
Color → Black and White

Want:

rgb2gray

```
A = imread('LawSchool.jpg');
bwA = rgb2gray(A);
imwrite(bwA,'LawSchoolBW.jpg')
```

How Does the Conversion Work?

r	g	b	gray
167	219	241	206
66	35	15	42
95	14	20	39
163	212	242	201
182	228	215	213
225	244	222	236
136	199	240	185

It's a complicated mapping

Why not take Average?

```
bwA = uint8(zeros(m,n))
for i=1:m
  for j = 1:n
     bwA(i,j) = ( A(i,j,1) +…
        + A(i,j,2) + A(i,j,3))/3;
  end
end
imwrite(bwA,'LawSchoolBW.jpg')
```

uint8 : unsigned 8-bit integer

Why not take Max?

```
bwA = uint8(zeros(m,n))
for i=1:m
  for j = 1:n
     bwA(i,j) = max([A(i,j,1) …
        A(i,j,2)  A(i,j,3)]);
  end
end
imwrite(bwA,'LawSchoolBW.jpg')
```

uint8 : unsigned 8-bit integer

Max:

Matlab:

Problem: Produce a Negative

Idea

If matrix A represents the image and

```
B(i,j) = 255 - A(i,j)
```

for all i and j, then B will represent the negative.