


- Previous Lecture:
 - 2-d array examples
- Today's Lecture:
 - Image processing
- Announcement:
 - Prelim 2 tonight 7:30-9pm



Accessing a submatrix

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

M

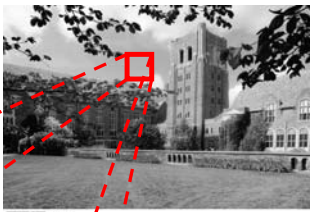
- **M** refers to the whole matrix
- **M(3, 5)** refers to one component of **M**
- **M(2:3, 3:5)** refers to a submatrix of **M**

row indices column indices

Lecture 15 7

A picture as a matrix

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

Lecture 15 10

Images can be encoded in different ways

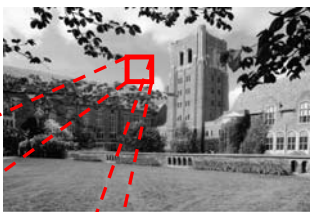
- Common formats include
 - JPEG: Joint Photographic Experts Group
 - GIF: Graphics Interchange Format
- Data are compressed
- We will work with jpeg files:
 - **imread**: read a .jpg file and convert it to a "normal numeric" array that we can work with
 - **imwrite**: write an array into a .jpg file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

Lecture 15 11

Grayness: a value in [0..255]

0 = black
255 = white

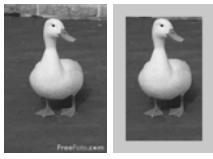
These are *integer* values
Type: **uint8**



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

Lecture 15 12

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

Lecture 15 13

Reading a jpeg file and displaying the image

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

% Show the data in array P as
% an image
imshow(P)
```

Lecture 15 14


```
% 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)
        if r<=width || r>nr-width || ...
            c<=width || c>nc-width
            P(r,c)= frameColor;
        end
    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
Lecture 15 18

A color picture is made up of RGB matrices → 3-d array



```
114 114 112 112 114 111 114 115 112 113
114 113 111 109 113 111 113 110 112 113
115 114 112 111 111 112 112 111 112 112
116 117 114 114 112 111 111 110 110 114
113 112 112 112 112 110 111 111 116 115
113 113 113 113 113 111 111 111 116 114
113 114 114 117 113 112 112 113 114 113
113 116 116 116 113 112 112 113 114 114
116 116 117 117 114 114 112 112 114 115

153 153 150 150 154 151 151 153 153 150 151
153 152 149 147 153 151 151 150 150 151
114 113 111 110 111 112 110 149 150 150
155 156 155 150 152 150 151 150 153 153
151 150 150 150 150 148 148 151 152 151
153 153 153 153 151 149 149 151 152 150
150 151 150 151 151 150 150 151 152 151
153 154 154 154 151 150 150 151 152 152
154 154 153 153 149 149 150 150 152 153

212 212 212 212 214 213 213 214 213 213
212 211 211 209 213 213 214 214 213 213
213 212 212 209 213 214 213 213 212
214 215 214 214 213 216 214 213 215 212
213 213 213 213 213 213 211 213 214 212
215 215 214 216 213 213 211 213 212 210
212 213 214 213 213 212 212 213 214 213
215 214 214 216 213 212 212 213 214 214
216 216 215 215 213 213 213 213 214 215
```

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

$$0 \leq A(i, j, 1) \leq 255$$

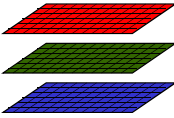
$$0 \leq A(i, j, 2) \leq 255$$

$$0 \leq A(i, j, 3) \leq 255$$

Lecture 15 20

A color picture is made up of RGB matrices → 3-d array

Color image



3-d Array

$$0 \leq A(i, j, 1) \leq 255$$



$$0 \leq A(i, j, 2) \leq 255$$

$$0 \leq A(i, j, 3) \leq 255$$

Operations on images amount to operations on matrices!

Lecture 15 21

Example: Mirror Image

LawSchool.jpg LawSchoolMirror.jpg

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.

Lecture 15 22

Reading and writing jpeg 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')
```

Lecture 15 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)

Lecture 15 24

%Store mirror image of A in array B

```
[nr,nc,np]= size(A);
for r= 1:nr
    for c= 1:nc
        B(r,c )= A(r,nc-c+1 );
    end
end
```

Lecture 15 25

```
[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
```

This is non-vectorized code.

```
[nr,nc,np]= size(A);
for p= 1:np
    for r= 1:nr
        for c= 1:nc
            B(r,c,p)= A(r,nc-c+1,p);
        end
    end
end
```

Both fragments create a mirror image of A.

A true
B false

Lecture 15 28

% 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
imshow(B) % Show 3-d array data as an image
imwrite(B,'LawSchoolMirror.jpg')
```

Lecture 15 30

Vectorized code simplifies things...
Work with a whole column at a time

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

Lecture 15 41

Consider a single matrix (just one layer)

```
[nr,nc,np] = size(A);
for c= 1:nc
    B( : ,c ) = A( : ,nc-c+1 );
end
```

The colon says "all indices in this dimension." In this case it says "all rows."

Lecture 15 44

Now repeat for all layers

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

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

```
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)
end
```



```
B = A(:,nc:-1:1,:)
```

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,'LawSchool.jpg')
```

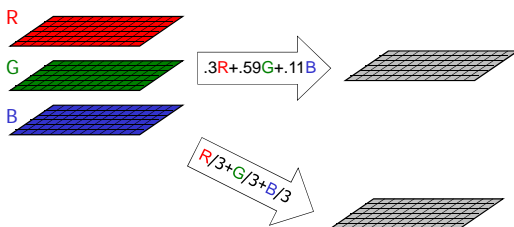
Can improve efficiency by initializing B to be a 3-d array of the appropriate size

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



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