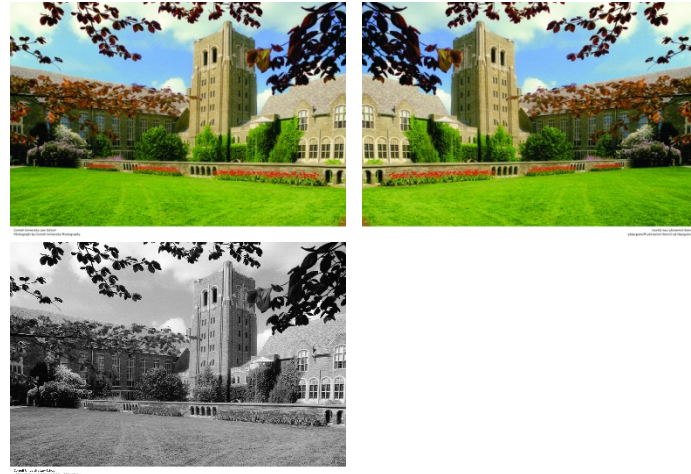


- Previous Lecture:
  - 2-d array examples

- Today's Lecture:
  - Image processing

- Announcements:

- Prelim 2 tonight 7:30-9pm
- Project 4 will be posted tonight. Due Monday Oct 27<sup>th</sup>



## Accessing a submatrix

**M**

2	-1	.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**

## Accessing a submatrix

**M**

2	-1	.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**

row indices



column indices



```
% Given an nr-by-nc matrix M.
```

```
% What is A?
```

```
for r= 1: nr  
    for c= 1: nc  
        A(c,r)= M(r,c) ;  
    end  
end
```

**A** A is M with the columns in reverse order

**B** A is M with the rows in reverse order

**C** A is the transpose of M

**D** A and M are the same

```
% Given an nr-by-nc matrix M.
```

```
% What is A?
```

```
for r= 1: nr
```

```
    for c= 1: nc
```

```
        A(c,r)= M(r,c) ;
```

```
    end
```

```
end
```

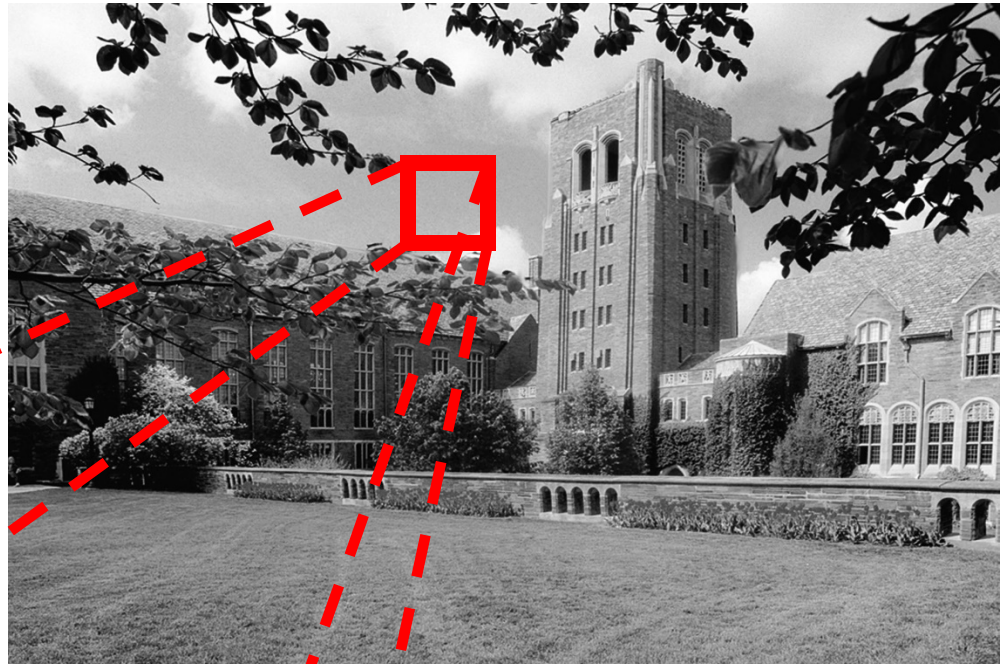
**M**

6	3	2	5
4	13	20	6
11	26	9	1

See transposeA.m

# A picture as a matrix

1458-by-2084



Cornell University Law School  
Photograph by Cornell University Photography

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

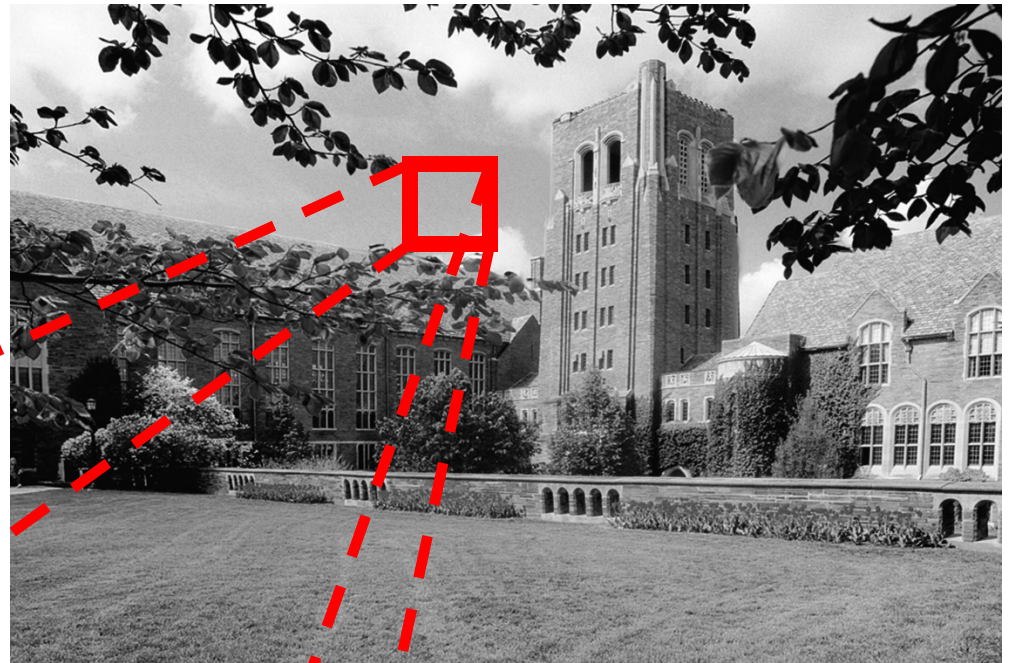
# Images can be encoded in different ways

- 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`

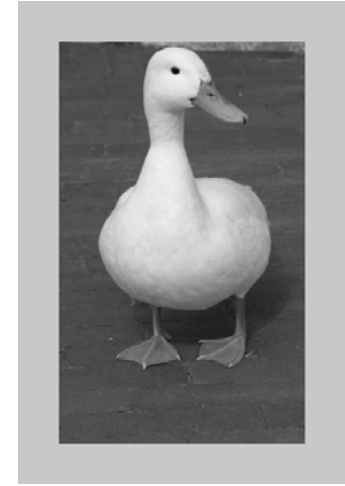
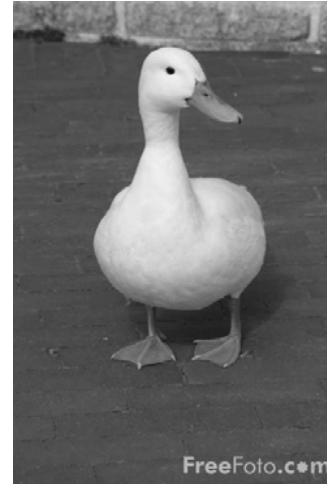


Cornell University Law School  
Photograph by Cornell University Photography

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



### 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

```
% 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
```

```
imshow(P)
```

```
% Frame a grayscale picture

P= imread('bwduck.jpg');
imshow(P)

% Change the "frame" color
width= 50;
frameColor= 200; % light gray
```

```
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)
```

```
% 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`

A color picture is made up of **RGB** matrices  $\rightarrow$  3-d array



```
114 114 112 112 114 111 114 115 112 113
114 113 111 109 113 111 113 115 112 113
115 114 112 111 111 112 112 111 112 112
116 117 116 114 112 115 113 112 115 114
113 112 112 112 112 110 111 113 116 115
115 115 115 115 113 111 111 113 116 114
112 113 116 117 113 112 112 113 114 113
115 116 118 118 113 112 112 113 114 114
116 116 117 117 114 114 112 112 114 115
```

```
153 153 150 150 154 151 152 153 150 151
153 152 149 147 153 151 151 153 150 151
154 153 151 150 151 152 150 149 150 150
155 156 155 152 152 155 151 150 153 153
151 150 150 150 150 148 149 151 152 151
153 153 153 153 151 149 149 151 152 150
150 151 152 153 151 150 150 151 152 151
153 154 154 154 151 150 150 151 152 152
154 154 153 153 149 149 150 150 152 153
```

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

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

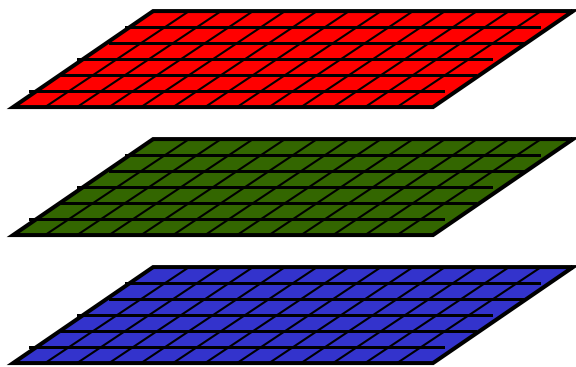
$$0 \leq A(i, j, 1) \leq 255$$

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

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

A color picture is made up of **RGB** matrices  $\rightarrow$  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!**



# Example: Mirror Image



Cornell University Law School  
Photograph by Cornell University Photography

`LawSchool.jpg`



Cornell University Law School  
Photograph by Cornell University Photography

`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.

## Reading and writing jpg files

```
% Read jpg image and convert to
```

```
% a 3D array A
```

```
A = imread('LawSchool1.jpg');
```

```
% Write 3D array B to memory as
```

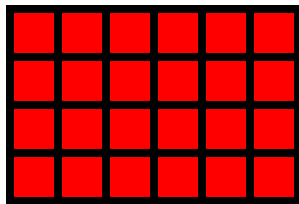
```
% a jpg image
```

```
imwrite(B, 'LawSchoolMirror.jpg')
```

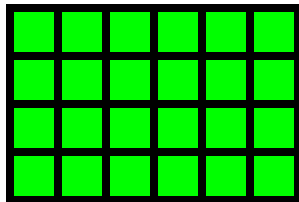
# A 3-d array as 3 matrices

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

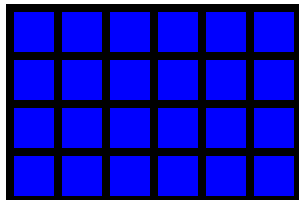
#rows                      #columns                      #layers (pages)



4-by-6



4-by-6



4-by-6

```
A(1:nr, 1:nc, 1)
```

```
M1 = A(:, :, 1)
```

```
M2 = A(:, :, 2)
```

```
M3 = A(:, :, 3)
```

**%Store mirror image of A in array B**

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

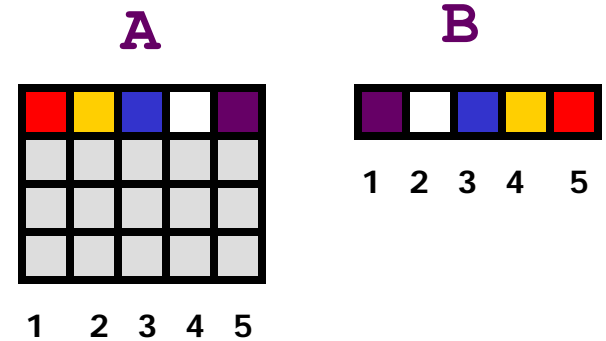
```
for r= 1:nr
```

```
    for c= 1:nc
```

```
        B(r,c )= A(r,nc-c+1 ) ;
```

```
    end
```

```
end
```



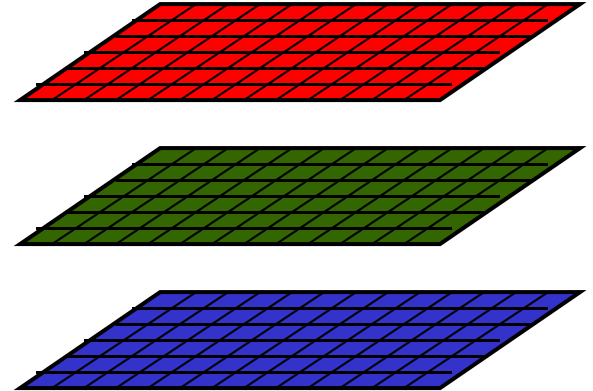
```
%Store mirror image of A in array B
```

```
[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
```

```

[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
end

```



```

[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
end

```

*Both fragments  
create a mirror  
image of A .*

A true

B false

```

[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

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

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

```
[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
```

```
imshow(B)    % Show 3-d array data as an image
```

```
imwrite(B,'LawSchoolMirror.jpg')
```



```

% 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')

```