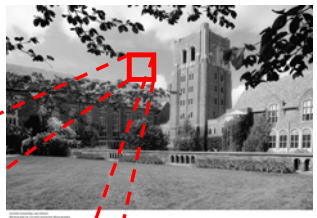
- Previous Lecture:
  - Image processing
    - Add frame, mirror
- Today's Lecture:
  - More image processing
    - Flipping an image
    - Color → grayscale
    - "Noise" filtering
    - (Watch online/read in book: Edge finding example)
- Announcements:
  - Discussion this week in the classrooms as listed on Student Center
  - Project 4 due Mon Oct 24th
  - Pick up your prelim paper during consulting hours

Lecture 16 2

Grayness: a value in [0..255]

0 = black
255 = white

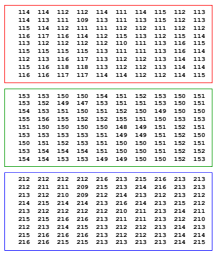
These are *integer* values
Type: `uint8`



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 16 4

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



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

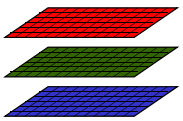
$0 \le A(i,j,1) \le 255$
$0 \le A(i,j,2) \le 255$
$0 \le A(i,j,3) \le 255$

Lecture 15 5

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

Color image

3-d Array



$0 \le A(i,j,1) \le 255$
$0 \le A(i,j,2) \le 255$
$0 \le A(i,j,3) \le 255$

Operations on images amount to operations on matrices!

Lecture 15 6

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 7

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')
```

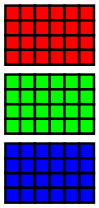
Lecture 15 8

A 3-d array as 3 matrices

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

#rows #columns #layers (pages)

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

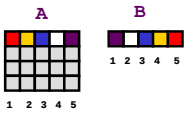


4-by-6 `M1= A(:,:,1)`

4-by-6 `M2= A(:,:,2)`

4-by-6 `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
```




```
[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 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` .

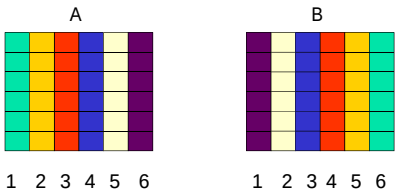
A true

B false


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

A= imread('LawSchool.jpg');
[nr,nc,np]= size(A);

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')
```

Vectorized code simplifies things…

Work with a whole column at a time



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

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."

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-c+1,1)
    B(:,c,2) = A(:,nc-c+1,2)
    B(:,c,3) = A(:,nc-c+1,3)
end
imwrite(B,'LawSchoolMirror.jpg')
```


Even more compact vectorized code to create a mirror image…

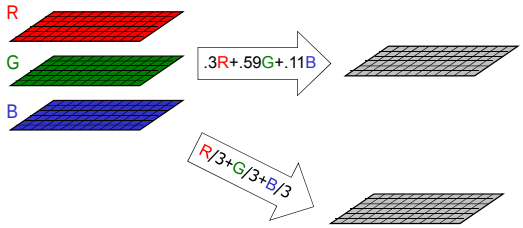
```
for c= 1:nc
    B(:,c,1) = A(:,nc-c+1,1)
    B(:,c,2) = A(:,nc-c+1,2)
    B(:,c,3) = A(:,nc-c+1,3)
end
```

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


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

Averaging the RGB values to get a gray value


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

vectorized operation

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
```

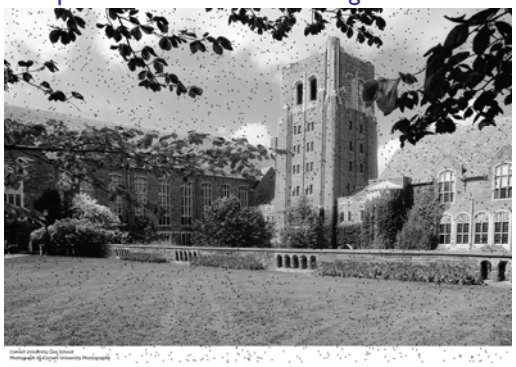
A: yes B: no

showToGrayscale.m

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

B = rgb2gray(A)

Clean up "noise" — median filtering



Dirt in the image!

Note how the "dirty pixels" look out of place

150	149	152	153	152	155
151	150	153	154	153	156
153	2	3	156	155	158
154	2	1	157	156	159
156	154	158	159	158	161
157	156	159	160	159	162

What to do with the dirty pixels?

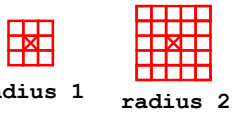
Assign "typical" neighborhood gray values to "dirty pixels"

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

What are "typical neighborhood gray values"?

Median

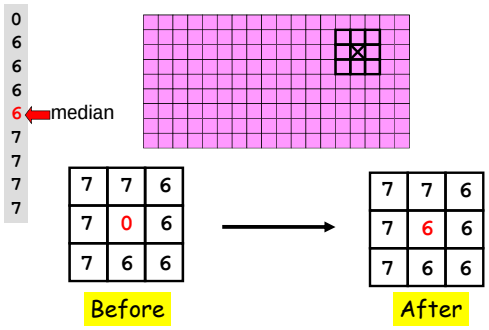
Mean



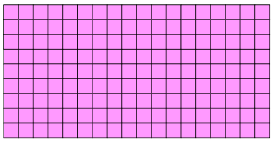
Median Filtering

- Visit each pixel
- Replace its gray value by the median of the gray values in the "neighborhood"

Using a radius 1 "neighborhood"



Visit every pixel; compute its new value.



```
m = 9
n = 18
```

```
for i=1:m
   for j=1:n
       Compute new gray value for pixel (i,j).
   end
end
```


What we need…

- (1) A function that computes the median value in a 2-dimensional array C:

```
m = medVal(C)
```

- (2) A function that builds the filtered image by using median values of radius r neighborhoods:

```
B = medFilter(A,r)
```


Computing the median

```
x : 21 89 36 28 19 88 43
```

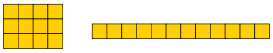
```
x = sort(x)
```

```
x : 19 21 28 36 43 88 89
```

```
n = length(x);   % n = 7
m = ceil(n/2);   % m = 4
med = x(m);      % med = 36
```

If n is even, then use : `med = x(m)/2 + x(m+1)/2`

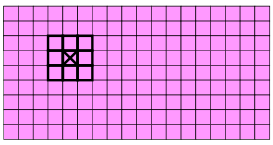
Median of a 2D array



```
function  med = medVal(C)
[nr,nc] = size(C);
x = zeros(1,nr*nc);
for r=1:nr
   x((r-1)*nc+1:r*nc) = C(r,:);
end
%Compute median of x and assign to med
% …
```

See `medVal.m`

Back to filtering…

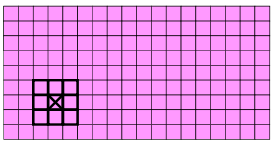


m = 9

n = 18

```
for i=1:m
   for j=1:n
       Compute new gray value for pixel (i,j)
   end
end
```


When window is inside…



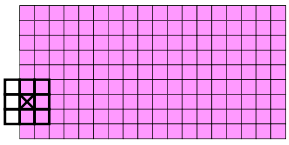
m = 9

n = 18

New gray value for pixel (7,4) =

```
medVal( A(6:8,3:5) )
```


When window is partly outside…



m = 9

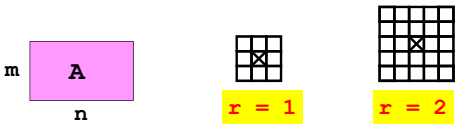
n = 18

New gray value for pixel (7,1) =

```
medVal( A(6:8,1:2) )
```


The Pixel (i,j) Neighborhood

```
iMin = max(1,i-r)
iMax = min(m,i+r)
jMin = max(1,j-r)
jMax = min(n,j+r)
C = A(iMin:iMax,jMin:jMax)
```




```
function B = medFilter(A,r)
% B from A via median filtering
% with radius r neighborhoods.

[m,n] = size(A);
B = uint8(zeros(m,n));
for i=1:m
   for j=1:n
       C = pixel (i,j) neighborhood
       B(i,j) = medVal(C);
   end
end
```


A

B = medianFilter(A,3)