Previous Lecture:
- Image processing
  - Add frame, mirror
Today’s Lecture:
- More image processing
  - Flipping an image
  - Color \rightarrow 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

Grayness: a value in [0..255]

0 = black
255 = white

These are integer values
Type: \texttt{uint8}

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

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

\[
\begin{array}{cccccc}
150 & 149 & 152 & 153 & 152 & 155 \\
151 & 150 & 153 & 154 & 153 & 156 \\
153 & 151 & 155 & 156 & 155 & 158 \\
154 & 153 & 156 & 157 & 156 & 159 \\
156 & 154 & 158 & 159 & 158 & 161 \\
157 & 156 & 159 & 160 & 159 & 162
\end{array}
\]

Operations on images amount to operations on matrices!

Example: Mirror Image

1. Read \texttt{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

```matlab
% 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')
```
A 3-d array as 3 matrices

\[ [nr, nc, np] = \text{size}(A) \]  
\# rows  
\# columns  
\# layers (pages)

\[
A(1:nr,1:nc,1) \]

\[
A(1:nr,1:nc,2) \]

\[
A(1:nr,1:nc,3) \]

% Store mirror image of A in array B

\[
[nr,nc,np] = \text{size}(A); \]

for \( r = 1:nr \)

for \( c = 1:nc \)

\[
B(r,c,p) = A(r,nc-c+1,p); \]

end

end

% Make mirror image of A -- the whole thing

\[
A = \text{imread}('LawSchool.jpg'); \]

\[
[nr,nc,np] = \text{size}(A); \]

\[
B = \text{zeros}(nr,nc,np); \]

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

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

\[
[nr,nc,np] = \text{size}(A); \]

for \( c = 1:nc \)

\[
B(:,c) = A(:,nc-c+1); \]

end

Consider a single matrix (just one layer)

\[
[nr,nc,np] = \text{size}(A); \]

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

```matlab
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...

```matlab
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 \(\rightarrow\) black and white

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

Averaging the RGB values to get a gray value

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

Averaging the RGB values to get a gray value

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

```matlab
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

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!

Assign “typical” neighborhood gray values to “dirty pixels”

What are “typical neighborhood gray values”?

Median

Mean

radius 1

radius 2

What to do with the dirty pixels?

Note how the “dirty pixels” look out of place.
Median Filtering

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

Using a radius 1 “neighborhood”

Before

After

What we need...

- (1) A function that computes the median value in a 2-dimensional array C:
  \[ m = \text{medVal}(C) \]

- (2) A function that builds the filtered image by using median values of radius r neighborhoods:
  \[ B = \text{medFilter}(A, r) \]

Computing the median

```plaintext
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

```plaintext
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…

\[
\begin{align*}
  m &= 9 \\
  n &= 18
\end{align*}
\]

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

When window is inside...

\[
\begin{align*}
  m &= 9 \\
  n &= 18
\end{align*}
\]

New gray value for pixel \((7,4)\) =

\[
\text{medVal}( A(6:8,3:5) )
\]

When window is partly outside...

\[
\begin{align*}
  m &= 9 \\
  n &= 18
\end{align*}
\]

New gray value for pixel \((7,1)\) =

\[
\text{medVal}( A(6:8,1:2) )
\]

The Pixel \((i,j)\) Neighborhood

\[
\begin{align*}
  i_{\text{Min}} &= \max(1,i-r) \\
  i_{\text{Max}} &= \min(m,i+r) \\
  j_{\text{Min}} &= \max(1,j-r) \\
  j_{\text{Max}} &= \min(n,j+r)
\end{align*}
\]

\[
C = A(i_{\text{Min}}:i_{\text{Max}},j_{\text{Min}}:j_{\text{Max}})
\]

\[
\begin{array}{ccc}
  m & A & \text{r = 1} \\
  n & & \text{r = 2}
\end{array}
\]

function \( B = \text{medFilter}(A,r) \)

% B from A via median filtering % with radius r neighborhoods.

\[
\begin{align*}
  [m,n] &= \text{size}(A); \\
  B &= \text{uint8} ( \text{zeros}(m,n) ); \\
  \text{for} \ i=1:m \\
    \text{for} \ j=1:n \\
      C &= \text{pixel}(i,j)\text{ neighborhood} \\
      B(i,j) &= \text{medVal}(C) ;
    \text{end}
  \text{end}
\end{align*}
\]

B = \text{medianFilter}(A,3)