Previous Lecture:
- Image processing
  - Add frame, mirror

Today’s Lecture:
- More image processing
  - Color to grayscale
  - “Noise” filtering
  - Edge finding

Announcements:
- Project 4 due Thursday after spring break
- There are no office/consulting hours during spring break; they resume on Monday after the break.
Grayness: a value in [0..255]

0 = black
255 = white

These are integer values
Type: uint8

<table>
<thead>
<tr>
<th></th>
<th>150</th>
<th>149</th>
<th>152</th>
<th>153</th>
<th>152</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>151</td>
<td>150</td>
<td>153</td>
<td>154</td>
<td>153</td>
<td>156</td>
</tr>
<tr>
<td>1</td>
<td>153</td>
<td>151</td>
<td>155</td>
<td>156</td>
<td>155</td>
<td>158</td>
</tr>
<tr>
<td>2</td>
<td>154</td>
<td>153</td>
<td>156</td>
<td>157</td>
<td>156</td>
<td>159</td>
</tr>
<tr>
<td>3</td>
<td>156</td>
<td>154</td>
<td>158</td>
<td>159</td>
<td>158</td>
<td>161</td>
</tr>
<tr>
<td>4</td>
<td>157</td>
<td>156</td>
<td>159</td>
<td>160</td>
<td>159</td>
<td>162</td>
</tr>
</tbody>
</table>
Vectorized code to create a mirror image

\[ A = \text{imread('LawSchool.jpg')} \]
\[ [nr, nc, np] = \text{size}(A); \]
\[ \text{for } c = 1:nc \]
\[ \quad B(:,c,1) = A(:,nc+1-c,1) \]
\[ \quad B(:,c,2) = A(:,nc+1-c,2) \]
\[ \quad B(:,c,3) = A(:,nc+1-c,3) \]
\[ \text{end} \]
\[ \text{imwrite}(B,'LawSchoolMirror.jpg') \]
Even more compact vectorized code to create a mirror image...

```matlab
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

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

\[ \frac{R}{3} + \frac{G}{3} + \frac{B}{3} \]

\[ 0.3R + 0.59G + 0.11B \]
Averaging the RGB values to get a gray value

\[
M(i,j) = 0.3R(i,j) + 0.59G(i,j) + 0.11B(i,j)
\]

for i = 1:m
    for j = 1:n
        \[
        M(i,j) = 0.3R(i,j) + 0.59G(i,j) + 0.11B(i,j)
        \]
    end
end
Averaging the RGB values to get a gray value

\[ M = 0.3R + 0.59G + 0.11B \]
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:

\[ B = \text{rgb2gray}(A) \]
Clean up “noise” — median filtering
Dirt in the image!

Note how the "dirty pixels" look out of place

<table>
<thead>
<tr>
<th>150</th>
<th>149</th>
<th>152</th>
<th>153</th>
<th>152</th>
<th>155</th>
</tr>
</thead>
<tbody>
<tr>
<td>151</td>
<td>150</td>
<td>153</td>
<td>154</td>
<td>153</td>
<td>156</td>
</tr>
<tr>
<td>153</td>
<td>2</td>
<td>3</td>
<td>156</td>
<td>155</td>
<td>158</td>
</tr>
<tr>
<td>154</td>
<td>2</td>
<td>1</td>
<td>157</td>
<td>156</td>
<td>159</td>
</tr>
<tr>
<td>156</td>
<td>154</td>
<td>158</td>
<td>159</td>
<td>158</td>
<td>161</td>
</tr>
<tr>
<td>157</td>
<td>156</td>
<td>159</td>
<td>160</td>
<td>159</td>
<td>162</td>
</tr>
</tbody>
</table>
What to do with the dirty pixels?

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

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>149</td>
<td>152</td>
<td>153</td>
<td>152</td>
<td>155</td>
</tr>
<tr>
<td>151</td>
<td>150</td>
<td>153</td>
<td>154</td>
<td>153</td>
<td>156</td>
</tr>
<tr>
<td>153</td>
<td>?</td>
<td>?</td>
<td>156</td>
<td>155</td>
<td>158</td>
</tr>
<tr>
<td>154</td>
<td>?</td>
<td>?</td>
<td>157</td>
<td>156</td>
<td>159</td>
</tr>
<tr>
<td>156</td>
<td>154</td>
<td>158</td>
<td>159</td>
<td>158</td>
<td>161</td>
</tr>
<tr>
<td>157</td>
<td>156</td>
<td>159</td>
<td>160</td>
<td>159</td>
<td>162</td>
</tr>
</tbody>
</table>
What are “typical neighborhood gray values”?

Median

Mean

radius 1

radius 2
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

\[
\begin{array}{ccc}
7 & 7 & 6 \\
7 & 0 & 6 \\
7 & 6 & 6 \\
\end{array}
\]

After

\[
\begin{array}{ccc}
7 & 7 & 6 \\
7 & 6 & 6 \\
7 & 6 & 6 \\
\end{array}
\]

median
Visit every pixel; compute its new value.

\[
\begin{array}{c}
\text{for } i=1:m \\
\quad \text{for } j=1:n \\
\quad \quad \text{Compute new gray value for pixel (i,j).} \\
\quad \text{end} \\
\text{end}
\end{array}
\]
Original:

\[ i = 1 \]
\[ j = 1 \]

Filtered:

Replace \( \ast \) with the median of the values under the window.
Original:

Filtered:

Replace $\times$ with the median of the values under the window.
Replace $\times$ with the median of the values under the window.
Replace $\times$ with the median of the values under the window.
Original:

\(i = 2\)

\(j = 1\)

Filtered:

Replace \(\times\) with the median of the values under the window.
Original:

\[ i = 2 \]
\[ j = 2 \]

Filtered:

Replace \( \times \) with the median of the values under the window.
Original:

\[ \begin{array}{c}
i = m \\
j = n
\end{array} \]

Filtered:

Replace \( \times \) with the median of the values under the window.
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

\[ x = \text{sort}(x) \]

\[ x : \begin{array}{c|c|c|c|c|c|c|}
21 & 89 & 36 & 28 & 19 & 88 & 43 \\
\end{array} \]

\[ x : \begin{array}{c|c|c|c|c|c|c|}
19 & 21 & 28 & 36 & 43 & 88 & 89 \\
\end{array} \]

\[ n = \text{length}(x); \quad % n = 7 \]
\[ m = \text{ceil}(n/2); \quad % m = 4 \]
\[ \text{med} = x(m); \quad % \text{med} = 36 \]

*If* \( n \) *is even, then use:* \[ \text{med} = x(m)/2 + x(m+1)/2 \]
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
for $i = 1:m$
  for $j = 1:n$
    Compute new gray value for pixel $(i,j)$
  end
end
When window is inside...

New gray value for pixel (7,4) =

\[ \text{medVal} \left( A(6:8,3:5) \right) \]
When window is partly outside...

New gray value for pixel (7,1) =

$$\text{medVal}( A(6:8,1:2) )$$
When window is partly outside…

New gray value for pixel (9,18) =

\[ \text{medVal}( A(8:9,17:18) ) \]
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
The Pixel (i,j) Neighborhood

\[
i_{\text{Min}} = i - r
\]
\[
i_{\text{Max}} = i + r
\]
\[
 j_{\text{Min}} = j - r
\]
\[
 j_{\text{Max}} = j + r
\]
\[
 C = A(i_{\text{Min}}:i_{\text{Max}},j_{\text{Min}}:j_{\text{Max}})
\]
The Pixel (i,j) Neighborhood

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

\[ C = A(i_{\text{Min}}:i_{\text{Max}}, j_{\text{Min}}:j_{\text{Max}}) \]
\[ B = \text{medianFilter}(A, 3) \]
Mean Filter with radius 3
Mean Filter with radius 10
Mean filter fails because the mean does not capture representative values.

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>149</td>
<td>152</td>
<td>153</td>
<td>152</td>
<td>155</td>
</tr>
<tr>
<td>151</td>
<td>150</td>
<td>153</td>
<td>154</td>
<td>153</td>
<td>156</td>
</tr>
<tr>
<td>153</td>
<td>2</td>
<td>3</td>
<td>156</td>
<td>155</td>
<td>158</td>
</tr>
<tr>
<td>154</td>
<td>2</td>
<td>1</td>
<td>157</td>
<td>156</td>
<td>159</td>
</tr>
<tr>
<td>156</td>
<td>154</td>
<td>158</td>
<td>159</td>
<td>158</td>
<td>161</td>
</tr>
<tr>
<td>157</td>
<td>156</td>
<td>159</td>
<td>160</td>
<td>159</td>
<td>162</td>
</tr>
</tbody>
</table>

| 85  | 86  |
| 87  | 88  |

mean-filtered values
Finding Edges
What is an edge?

Near an edge, grayness values change abruptly

200  200  200  200  200  200  200
200  200  200  200  200  200  100
200  200  200  200  100  100  100
200  200  100  100  100  100  100
200  100  100  100  100  100  100
General plan for showing the edges in an image

- Identify the “edge pixels”
- Highlight the edge pixels
  - make edge pixels white; make everything else black

```
200  200  200  200  200  200
200  200  200  200  200  100
200  200  200  200  100  100
200  200  200  100  100  100
200  200  100  100  100  100
200  100  100  100  100  100
```
General plan for showing the edges in an image

- Identify the “edge pixels”
- Highlight the edge pixels
  - make edge pixels white; make everything else black
Finding Edges
The Rate-of-Change-Array

Suppose \( A \) is an image array with integer values between 0 and 255. Let \( B(i,j) \) be the maximum difference between and its eight neighbors. So \( B(i,j) \) is the maximum value in

\[
A(\max(1,i-1):\min(m,i+1),\ldots,\max(1,j-1):\min(n,j+1)) - A(i,j)
\]
Rate-of-change example

<table>
<thead>
<tr>
<th></th>
<th>90</th>
<th>81</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>62</td>
<td>60</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>57</td>
<td>58</td>
<td></td>
</tr>
</tbody>
</table>

Rate-of-change at middle pixel is 30

Be careful! In “uint8 arithmetic”
57 - 60 is 0
Edge finding: Rate-of-change matrix

\[ B_{ij} = \max \left\{ |A_{pq} - A_{ij}| \mid \forall (p, q) \in \mathbb{N} \right\} \]

Let \( \mathbb{N} \) be the set of subscripts \((p, q)\) that is the neighborhood of \((i, j)\).

A: grayness values

B: rate-of-change values

\( A_{ij} \)

\( B_{ij} \)
Edge finding: Rate-of-change matrix

Let \( \mathbf{N} \) be the set of subscripts \((p,q)\) that is the neighborhood of \((i,j)\)

\[
B_{ij} = \max \left\{ \left| A_{pq} - A_{ij} \right| \mid \forall (p, q) \in \mathbf{N} \right\}
\]

High rate of change \(\Leftrightarrow\) Abrupt change in grayness \(\Leftrightarrow\) An edge

\[
E_{ij} = \begin{cases} 
white & \text{if } B_{ij} > \tau \\
black & \text{if } B_{ij} \leq \tau 
\end{cases}
\]
function Edges(jpgIn,jpgOut,tau)
% jpgOut is the "edge diagram" of image jpgIn.
% At each pixel, if rate-of-change > tau
% then the pixel is considered to be on an edge.

A = rgb2gray(imread(jpgIn));
[m,n] = size(A);
B = uint8(zeros(m,n));
for i = 1:m
    for j = 1:n
        B(i,j) = ?????
    end
end
Recipe for rate-of-change $B(i,j)$

% The 3-by-3 subarray that includes $A(i,j)$
% and its 8 neighbors (for an interior pixel)
Neighbors = A(i-1:i+1,j-1:j+1);

% Subtract $A(i,j)$ from each entry
Diff = abs(double(Neighbors) - double(A(i,j)));

% Compute largest value in each column
colMax = max(Diff);

% Compute the max of the column max’s
B(i,j) = max(colMax);
function Edges(jpgIn,jpgOut,tau)
% jpgOut is the "edge diagram" of image jpgIn.
% At each pixel, if rate-of-change > tau
% then the pixel is considered to be on an edge.

A = rgb2gray(imread(jpgIn));
[m,n] = size(A);
B = uint8(zeros(m,n));
for i = 1:m
    for j = 1:n
        Neighbors = A(max(1,i-1):min(i+1,m), ...
                        max(1,j-1):min(j+1,n));
        B(i,j)=max(max(abs(double(Neighbors)- ...
                        double(A(i,j)))));
    end
end
end
“Edge pixels” are now identified; display them with maximum brightness (255)

\[
\begin{align*}
A &= \\
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 90 & 90 & 90 \\
1 & 1 & 90 & 90 & 90 & 90 & 90 \\
1 & 90 & 90 & 90 & 90 & 90 & 90
\end{bmatrix}
\end{align*}
\]

Threshold

\[
\text{if } B(i,j) > \text{tau} \quad B(i,j) = 255; \quad \text{end}
\]

\[
\begin{align*}
B(i,j) &= \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 89 & 89 & 89 & 89 \\
0 & 0 & 89 & 89 & 89 & 0 & 0 \\
0 & 89 & 89 & 0 & 0 & 0 & 0 \\
0 & 89 & 0 & 0 & 0 & 0 & 0 \\
0 & 89 & 0 & 0 & 0 & 0 & 0 \\
0 & 89 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{align*}
\]
function Edges(jpgIn,jpgOut,tau)
% jpgOut is the “edge diagram” of image jpgIn.
% At each pixel, if rate-of-change > tau
% then the pixel is considered to be on an edge.
A = rgb2gray(imread(jpgIn));
[m,n] = size(A);
B = uint8(zeros(m,n));
for i = 1:m
    for j = 1:n
        Neighbors = A(max(1,i-1):min(i+1,m), ...
                        max(1,j-1):min(j+1,n));
        B(i,j)=max(max(abs(double(Neighbors)- ...
                        double(A(i,j))));
        if B(i,j) > tau
            B(i,j) = 255;
        end
    end
end
end
function Edges(jpgIn,jpgOut,tau)
% jpgOut is the “edge diagram” of image jpgIn.
% At each pixel, if rate-of-change > tau
% then the pixel is considered to be on an edge.
A = rgb2gray(imread(jpgIn));
[m,n] = size(A);
B = uint8(zeros(m,n));
for i = 1:m
    for j = 1:n
        Neighbors = A(max(1,i-1):min(i+1,m), ...
                       max(1,j-1):min(j+1,n));
        B(i,j)=max(max(abs(double(Neighbors)- ...
                       double(A(i,j))));
        if B(i,j) > tau
            B(i,j) = 255;
        end
    end
end
imwrite(B,jpgOut,'jpg')
tau = 30
Edge finding: Effect of edge threshold, $\tau$