


- Previous Lecture:
 - Image processing
 - Add frame
 - Mirror
 - Color to grayscale
- Today's Lecture:
 - More image processing
 - "Noise" filtering
 - Edge finding




Lecture 16 3

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



114	114	112	112	114	111	114	111	112	113
114	113	111	109	111	111	113	111	112	112
115	114	112	111	111	112	112	111	111	112
116	117	114	114	112	111	111	111	110	114
119	112	112	112	112	110	111	111	111	115
119	119	115	115	111	111	111	111	116	114
119	113	114	117	113	112	112	113	114	113
119	116	118	118	112	112	112	113	114	114
116	117	117	114	114	112	112	114	114	115

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

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

E.g., color image data is stored in a 3-d `uint8` array **A**:

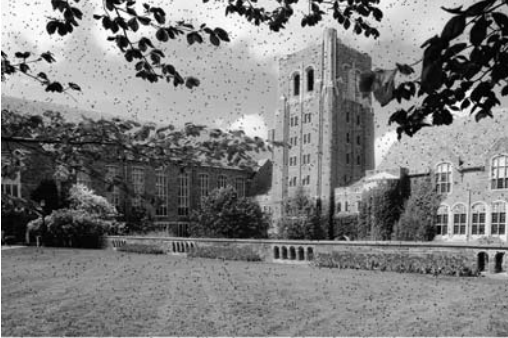
$$0 \leq A(i, j, 1) \leq 255$$

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

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

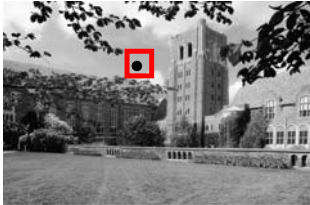
Lecture 15 5

Clean up "noise" — median filtering



Lecture 16 16

Dirt in the image!

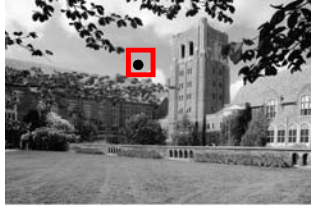


Note how the "dirty pixels" look out of place

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

Lecture 16 17

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

Lecture 16 18

What are “typical neighborhood gray values”?

Median
Mean

radius 1 radius 2

Lecture 16 19

Median Filtering

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

Lecture 16 20

Using a radius 1 “neighborhood”

0
6
6
6
6 ← median
7
7
7
7

7	7	6
7	0	6
7	6	6

Before

7	7	6
7	6	6
7	6	6

After

Lecture 16 21

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
    
```

Lecture 16 22

Original:

$i = 1$
 $j = 1$

Filtered:

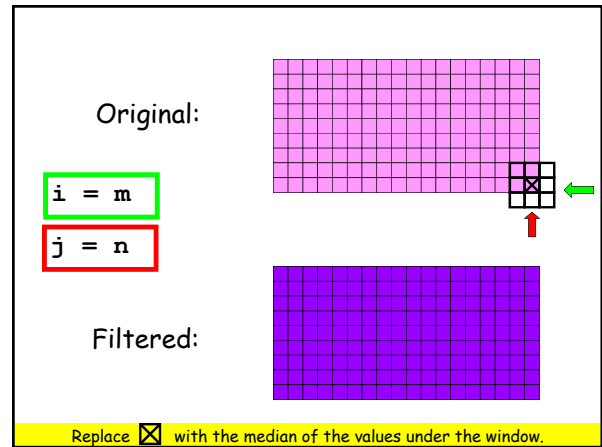
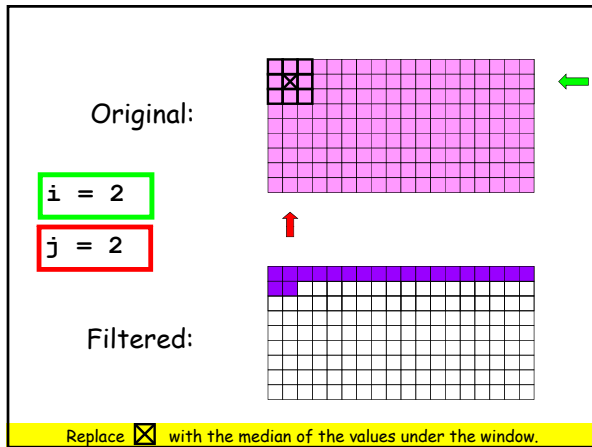
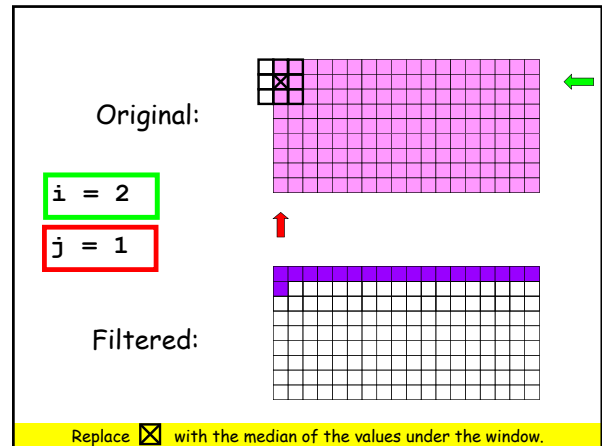
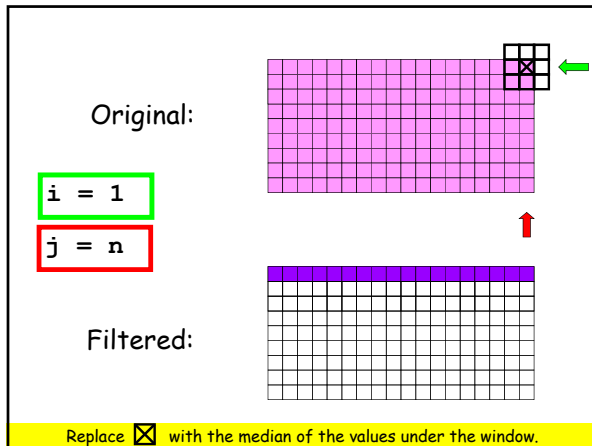
Replace \otimes with the median of the values under the window.

Original:

$i = 1$
 $j = 2$

Filtered:

Replace \otimes 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 = medVal(C)
```
- (2) A function that builds the filtered image by using median values of radius r neighborhoods:


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

Lecture 16 30

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
```

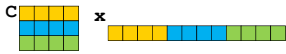
```
m = ceil(n/2); % m = 4
```

```
med = x(m); % med = 36
```

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

Lecture 16 31

Median of a 2D array



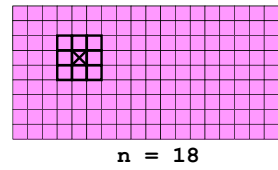
```

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

Lecture 16 32

Back to filtering...

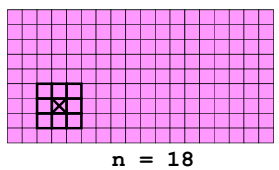


```

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

Lecture 16 33

When window is inside...



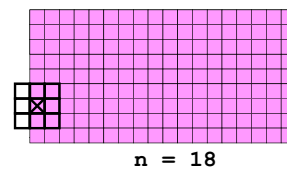
New gray value for pixel (7,4) =

```

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

Lecture 16 34

When window is partly outside...



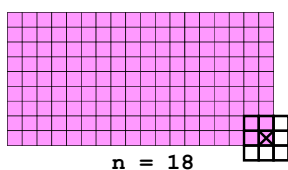
New gray value for pixel (7,1) =

```

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

Lecture 16 35

When window is partly outside...



New gray value for pixel (9,18) =

```

medVal( A(8:9,17:18) )
    
```

Lecture 16 36

```

function B = MedianFilter(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
    
```

Lecture 16 37

The Radius r Neighborhood of Pixel (i,j)

```

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)
    
```

See MedianFilter.m

A

r = 1

r = 2

Lecture 16 39

A

B = MedianFilter(A, 3)

Lecture 16 40

Mean Filter with radius 3

Lecture 16 41

Mean Filter with radius 10

Lecture 16 42

Mean filter fails because the mean does not capture representative values.

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

85	86
87	88

mean-filtered values

Lecture 16 43


Finding Edges

Lecture 16 47

What is an edge?

Near an edge, grayness values change abruptly

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




Lecture 16 48

General plan for showing the edges in in 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





Lecture 16 49

General plan for showing the edges in in 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

BLACK WHITE BLACK

Lecture 16 50

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 itself and its eight neighbors.

So $B(i, j)$ is the maximum value in

$$\max_{\substack{A(\max(1, i-1) : \min(m, i+1), \\ \max(1, j-1) : \min(n, j+1))}} - A(i, j)$$

Neighborhood of $A(i, j)$

Lecture 16 56

Rate-of-change example

90	81	65
62	60	59
56	57	58

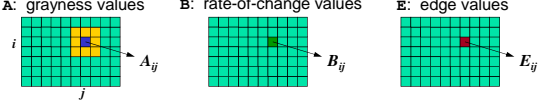
Rate-of-change at middle pixel is 30

Be careful! In "uint8 arithmetic" 57 - 60 is 0

Lecture 16 57

Edge finding: Rate-of-change matrix

A: grayness values B: rate-of-change values E: edge values



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

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

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

$$E_{ij} = \begin{cases} \text{white} & \text{if } B_{ij} > \tau \\ \text{black} & \text{if } B_{ij} \leq \tau \end{cases}$$

Lecture 16

```
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)); % Built-in function to
[m,n] = size(A); % convert to grayscale.
B = uint8(zeros(m,n)); % Returns 2-d array.
for i = 1:m
    for j = 1:n

        B(i,j) = ?????

    end
end
```

Lecture 16 61

Recipe for rate-of-change $B(i, j)$

```
% The subarray that includes A(i,j) and its
% neighbors
Neighbors = A(max(1,i-1):min(m,i+1), ...
              max(1,j-1):min(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);
```

Lecture 16 62

```
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
end

"Edge pixels" are now identified; display them with maximum brightness (255)

A					
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1	1	90	90
1	1	1	90	90	90
1	1	90	90	90	90
1	1	90	90	90	90

threshold
if $B(i,j) > \tau$
 $B(i,j) = 255;$
end

B					
0	0	0	0	0	0
0	0	0	89	89	89
0	0	89	89	0	0
0	89	89	0	0	0
0	89	0	0	0	0
0	89	0	0	0	0

0	0	0	0	0	0
0	0	0	255	255	255
0	0	255	255	0	0
0	255	255	0	0	0
0	255	0	0	0	0
0	255	0	0	0	0

Lecture 16 65

```
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')
```

