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

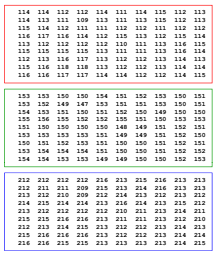
Grayness: a value in [0..255]

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

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



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

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

Clean up "noise" — median filtering

Dirt in the image!

Note how the "dirty pixels" look out of place

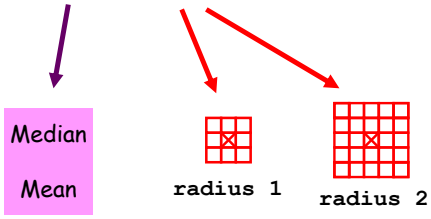
150	149	152	153	152	155
151	150	153	154	153	156
153	2	3	156	155	158
154	2	1	157	156	159
156	154	158	159	158	161
157	156	159	160	159	162

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

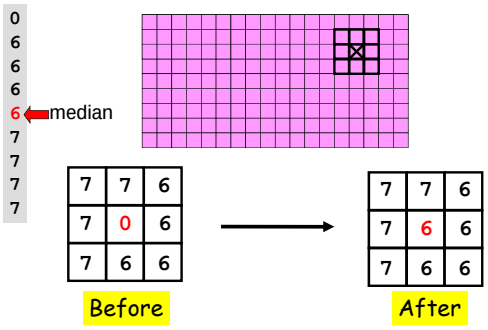
What are "typical neighborhood gray values"?



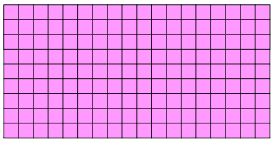
Median Filtering

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

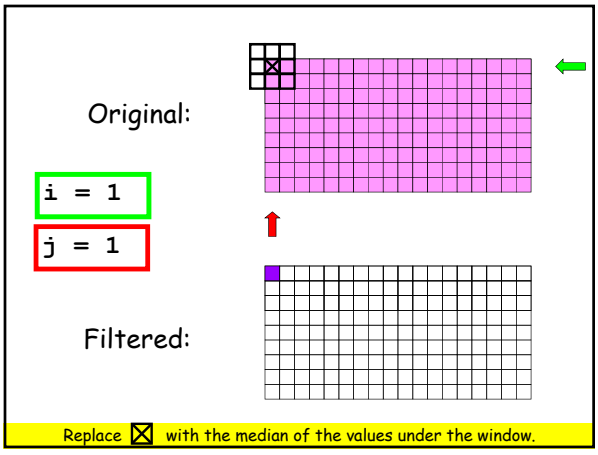
Using a radius 1 "neighborhood"

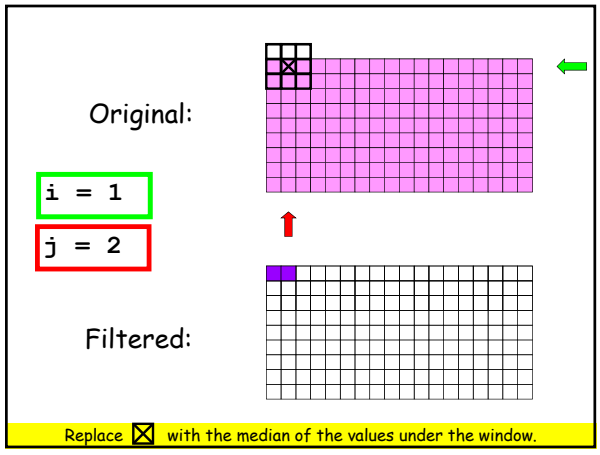


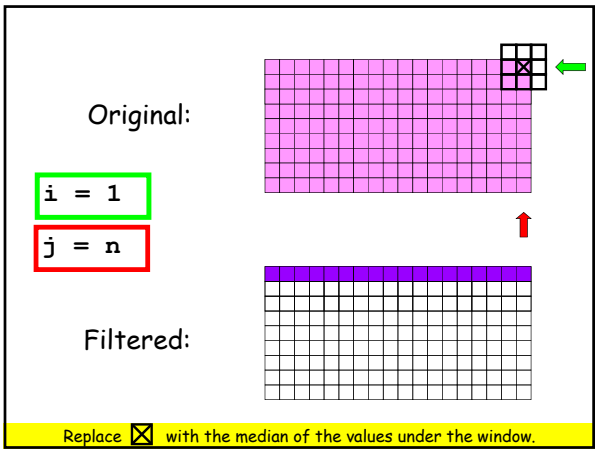
Visit every pixel; compute its new value.

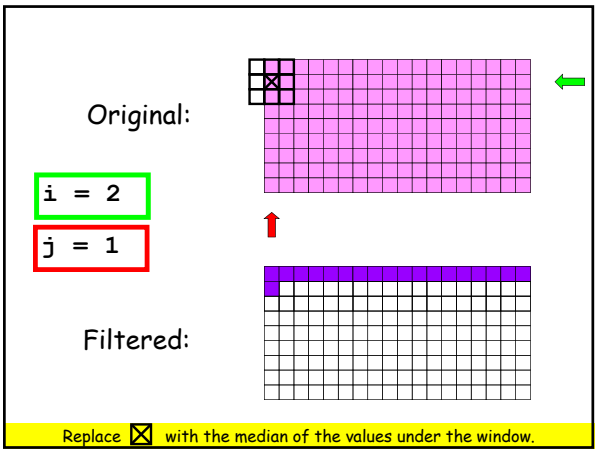


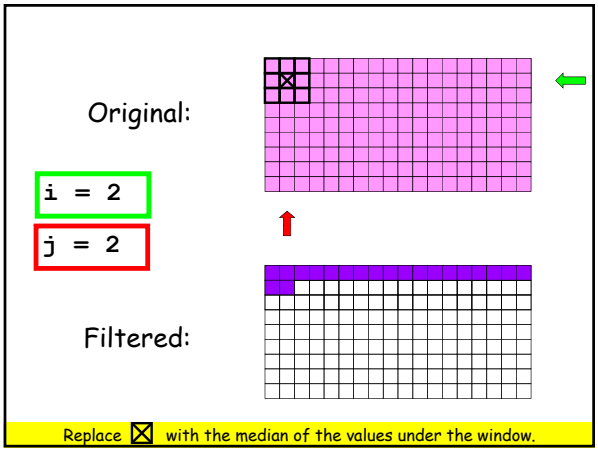

```
for i=1:m
   for j=1:n
        Compute new gray value for pixel (i,j).
   end
end
```

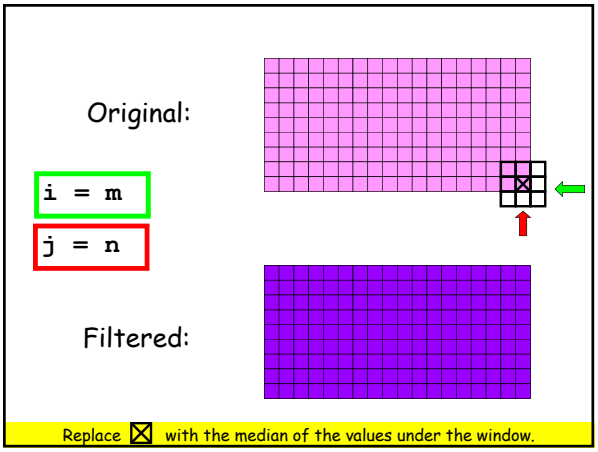













What we need…

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


Computing the median

x :	21	89	36	28	19	88	43

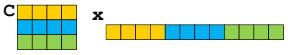
```
x = sort(x)
```

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`

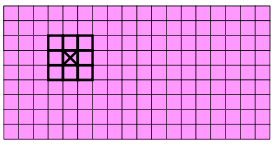
Median of a 2D array




```
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…

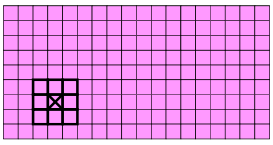


m = 9

n = 18

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


When window is inside…



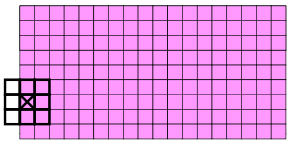
m = 9

n = 18

New gray value for pixel (7,4) =

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

When window is partly outside…



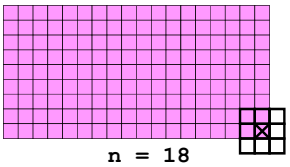
m = 9

n = 18

New gray value for pixel (7,1) =

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

When window is partly outside…



m = 9

n = 18

New gray value for pixel (9,18) =

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

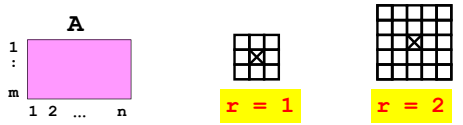

```
function B = MedianFilter(A,r)
% B from A via median filtering
% with radius r neighborhoods.

[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
```

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



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.

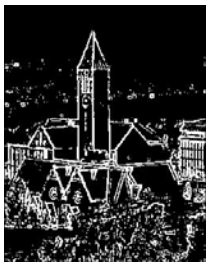
150	149	152	153	152	155
151	150	153	154	153	156
153	2	3	156	155	158
154	2	1	157	156	159
156	154	158	159	158	161
157	156	159	160	159	162

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

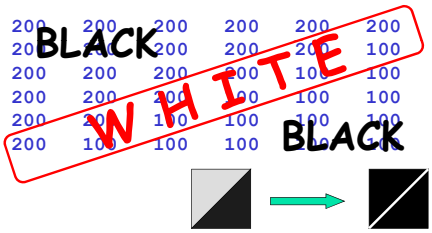
General plan for showing the edges in in image

- 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

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



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

```
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)

Rate-of-change example

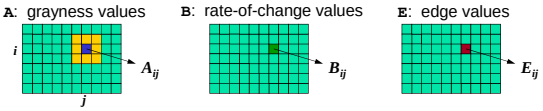
90	81	65
62	60	59
56	57	58

Rate-of-change at middle pixel is 30

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

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)

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

High rate of change ⇔ Abrupt change in grayness ⇔ An edge

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

```
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
```

Built-in function to convert to grayscale. Returns 2-d array.

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);
```



```
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
```

"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

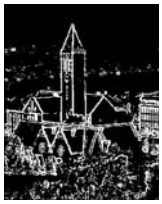
```
if B(i,j) > tau
    B(i,j) = 255;
end
```

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


```
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, τ

τ ⟶