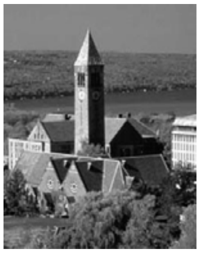
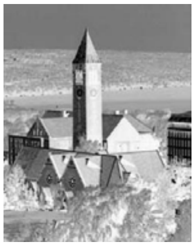
Recall

- An image in Matlab is just an array
  - A 2D array of uint8 values for a gray-scale image
  - A 3D array consisting of 3 layers (red, green, blue) for a color image
    - Each layer is a 2D array of uint8 values
- Images in a file are usually compressed
  - Matlab uses imread and imwrite
- Matlab uses imshow or image to display an image

Problem: Produce a Negative





uint8 values

- uint8
  = unsigned 8-bit integer
  - 2^8 = 256
    - Values are between 0 and 255 (inclusive)
- Arithmetic with uint8 produces uint8 results
  - Results that are too big are replaced with 255
  - Results that are negative are replaced with 0

- The Matlab *Workspace* shows the type for each of your variables
  - imread creates an array of type uint8
  - imwrite converts numbers to uint8 before writing

Idea for Cleaning a Dirty Image

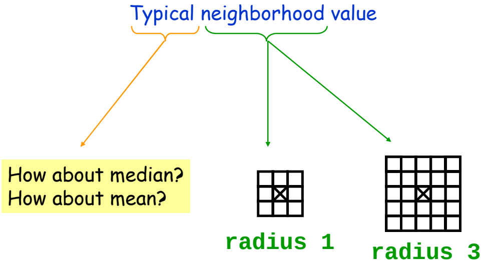
1458-by-2084

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

Assign "typical" neighborhood value to each dirty pixels

Getting Precise

Typical neighborhood value



How about median?
How about mean?

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 using median values of radius r neighborhoods:

   B = medFilter(A,r)

Median of a 2D Array

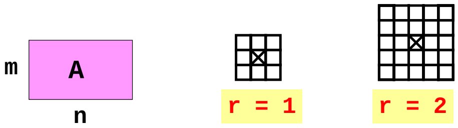
```
function med = medVal(C)
% Return the median value in the 2D array C.

% Assemble C's entries into a 1-dim array and sort
[p,q] = size(C);
n = p*q;
v = C(1:n);     % Can access 2D-array with 1D subscripts
v = sort(v);

% Compute median of v and assign to med
```

Filtering by Median

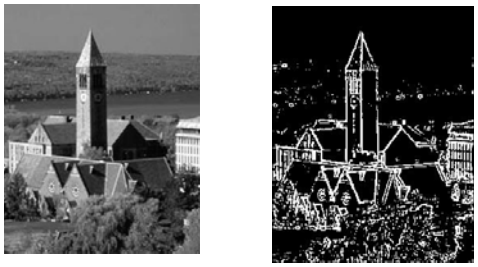
```
function B = MedianFilter(A,r)
% B is a uint8 array obtained from A by median filtering
% with radius r neighborhoods.
[m,n] = size(A);
B = uint8(zeros(m,n));
for i=1:m
   for j=1:n
      C = pixel (i,j) neighborhood
      B(i,j) = medVal(C);
   end
end
```

The Pixel (i,j) Neighborhood

```
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)
```



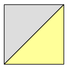
Finding Edges



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
```



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 A(i,j) and any of its eight neighbors