Finding Red Pixels – Part 1

Prof. Noah Snavely
CS1114
http://cs1114.cs.cornell.edu

Cornell University
Computer Science

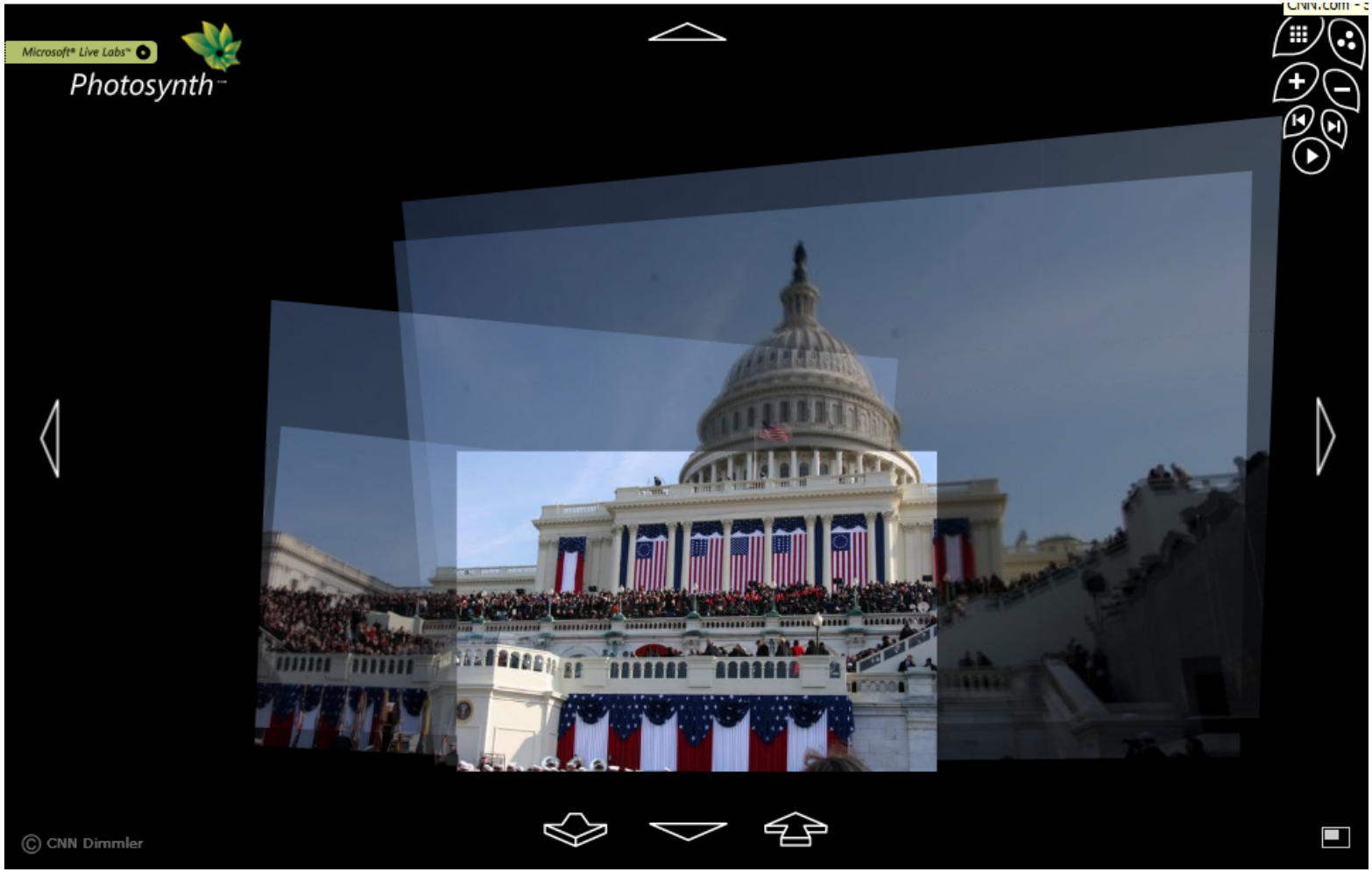
Administrivia

- Activate your CSUG accounts
- Assignment 1 will be out tomorrow, due Friday, Feb. 6
  - Will be graded in demo sessions
- Quiz 1 will be next Thursday
- Evening guest lecture on Tuesday
  - Robert Kleinberg on graphs

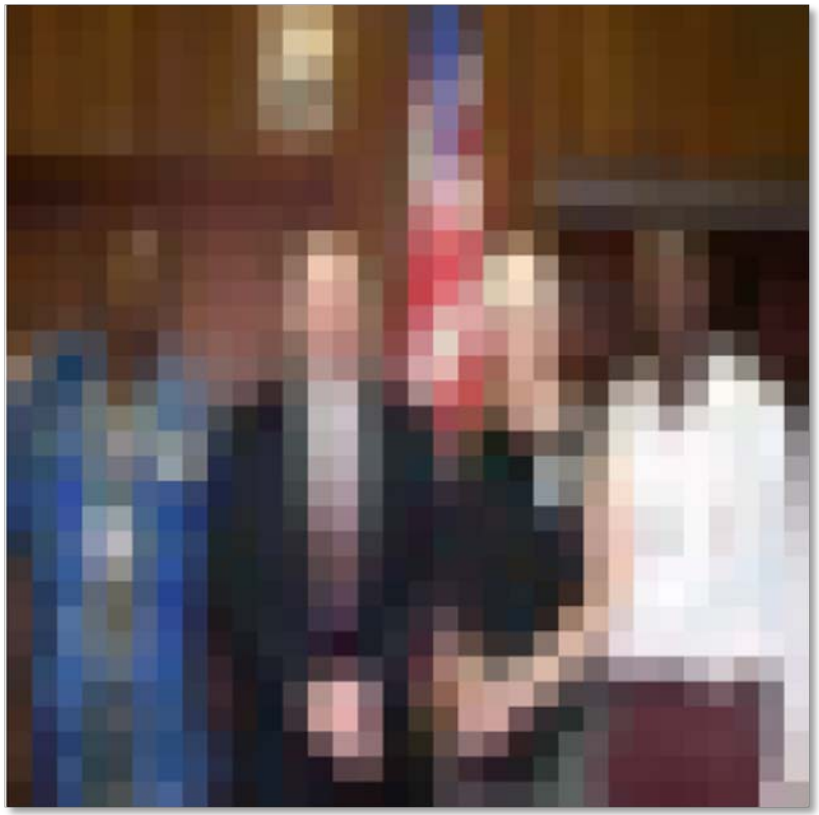
Tracking a lightstick

- We will spend the first part of CS1114 trying to track a red lightstick
- On the way we will cover important CS themes
  - Fundamental algorithms
  - Good programming style
  - Computational problem solving

What can we do with computer vision?



Human vision



Source: "80 million tiny images" by Torralba, et al.

Question: How many people are in this image?

Interpreting images

Q: Can a computer (or robot) understand this image?

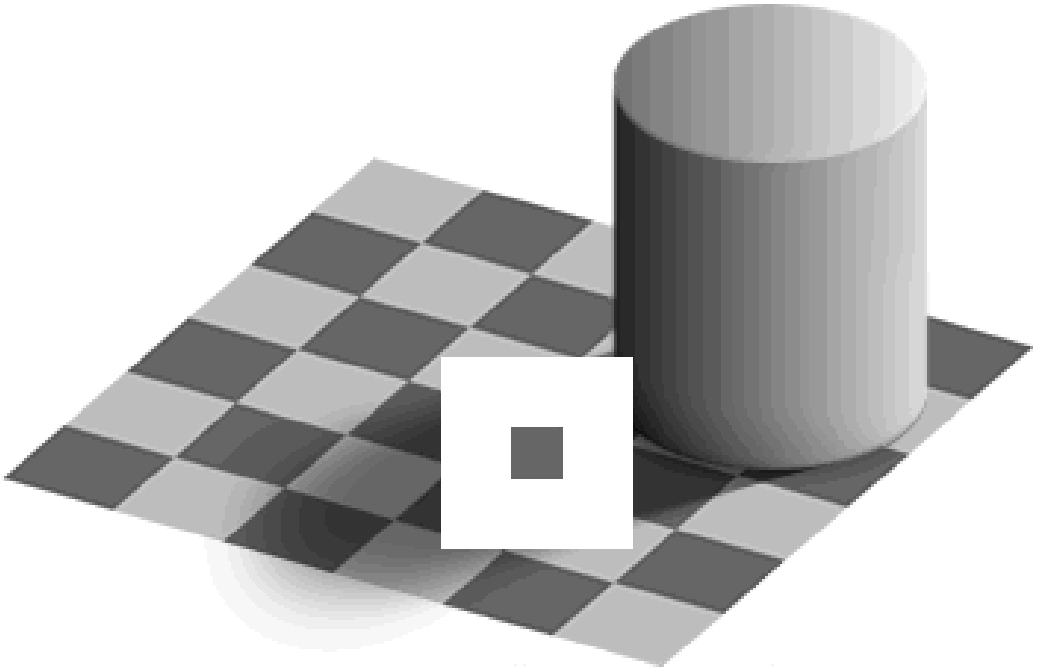
A: Yes and no (mostly no)

Human vision has its shortcomings…

Sinha and Poggio, *Nature*, 1996

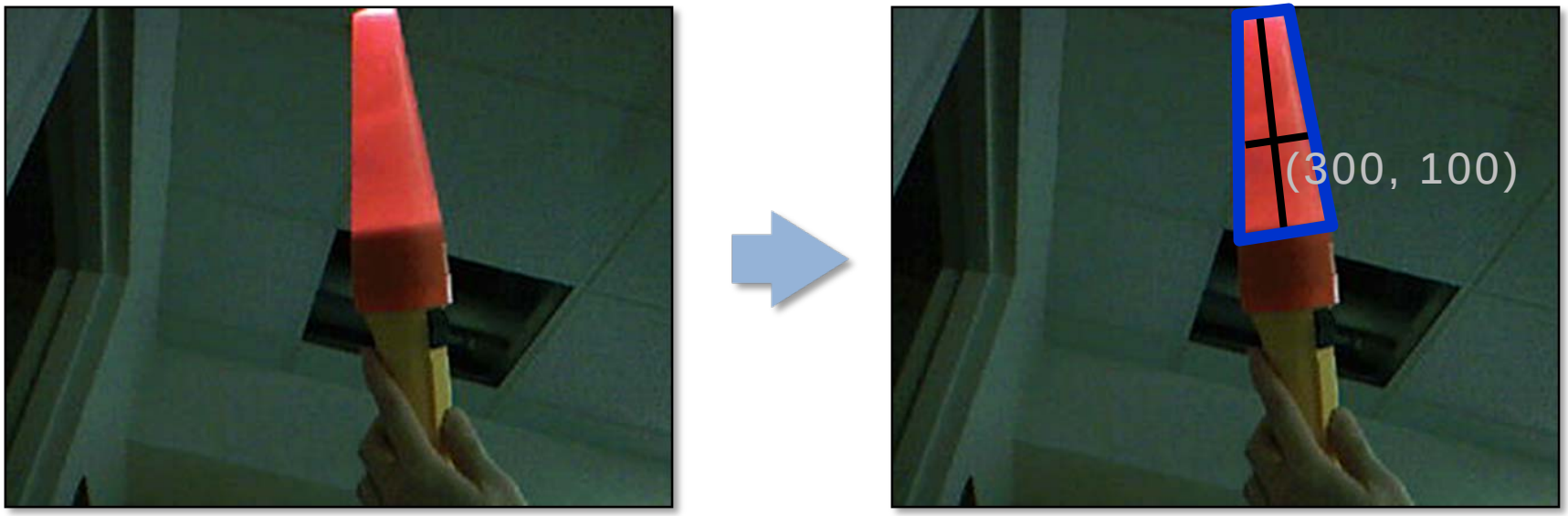
Credit: Steve Seitz

Human vision has its shortcomings…



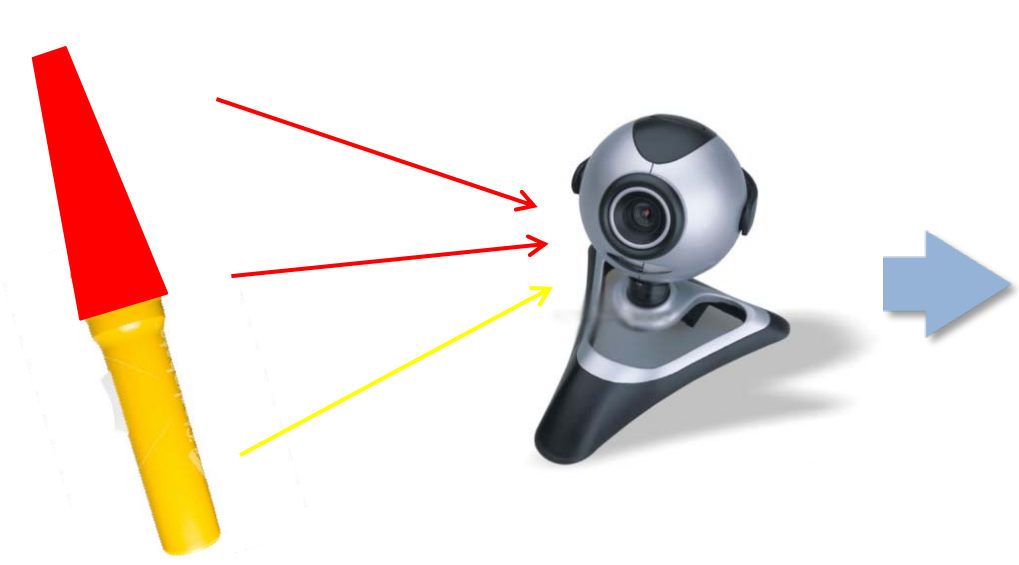
by Ted Adelson, slide credit Steve Seitz

Interpreting images



Q: Can a computer (or robot) find the lightstick?
A: With your help, yes!*

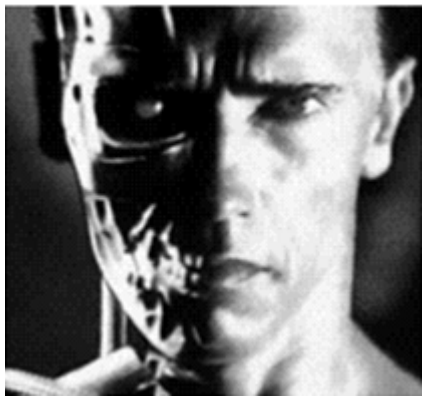
What is an image?



"lightstick1.jpg"

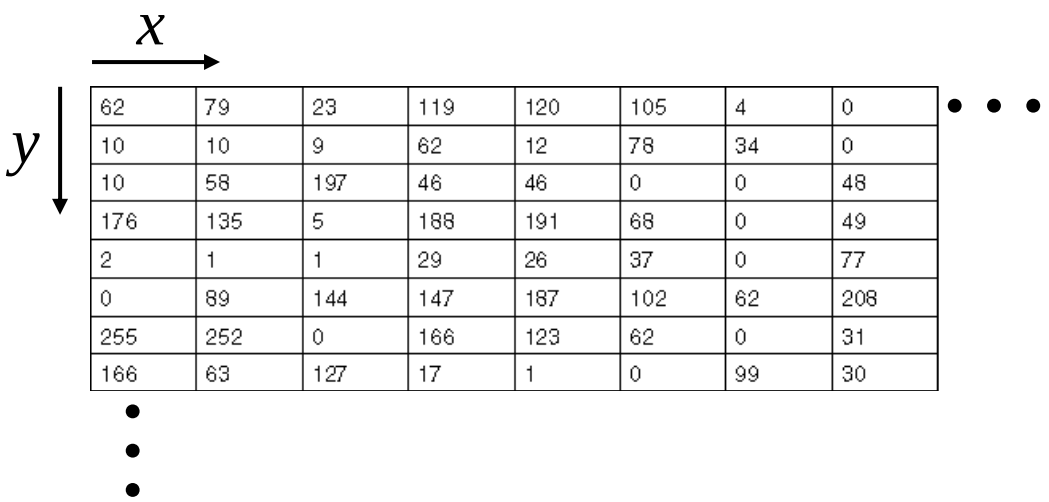
What is an image?

- A grid of numbers (*intensity values*)



snoop

3D view

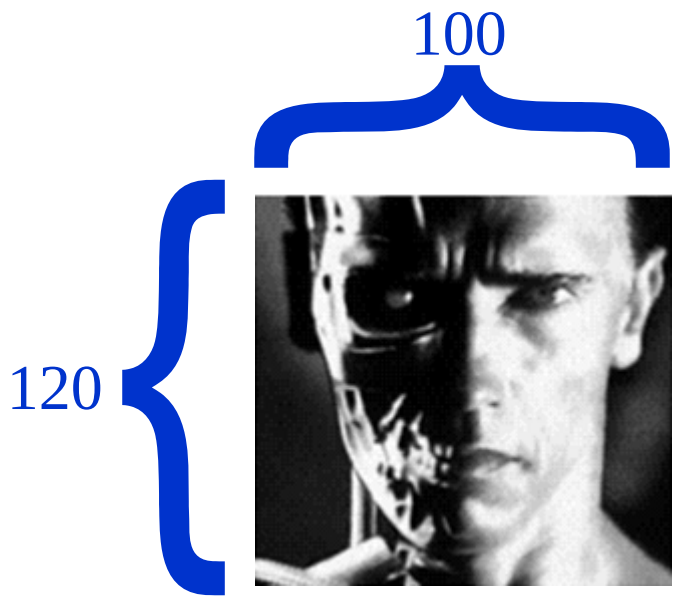


62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

- Intensity values range between 0 (black) and 255 (white)

What is an image?

- A grid of numbers (*intensity values*)
- In Matlab, a *matrix*




```
[ 10 30 40 106 123 … ;
   8 49 58 112 145 … ;
  16 53 86 123 152 … ;
            …          ]
```

120 x 100 matrix

Matrices in Matlab

- 1D matrix is often called a vector
  - Similar to arrays in other languages

```
A = [ 10 30 40 106 123 ]
```

Row vector
(or 1 x 5 matrix)

```
A(1) ==  10
A(4) == 106
```

```
B = [  10 ;
       30 ;
       40 ;
      106 ;
      123 ]
```

Column vector
(or 5 x 1 matrix)

Matrices in Matlab

```
C = [ 10 30 40 106 123 ;
       8 49 58 112 145 ;
      16 53 86 123 152  ]
```

3 x 5 matrix

```
C(1,1) == 10
```

```
C(2,4) == 112
```

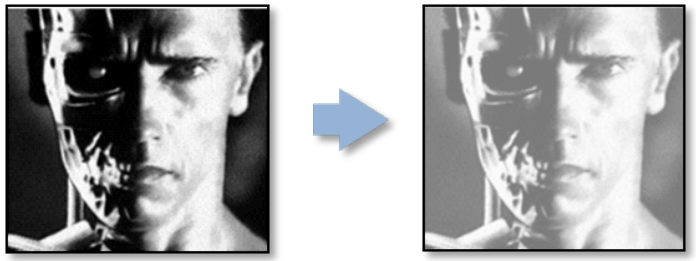
can also *assign* to a matrix entries

```
C(1,1) = C(1,1) + 1
```

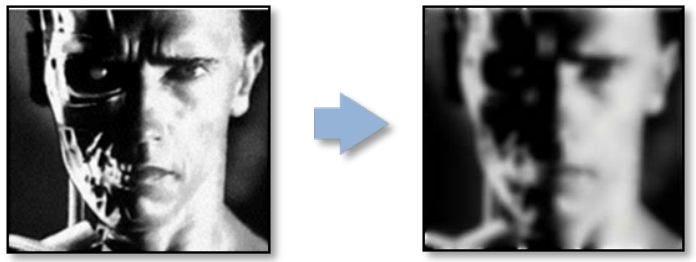
Image processing

- We often want to modify an image by "doing something" to each pixel:

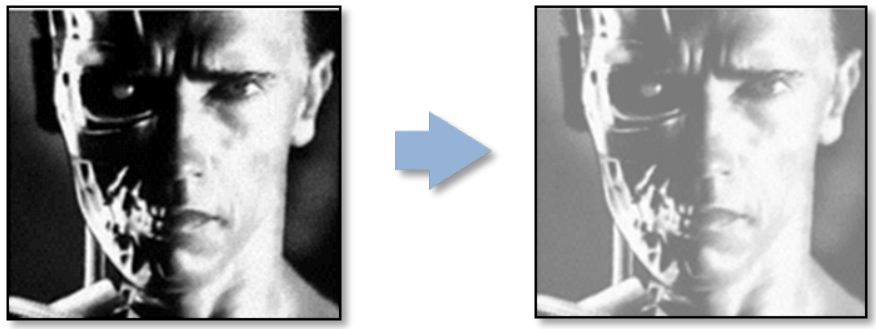
Brighten



Blur



Brightening an image



Q: What does this mean in terms of matrix entries?

```
[ 10 30 40 106 123 ;
   8 49 58 112 145 ;
  16 53 86 123 152  ]
```

A: Increase each element by some amount (say, 20)

Brightening an image (Take 1)

```
D = [ 10 30 40 106 123 8 49 58 112 145 16 53 ]
```

```
D(1)  = D(1) + 20;
D(2)  = D(2) + 20;
D(3)  = D(3) + 20;
D(4)  = D(4) + 20;
D(5)  = D(5) + 20;
D(6)  = D(6) + 20;
D(7)  = D(7) + 20;
D(8)  = D(8) + 20;
D(9)  = D(8) + 20;
D(10) = D(10) + 20;
D(11) = D(11) + 20;
D(12) = D(12) + 20;
D(13) = D(13) + 20;
D(14) = D(14) + 20;
D(15) = D(15) + 20;
```

Avoiding duplicate code

- Programming languages are designed to make this easy
  - It's a huge theme in language design
  - Many new programming techniques are justified by this
    - Object-oriented programming, higher-order procedures, functional programming, etc.

Why is it a bad idea to duplicate code?

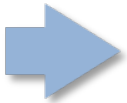
```
D(1)  = D(1) + 20;
D(2)  = D(2) + 20;
D(3)  = D(3) + 20;
D(4)  = D(4) + 20;
D(5)  = D(5) + 20;
D(6)  = D(6) + 20;
D(7)  = D(7) + 20;
D(8)  = D(8) + 20;
D(9)  = D(8) + 20;
D(10) = D(10) + 20;
D(11) = D(11) + 20;
D(12) = D(12) + 20;
```

- Hard to write
- Hard to modify
- Hard to get right
- Hard to generalize
- Programmer's "intent" is obscured

Brightening an image (Take 2)

- Using *iteration*

```
D(1)  = D(1) + 20;
D(2)  = D(2) + 20;
D(3)  = D(3) + 20;
D(4)  = D(4) + 20;
D(5)  = D(5) + 20;
D(6)  = D(6) + 20;
D(7)  = D(7) + 20;
D(8)  = D(8) + 20;
D(9)  = D(9) + 20;
D(10) = D(10) + 20;
D(11) = D(11) + 20;
D(12) = D(12) + 20;
```



```
for i = 1:12
    D(i) = D(i) + 20;
end
```

- Much easier to understand and modify the code
- Better expresses programmer's "intent"

Many advantages to iteration

- Can do things with iteration that you can't do by just writing lots of statements
- Example: increment every vector cell
  - Without knowing the length of the vector!

```
len = length(D); % New Matlab function
for i = 1:len
    D(i) = D(i) + 20;
end
```

Introducing iteration into code

- Programming often involves “clichés”
  - Patterns of code rewriting
  - I will loosely call these “design patterns”
- Iteration is our first example

Brightening 2D images

```
C = [ 10 30 40 106 123 ;
       8 49 58 112 145 ;
      16 53 86 123 152  ]
```

3 x 5 matrix

```
for row = 1:3
    for col = 1:5
        C(row,col) = C(row,col) + 20;
    end
end
```

Called a “nested” for loop

Brightening 2D images

```
for row = 1:3
    for col = 1:5
        C(row,col) = C(row,col) + 20
    end
end
```

- What if it's not a 3x5 matrix?

```
[nrows,ncols] = size(C) % New Matlab function
for row = 1:nrows
    for col = 1:ncols
        C(row,col) = C(row,col) + 20
    end
end
```

Using iteration to count

```
nzeros = 0;
[nrows,ncols] = size(D);
for row = 1:nrows
    for col = 1:ncols
        if D(row,col) == 0
            nzeros = nzeros + 1;
        end;
    end;
end;
```

```
D = [ 10 30  0 106 123 ;
       8 49 58   0 145 ;
      16  0 86 123 152  ]
```

Using iteration to count

```
nzeros = 0;
[nrows,ncols] = size(D);
for row = 1:nrows
    for col = 1:ncols
        if D(row,col) == 0
            nzeros = nzeros + 1;
        end;
    end;
end;
```

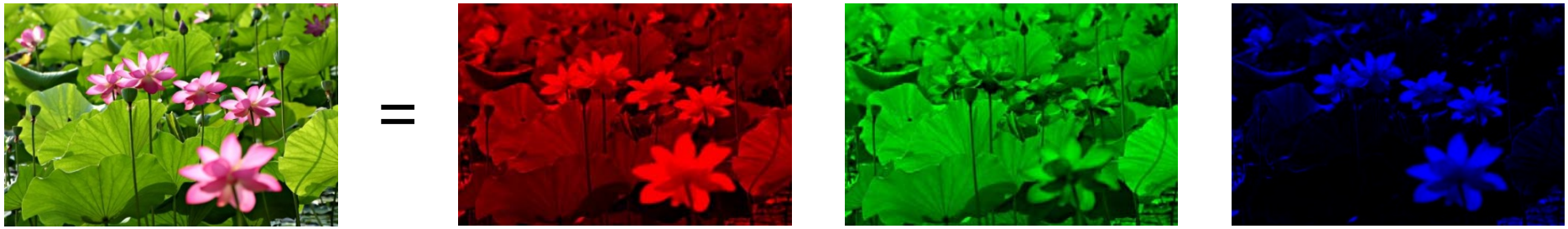
If D is an image, what are we counting?

What about red pixels?

- A grayscale image is a 2D array
  - Brightest = 255, darkest = 0

What about red pixels?

- A color image is 3 different 2D arrays
  - For red/green/blue values (RGB)
  - We provide a way to create these 3 arrays
    - ♦ Why are there 3?



What about red pixels?

- Example colors:

red(1,1) == 255, green(1,1) == blue(1,1) == 0

red(2,1) == 100 == green(2,1) == blue(2,1)

red(3,1) == 0 == green(3,1) == blue(3,1)

red(3,1) == 255 == green(3,1), blue(3,1) == 0

For next time

- Visit the lab, try out the rest of the Matlab tutorial
- Make sure your CSUG account is activated