CS 1114: Introduction to Computing Using MATLAB and Robotics

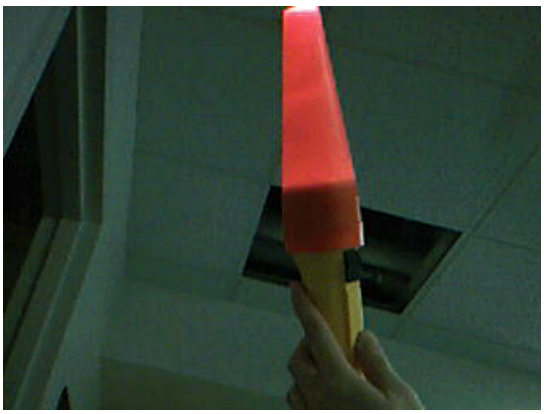
Prof. Graeme Bailey

http://cs1114.cs.cornell.edu

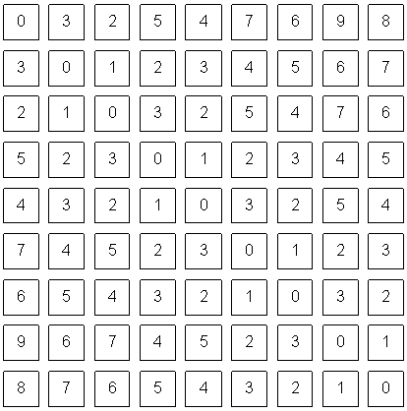
(notes modified from Noah Snavely, Spring 2009)

Major CS1114 Projects

- From a camera, figure out the position of a bright red lightstick
  - Use this to guide a robot around



What we see



What the robot sees

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?

How many black pixels?

```
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
```

What if we need to execute this code many times?

Turning this into a function

```
function [ nzeros ] = count_zeros(D)
% Counts the number of zeros in a matrix
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
```

Save in a file named **count_zeros.m**

```
count_zeros([1 3 4 0 2 0])
```

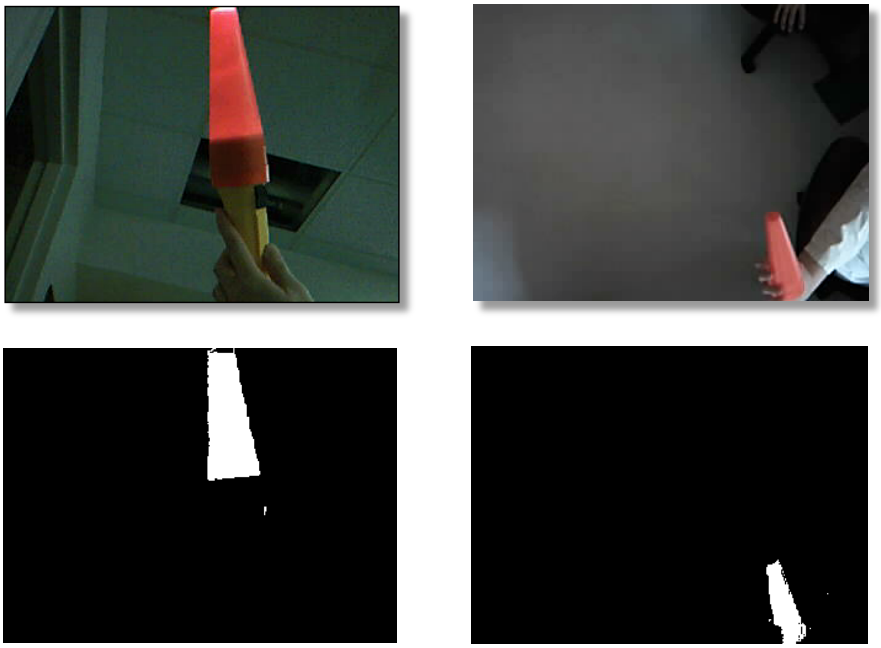

What about red pixels?

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

How many red pixels?

```
img = imread('wand1.bmp');
[red, green, blue] = image_rgb(img);
nreds = 0;
[nrows,ncols] = image_size(img);
for row = 1:nrows
    for col = 1:ncols
        if red(row,col) == 255
            nreds = nreds + 1;
        end
    end
end
```

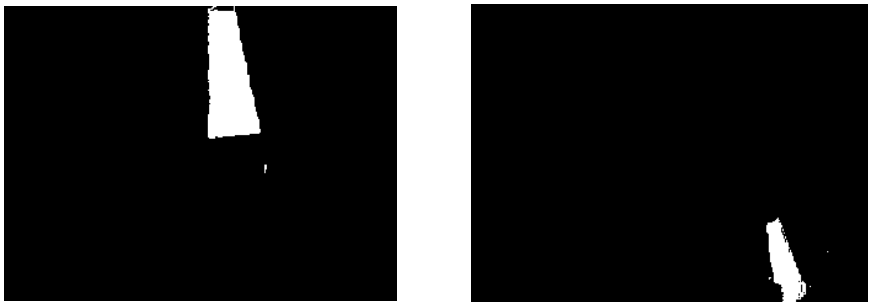

- We've counted the red pixels in Matlab
  - Can anything go wrong?



- *Question: devise a thresholding function*

Finding the lightstick

- We've answered the question: is there a red light stick?



- But the robot needs to know **where** it is!

Finding the rightmost red pixel

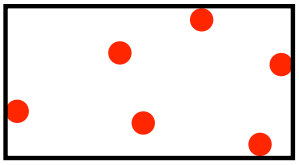
- We can always process the red pixels as we find them:

```
right = 0;
for row = 1:nrows
    for col = 1:ncols
        if red(row,col) == 255
            right = max(right,col);
        end
    end
end
```

Finding the lightstick – Take 1

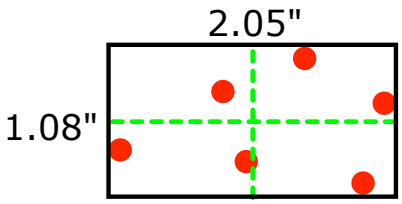
- Compute the bounding box of the red points
- The bounding box of a set of points is the smallest rectangle containing all the points
  - By "rectangle", we really mean "rectangle aligned with the X,Y axes"

Finding the bounding box



- Each red pixel we find is basically a point
  - It has an X and Y coordinate
  - Column and row
    - Note that Matlab reverses the order

What does this tell us?



Aspect ratio: 2.05/1.08 = 1.9

- Bounding box gives us some information about the lightstick
  - Midpoint → rough location
  - Aspect ratio → rough orientation
    (aspect ratio = ratio of width to height)

Computing a bounding box

- Two related questions:
  - Is this a good idea? Will it tell us **reliably** where the light stick is located?
  - Can we compute it quickly?

Computing a bounding box

- Lots of CS involves trying to find something that is both useful and efficient
  - To do this well, you need a lot of clever ways to compute things efficiently (i.e., algorithms)
  - We're going to learn a lot of these in CS1114

Before next time

- Go through all of the matlab intro
- Check to see that your CSUG account works
- *Think* about how you might find interesting regions in a static image *(don't just look it up, have fun thinking for yourself)*.