CS 1114: Finding things

Prof. Graeme Bailey

http://cs1114.cs.cornell.edu

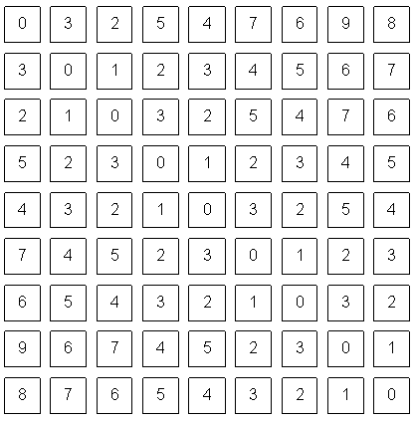
(notes modified from Noah Snavely, Spring 2009)

Cornell University
Computer Science

Major CS1114 Projects

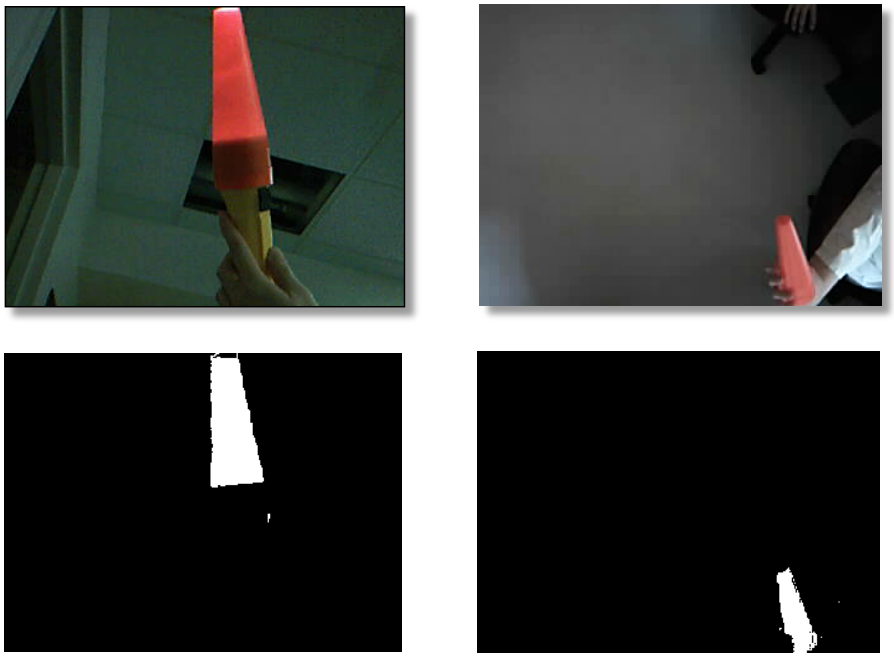
- From a camera, figure out the position of a bright red lightstick
  - Use this to guide a robot around

What we see



What the robot sees

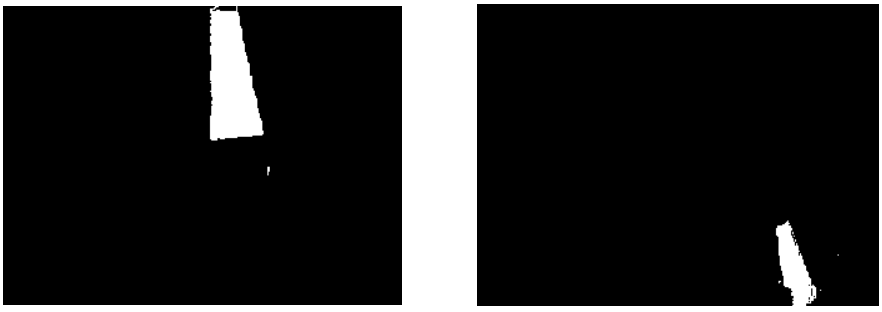
- We've counted the red pixels in Matlab
  - Can anything go wrong?



- *Question: devise a thresholding function*

Finding the lightstick

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

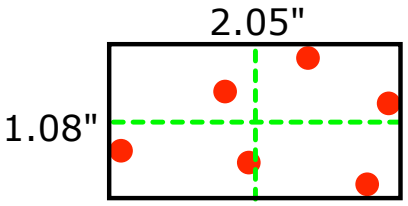


- But the robot needs to know **where** it is!

Finding the 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"

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

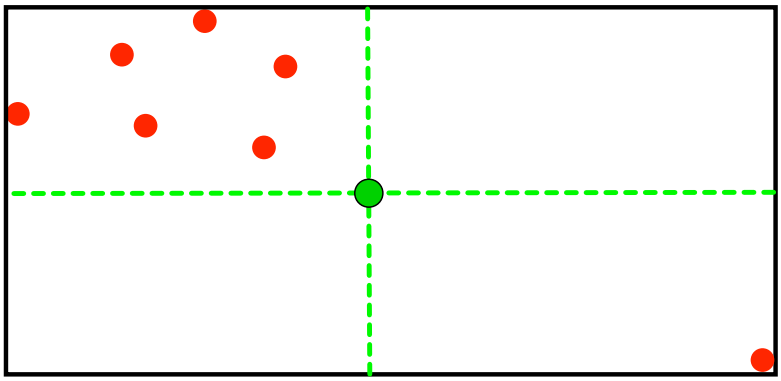
  (aspect ratio = ratio of width to height)

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?

Beyond the bounding box

- Computing a bounding box isn't hard
  - Hint: the right edge is computed by the code we showed in lecture 2.
- Does it work?



Finding the lightstick – Take 2

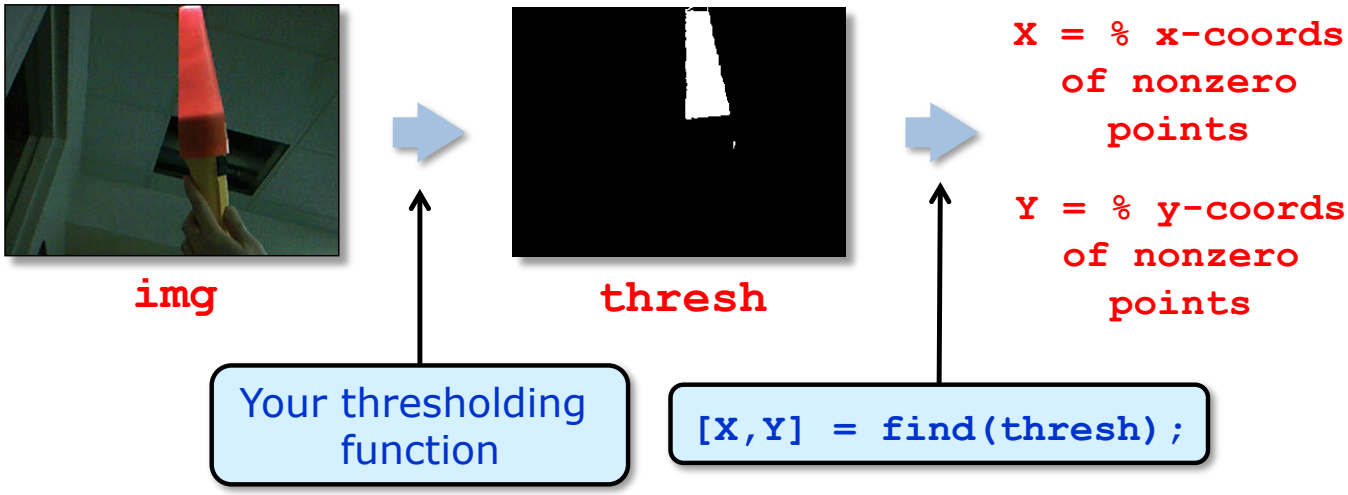
- How can we make the algorithm more robust?
  - New idea: compute the centroid
- Centroid:

  (average x-coordinate, average y-coordinate)
  - If the points are scattered uniformly, this is the same as the midpoint of the bounding box
  - Average is sometimes called the mean
  - Centroid = center of mass

Computing the centroid?

- We could do everything we want by simply iterating over the image as before
  - Testing each pixel to see if it is red, then doing something to it
- It's often easier to iterate over *just* the red pixels
- To do this, we will use the Matlab function called **`find`**

The `find` function



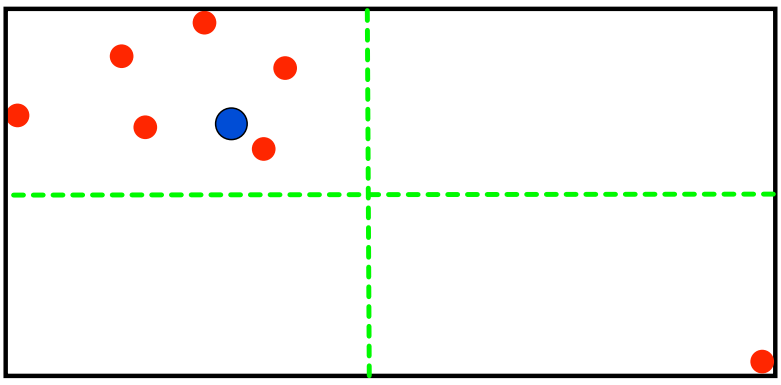
Using find on images

- We can get the x- and y- coordinates of every red pixel using `find`
  - Now all we need to do is to compute the average of these numbers
  - We will leave this as a homework exercise
    - You might have done this in high school

Q: How well does this work?

- A: Still not that well
  - One “bad” red point can mess up the mean
- This is a well-known problem
  - What is the average weight of the people in this kindergarten class photo?

How well does this work?



Aside: What's the difference between = and == ?

- Answer: a lot

- = : assignment

```
x = 2;
```

- == : test for equality

```
if x == 2
    y = 4;
end
```

- ... very important not to get these mixed up

Aside: Logical operators

- && -- logical "and"
- || -- logical "or"

```
if (x == 1 && y == 2) || z == 3
    % do something interesting
else
    % do something else
end
```

Aside: Types in Matlab

- Different types of numbers:
  - Integer (int) { 17, 42, -144, … }
    - Signed
    - Unsigned
      - **8-bit (uint8) [0 : 255]**
      - 16-bit (uint16) [0 : 65,535]
  - Floating point (double) { 3.14, 0.01, -20.5, … }

Default for images

Aside: Converting between types

- Most numbers in Matlab are **double** by default (images are an exception)
- Various functions for converting numbers:

```
double   uint8   uint16
```

- What happens when we do this:

```
uint8(200) + uint8(200) % Result = ?
```

Aside: Images in different formats

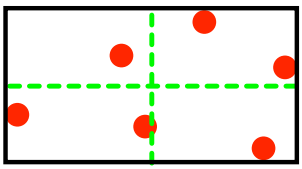
- `uint8` : intensities in range [0-255]
- `uint16` : intensities in range [0-65535]
- `double` : intensities in range [0.0-1.0]

- `imdouble(img)` converts an image to double format
- `double(img)` almost converts an image to double format

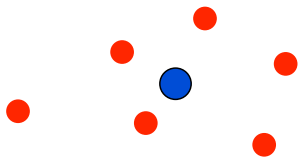
Finding the lightstick center

- : two approaches

Bounding box



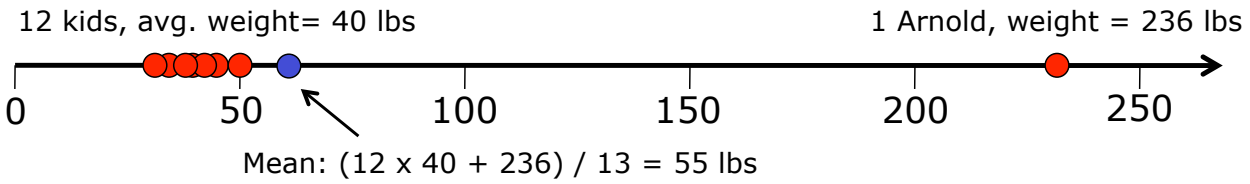
Centroid



- Both have problems…

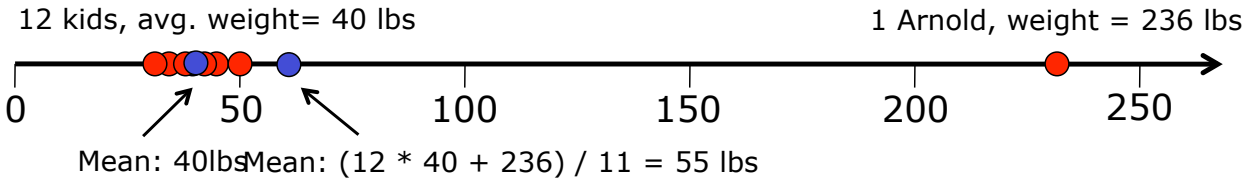
How can we do better?

- What is the average weight of the people in this kindergarten class photo?



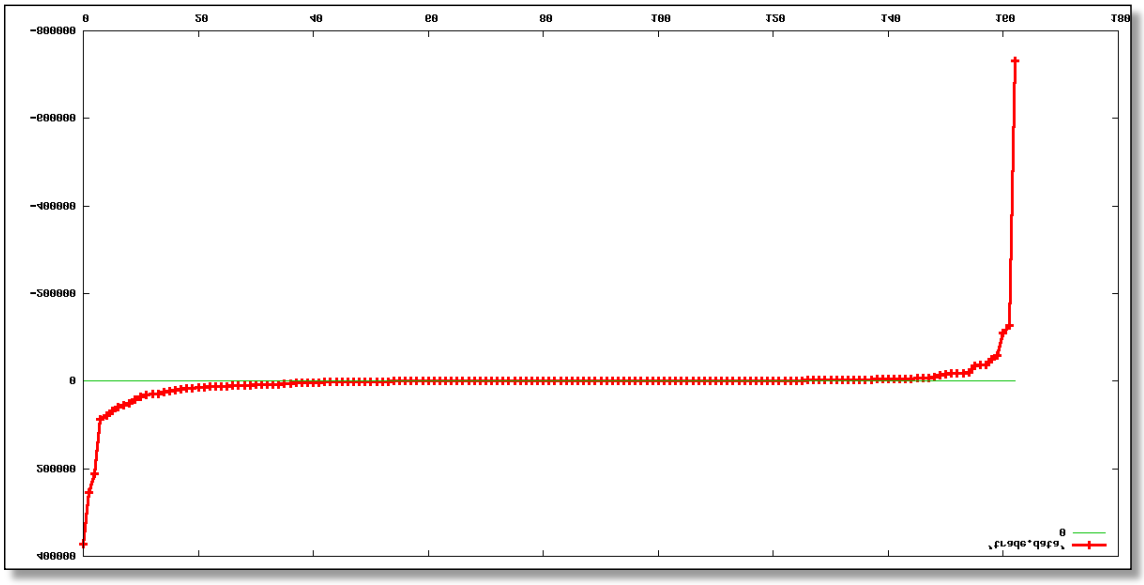
How can we do better?

- Idea: remove maximum value, compute average of the rest



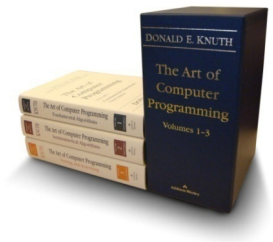
How can we do better?

- Trade deficit by country



How can we avoid this problem?

- Consider a simple variant of the mean called the "trimmed mean"
  - Simply ignore the largest 5% and the smallest 5% of the values
  - Q: How do we find the largest 5% of the values?



D.E. Knuth, *The Art of Computer Programming*
Chapter 5, pages 1 – 391

Easy to find the min (or max)

```
A = [11 18 63 15 22 39 14 503 20];
m = -1;  % Why -1?
for i = 1:length(A)
    if (A(i) > m)
        m = A(i);
    end
    % Alternatively: m = max(m,A(i));
end
```


How to get top 5%?

- First, we need to know how many cells we're dealing with
  - Let's say there are 100 → remove top 5
- How do we remove the biggest 5 numbers from an array?

Removing the top 5% -- Take 1

```
% A is a vector of length 100
for i = 1:5
    % Find the maximum element of A
    %    and remove it
end
```

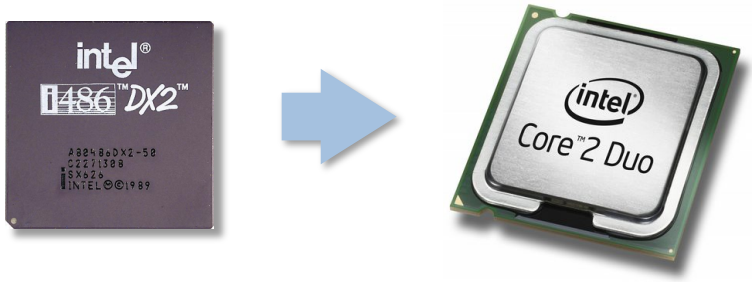

How good is this algorithm?

```
% A is a vector of length 100
 for i = 1:5
     % Find the maximum element of A
     %    and remove it
 end
```

- Is it correct?
- Is it fast?
- Is it *the fastest* way?

How do we define fast?

- It's fast when `length(A) = 20`
- We can make it faster by upgrading our machine



- So why do we care how fast it is?
- What if `length(A) = 6,706,993,152` ?

How do we define fast?

- We want to think about this issue in a way that doesn't depend on either:
  A. Getting really lucky input
  B. Happening to have really fast hardware

Where we are so far

- Finding the lightstick
  - Attempt 1: Bounding box (not so great)
  - Attempt 2: Centroid isn't much better
  - Attempt 3: Trimmed mean
    - Seems promising
    - But how do we compute it quickly?
    - The obvious way doesn't seem so great…
    - But do we really know this?

Things to do

- *Think* about how you might solve these issues of finding interesting regions in a static image *(don't just look it up, have fun thinking for yourself)*.
- Homework 1 is due next Monday – work with each other, ask questions, have fun!