Centroids and related measures

Prof. Ramin Zabih

http://cs100r.cs.cornell.edu

Cornell University
Computer Science

Administrivia

- Assignment 1 is due on Friday
  - Lots of TA time available (check the web)
- Quiz 1 is Thursday
  - Coverage through today

Find the "center" of red pixels

- We will now focus on this problem
  - Old algorithm: compute bounding box, take the box midpoint
  - First attempt to be robust: centroid
- The centroid has the average x-coordinate and the 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

Finding the red pixel indices

- 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
- However, it's often convenient to iterate over the red pixels, rather than the image
- To do this, we will use the Matlab function called `find`
  - We will cover this in section tomorrow

Using find on images

- We can get the x- and y- coordinates of every red pixel using `find`
  - This makes it simple to determine the mean x-coordinate and y-coordinate!
  - Now all we need to do is to compute the average of these numbers
  - We will leave this as a homework exercise
    - Probably you did this in high school
- So you can easily compute the centroid of the red pixels

How to do better?

- This still doesn't work that well
  - One "bad" red point can mess up the mean
    - And thus the centroid
- This is a well-known problem
  - What is the average weight of the people in this kindergarten class photo?

How can we avoid this problem?

- We'll first look at a simple variant of the mean called the "trimmed mean"
  - We will simply ignore the largest 5% of the values and the smallest 5% of the values
  - How do we find the largest 5% of the values?

Easy to find min (or max)

- We just need an initial value that is too small to be the max
  - Or too large to be the min
    - You can avoid this (how?) but it's less elegant

```
A = [11 18 63];
m = -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 are dealing with
  - Let's say there are 100
- There is a really dumb way to find top 5
  - Take the top one, remove it, repeat
  - Also works for bottom 5
- This seems like a pretty dumb approach

Can we quantify "dumbness"?

- This algorithm isn't always dumb
  - Example: if you have 20 points and you want to get the top 5%
- Algorithm isn't always slow, either
  - Chip makers like Intel tend to rescue programmers from themselves
- We want to think about this issue in a way that doesn't depend either
  - Getting really lucky some of the time
  - Happening to have really fast hardware

Summary

- Bounding box is a (bad) first attempt at finding the lightstick
  - Centroid isn't much better
- Trimmed mean seems promising
  - But how do we compute it fast?
  - The obvious way seems dumb
  - But do we really know this?
- Need to quantify how smart or dumb an algorithm is

An elegant approach exists

- You will learn it in later CS courses
  - But I'm going to steal their thunder and explain the basic idea to you here
  - It's called "big-O notation"
    - More formally, "asymptotic analysis"
- Two basic principles
  - Think about the worst case
    - Don't depend on luck
  - Think in a hardware-independent way
    - Don't depend on Intel!

Consider our algorithm

- What is the worst case?
  - Finding the median!
- Suppose our input has *n* points
  - Roughly how much work do we need to do, as a function of *n*, in the worst case?
  - Examine *n* elements to find the biggest
  - Examine *n-1* elements to find the biggest
  - …
  - Do this *n/2* times