Finding Red Pixels – Part 2

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

- You should all have access to the Upson 317 lab, CSUG accounts
  - If not, please let me know
- Your card should now unlock Upson 319
Administrivia

- Assignment 1 posted, due next Friday by 5pm
  - You should have all gotten email from me announcing the assignment
- Quiz 1 on Thursday
- No evening lecture tonight
Administrivia

- Office hours are posted on the website
Correction from last time

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

\[
\begin{align*}
D(1) &= D(1) + 20; \\
D(2) &= D(2) + 20; \\
D(3) &= D(3) + 20; \\
D(4) &= D(4) + 20; \\
D(5) &= D(5) + 20; \\
D(6) &= D(6) + 20; \\
D(7) &= D(7) + 20; \\
D(8) &= D(8) + 20; \\
D(9) &= D(9) + 20; \\
D(10) &= D(10) + 20; \\
D(11) &= D(11) + 20; \\
D(12) &= D(12) + 20; \\
\end{align*}
\]

\[
\text{for } i = 1:12 \\
\quad D(i) = D(i) + 20; \\
\text{end}
\]

\[ D = D + 20; \]

- "Vectorized" code
- Usually much faster than loops
- **But please use for loops for assignment 1**
Why 256 intensity values?

8-bit intensity ($2^8 = 256$)

5-bit intensity ($2^5 = 32$)

5-bit intensity with noise
Why 256 intensity values?

Today’s (typical) displays:  
256 * 256 * 256 = 16,777,216 colors
How many black pixels?

```matlab
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?
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
count_zeros([1 3 4 0 2 0])
What about red pixels?

\[ \text{red}(1,1) = 255, \text{green}(1,1) = \text{blue}(1,1) = 0 \]
How many red pixels?

```matlab
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
end
```
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?
Are we done?

- Assignment 1: come up with 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:

```plaintext
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”, I 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?

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

Aspect ratio: $2.05/1.08 = 1.9$
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 efficiently compute things (i.e., algorithms)
  - We’re going to learn a lot of these in CS1114
Beyond the bounding box

- Computing a bounding box isn’t hard
  - Hint: the right edge is computed by the code we showed a few slides ago
  - You’ll write this and play with it in A2

- 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

```
[X,Y] = find(thresh);
```

X = % x-coords of nonzero points
Y = % y-coords of nonzero points
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?
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
Converting between types

- Most numbers in Matlab are **double** by default (images are an exception)
- Various functions for converting numbers:
  
  \[ \text{double} \quad \text{uint8} \quad \text{uint16} \]

- What happens when we do this:
  \[ \text{uint8}(200) + \text{uint8}(200) \quad \text{\% Result} = ? \]
Images in different formats

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

- `imdbyte(img)` converts an image to double format
- `double(img)` almost converts an image to double format
For next time

- Attend section tomorrow in the lab
- Reminder: Quiz on Thursday, beginning of class