Project Goals:

Build a program to detect and read text from a high contrast images (i.e difference between text and background is clear). Build this in Matlab then create an iPhone app to do the same thing.

Goals

1. Detect Text
2. Recognize Text
3. Create and iPhone app to do the above

Implementation:

1. Method Recognize Characters
    a. Find pixels of high contrast
    b. Connect the pixels to the touching pixels of high contrast in order to box in the letters
2. Compare to letters (calls method recognize characters)
    a. Use the connecting information in order to put each letter into a box so that we know the min X and min Y of the images
    b. Scale each letter to equal the scale of the letters in the database
    c. Compare each box to a database of letters
    d. Use square differences to find best match

Hypothesis: If we use the method above then we should be able to detect and recognize any of the letters/fonts in our database along with simple fonts of different sizes outside of our database with a high accuracy.

Testing:

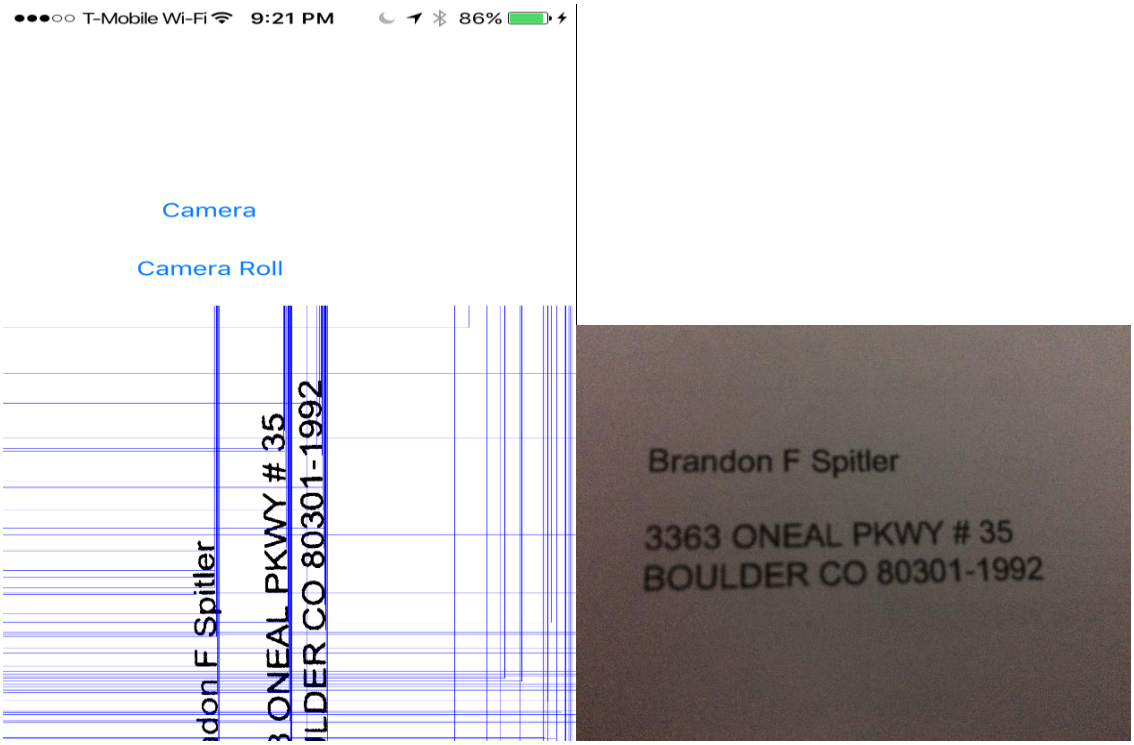
1. Test against a variety of images including
    a. Same fonts as the database
    b. Different simplistic fonts/sizes as the database
    c. With charters that are not in the database

Results

1. Recognize Characters
    a. Errors
        i. Boxes in Noise if it is to dark
    b. Works as seen in the figure below for every sentence in our database
        i. Detects dark regions and puts them into boxes.

2. Recognize Characters
    a. Detects a false letter if the above method boxes in noise.
    b. I(upper case i), i, and l's get confused
    c. False positive for 5 as S in "image.jpg"
    d. Same font
        i. "image.jpg" – "The quick brown fox jumped over the lazy dogs!"
            1. 2 false I due to noise
            2. 5 as capital S
            3. ! becomes an I
            4. Everything else is correct
        ii. "sentence.jpg" – "Lucy is going to the park and she is taking the dog for a walk."
            1. Read L lower case as i
            2. 2 false i's
            3. The "." is an i
            4. Everything else is correct

    e. Different font
        i. "1.jpg" - "HELLO WORLD"
            1. 100 percent sort
                a. Mixed up the w and o as upper and lower case
        ii. Assumed same errors as above
3. Results of IPhone app
    a. No goals achieved
        i. Characters are almost boxed in correctly
            1. For some reason the max y and max x come back as -1 even though they should only be set to the correct value and INT_MIN
                a. When min xs and maxs exs where switched in the code the min x's and y's had the same problem but the max x and y's where fixed
            2. Results below



Computer vision methods used

1. Creating black and white images
2. Looping over pixels and finding spots with dark pixels
3. Connecting the dark pixels into graphs
4. Finding the high and low x and y's of the graphs
5. Pulling sub images out of an image based on the x and y's above
6. Scaling images
7. Min squared differences

Limitations to our method

1. Weird/bad fonts will lead to false positives
2. Dark Noise will lead to false positives
3. Things not in the database will lead to false positives
4. I(upper case i), i, and l's.
5. Upside down photos
6. Font inside of dark area on the photograph
7. Scale of letter does not matter detects even if font was not meant to be read.

Future fixes

1. Gradients/angle/slope comparison to database should help detect more fonts. IE comparison of general shape instead of direct comparison.
2. Size of font should play in to the algorithm

    a. Should stop confusion of I's and l
    b. Stop dark noise leading to false positives
3. Add to the database
4. Rotations of each image to find the best match but adding weight to the same direction as every other letter/right side up so that M and W's don't get confused

Contributions

Daniel Velasco

a. Making the image black and white
b. finding high dark pixels by looping over all pixels
c. Grouping the pixels into a graph
d. Finding the high and low x and y's of each graph

Brandon Spitler

e. Scaling the letters found above to match with the size of the database letters
f. Least squared sum differences between the database and the found letter
g. Printing out the letter
h. Implementing all of the above code into an iPhone app

Things learned

1. Detecting features
    a. Finding areas of high contrast
    b. Collecting connecting dark pixels into a "box"
2. Combing algorithms such as NCCS, scaling, black and white, and future detection into a single algorithm.
3. Dealing with images outside of Matlab is awful both in debugging, and reading in the images. Matlab makes looping over the pixels really easy whereas objective c forced me to put them into one dimensional arrays making it much more complex.

Advice

Only use Matlab the iPhone app took longer than anything in this class to program (and it does not even work) because of getting used to working with images, a new language, and a new interface. It just made everything much more difficult than it had to be; we could have used the time to implemented some cooler text recognition other the least squared differences (such as gradients).

We ran into GitHub problems where for some reason I could not push to my partners GitHub. It made it extremely difficult to move forward. So before you start make sure both teammates have full access to the Github.