Project Name:
OCR (Optimal Character Recognition) & Translation

Group members:
Annie Lin
annielin@college.harvard.edu
Hillary Do
hillaryjiado@college.harvard.edu
Nhu Nguyen
nhunguyen@college.harvard.edu
Keon Ho (Chris) Lim
klim01@college.harvard.edu

Brief Overview
We are going to try to build a machine learning program that will recognize characters from an image file and return computer-readable text, such as what http://www.free-ocr.com/ does. To accomplish this, we are thinking of writing functions that will separate the image into sections for each character, then we will examine those sections to determine what character each of the sections contain. We will use different criteria, patterning matching, and criteria from previous examples to determine what the characters will be. We have different level goals for the project. Minimally, we would like to implement OCR that will recognize binary characters (0's and 1's) and translate the text from that binary. Our next goals would be to get our program to recognize all numbers, characters, then symbols. If we accomplish that, we hope to be able to recognize and translate code into text and then compile that code.

Feature List
Core Features

- We will recognize binary (0's and 1's) with our program using a core matrix matching OCR algorithm as describe in http://en.wikipedia.org/wiki/Optical_character_recognition#Character_recognition.
- We will translate that binary into text (ASCII) using CS50 knowledge http://www.cplusplus.com/doc/ascii/.

Cool Features

- With our OCR algorithm:
    - We will recognize all numbers.
    - We will recognize all english characters.
    - We will recognize symbols.
- If the inputted image contains code, we will recognize the characters, and compile the code.

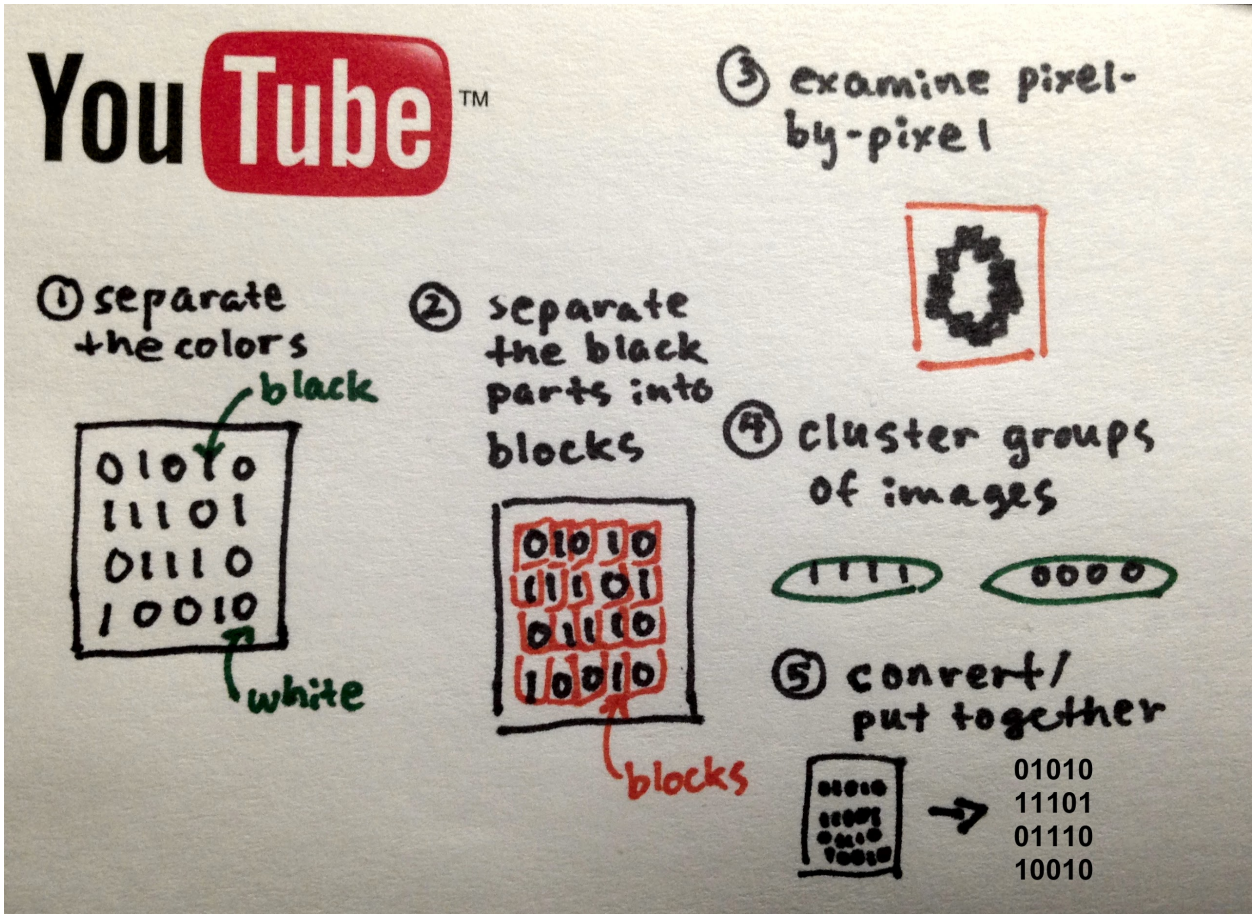
Technical Specification

We will have two classes, images and characters (numbers and symbols will be subclasses of characters).

Some of the functions we are planning on implementing include:

- If I Were A Character: We want to separate what are characters and what are not characters, so basically white from the what the color of the text will be. We are planning on doing this by utilizing color saturation, and how different parts of the image are from others.
- Partitioning: We would like to be able to separate the image into blocks that will contain individual pixels. When we come across a black pixel, we determine the bounds of that character by searching for the upward, downward, leftmost, and rightmost bounds by traversing through the contiguous black pixels; these bounds would then form the location and dimensions of the box. The boxes for each character are stored in some data structure, which are then fed into the next stages.
- Pixels Run the World: Since each character is a set of pixels, we will implement a function that analyzes the interaction between pixels and how they are arranged in relation to each other. “0” will have more black pixels than “1”. The pixels in “1” are arranged differently, specifically “1” has more lines and sharp angles than “0”, whereas “0” is more curvy than “1”. Examining the pixels intersections will tell us where the lines and curves occur. By looking at how each pixel is placed in relation to its surrounding pixel, we can define the feature of the character that defines itself. Alternatively, we can analyze the width-height relation between the characters. “0”’s width is significantly larger than “1”’s width, thus we can easily distinguish these characters based on width. The white area in between “0” needs to be taken into consideration because it determines how wide the character is. We can also do this by considering the color intensity of each pixel, since pixels on the edge of each character are lighter than pixels in between the edges.
- All the Single C*haracters*, Now Put a C*luster* on It: Suppose we have a dataset of images, each representing a number. By creating ten representative clusters, one for each number, we can group data based on how they correspond to the cluster center. We plan on creating a clustering algorithm based on measuring distance to group them. We will look at them as a 2-d vector, analyzing the mean across a set of points and group them based on that.
- Irreplaceable Text: We have to put the characters that we recognize together and convert it to a stream of text. This involves taking in a “box” (from partitioning) as an input, determining the character (from above), and appending it to a buffer/string.

We think that we could be leaving out some important functions, or missing some ways for modularization, but we will figure it out once we do more research/get more feedback and start coding.



Next Steps

Before before we begin writing our final technical specification for the next checkpoint, we will have:

- determined what language we are using. (We are thinking of doing our program in Java, but we will research other languages with good core libraries that will help to accomplish our purposes.)
- set up our environment. (We will just be using the CS50 Appliance.)
- done more research. (We will try to do more research on how other entities have done OCR, and perhaps check our implementations of OCR in other languages.)


THANKS
AARON!