Brian Scheuermann

15-112

5/4/17

Kosbie

Project Design

The problem that I am solving is in essence converting mathematics on paper into a more meaningful and usable form. At a basic level, I wanted my program to be able to read a mathematical expression and show what else there is to it. My biggest inspiration was the app, Photomath, which was popular in my area a couple of years ago. I liked its simple and friendly interface, as well as the surprising amount of mathematical statements that it could interpret. I wanted my program to be similar in that it supported a lot of different ways to interact with the math that I scanned.

I approached the problem by splitting it into three separate parts: UI, interpreting text from paper, and then interpreting that text. For the UI, I simply tried to make it as straightforward as possible, while also making the different features my program offers. I tried to make a the scanning process as much like a regular camera as possible, while also making it easier to change the area that it scans to improve accuracy. One app that I took inspiration for the UI was Desmos graphing calculator. I find this to be probably the best online graphing calculator because it looks nice, it is snappy and easy to use, and it makes most all the options that the user has as obvious as possible, without letting them get in the way.

For solving the problem of interpreting text on paper, I decided to use third party image processing and ML modules (I used OpenCV's). I decided to do this because I did not feel confident that I could get all of the other features in that I wanted if I took the time to write my

own. One thing that these modules did not support that I needed was determining the spacing of characters relative to each other (ie x^2 is different from $x2(=2x)$).To do this, I used the rectangles that were used to check for characters and found their average areas and average y values on the screen - since exponents would be much higher and smaller relative to other characters. I also used heuristics to help with shorter equations - x^2 does not have a very accurate average character size as there are only two characters, but luckily it is almost unheard of to write $x2$ as opposed to $2x$.

The part of the problem where I did most of the difficult work was in parsing the polynomials and expressions once I scanned them. To solve this, I created an evaluate PEMDAS function that searched in the order of operations to recursively simplify the expression. As for polynomials, I decided to store them as a dictionary, as that made sense for polynomials, as they are mutable (we can add and subtract polynomials), and once we simplify, each degree points to a unique coefficient. With this, I was able to more easily find derivatives to use a bisection algorithm to find zeros - to this, I found where the derivative equals zero (since there can only be at most one zero in between these) and then searched between them. To find where the derivative was zero, I had to keep using my original function to break it down into an equation that I could solve (an equation of with two different degrees, in my case). I also made sure to check the sign of the second derivative versus the sign of the function to check for turning points on the edges, so that I did not search off to infinity.