Expresso: Typesetting Handwritten Mathematical Expressions on the Post-PC Tablet Computer

[Project Proposal]

Josef Lange
University of Puget Sound
1500 N. Warner St.
Tacoma, WA 98416
jlange@pugetsound.edu

ABSTRACT

This paper sets forth to describe my project fulfilling the capstone requirement of the Bachelor's Degree in Computer Science at the University of Puget Sound. My project is to research, design, and implement a tablet-based software solution for the capture of handwritten mathematical expressions and compilation of its mathematical representation in an appropriate representative language (such as LaTeXor MathML).

Subjects approached in my project include image processing, artificial intelligence, application development for the Apple iOS platform (particularly in the tablet form factor), as well as the adherence to a highly disciplined development cycle. I will attempt to most accurately and most briefly explain any uncommon concepts described hereafter.

1. INTRODUCTION

The utility of tablet has only been recently fully realized, with the nascence of the "Post-PC" tablet, migrating away from the standard desktop usage paradigm commonly associated with computing. The modern tablet focuses on media consumption and so-called "basic" computing. In today's reality, most "Post-PC" tablets are incredibly powerful both in hardware and in software, supporting complex and challenging computations including the decoding of video, image processing, interpretation of multiple inputs, managing several network connections, and displaying high-resolution two- and three-dimensional graphics.

My project aims to utilize this now-popular device class for the capture and conversion of handwritten mathematics to relevant language to typesetting systems. Predominantly, this functionality has been researched and implemented on desktop systems, due in part to the prior lack of power in tablet devices. With modern tablets, the pieces line up: the devices already have built-in digitizers and are now powered with processors comparable in processing power to modern desktop and notebook systems.

2. EXISTING SOLUTIONS

2.1 General Handwriting Recognition

Handwriting recognition is a complex problem, only a segment of which my project wishes to solve. Existing solutions to handwriting recognition exist, and some perform at a useable level. Frequently, these systems require a period of "training" before any recognition can be achieved. Others use artificial intelligence to conclude what a sequence of handwriting is meant to be. For full-on text recognition, this is useful and almost required for any success. Commercial products exist, including but not limited to software built into Microsoft's Windows and Apple's Mac OS X Operating Systems. Additionally, many third-party companies have produced software for a similar purpose.

2.2 Mathematical Expression Capture

General handwriting recognition solutions are frequently overpowered for the niche of mathematical expressions, in which there are fewer possible logical constructs, most of which follow a fairly standard form. Because of this, the subset of things needing recognition, and possibilities for what they could be recognized as, is significantly smaller. Academic projects, including those by Matsakis[2], Smithies et. al.[3], and Levin[1] exhibit valid solutions for the desktop model of usage, though are not implemented in a way that is ultimately accessible for the common user.

3. PROPOSED SOLUTION

My project will, if successfully carried out, produce a feature-complete, useable piece of software for the Apple iOS platform. This software will have a singular function: to capture handwritten mathematical expressions via the digitizing screen, and compile the interpreted characters into the appropriate mathematical representational language (namely LaTeXor MathML).

4. MOTIVATION & IMPORTANCE

Since my childhood I've been enamored with touch devices, from Apple Newtons, to Palm Pilots, to Pocket PCs, and now to the likes of the Apple iPad, Amazon Kindle Fire, and Microsoft Surface. I've tried and tried to adopt them as a universal tool throughout my time in school with little

ultimate success. Certain applications were just lacking or not even present. Handwriting recognition is no exception. Palm's Graffiti system was uncomfortable and the software behind others' was unpredictable. When the iPad was announced and released, I was disappointed to see no integrated handwriting recognition. Soon I realized that the software keyboard would actually be more productive in the long run for actual typing. The size and shape of the tablet, though, reminded me of the small whiteboard "slates" I've used before to scribble out math equations.

At this point in time, I was particularly frustrated with a homework assignment I was working on that required me to typeset mathematical equations in LaTeX. "There's got to be a better way to do this," I said to myself, hammering away at the keyboard to correctly convert my written-out equations into code. The pieces came together and I realized how great it would be to be able to jot down an equation on my iPad and have the LaTeX code go into my document.

This software could easily improve the classroom environment in all levels of schooling. Getting mathematical equations into type is a frustrating task for many teachers in the K-12 levels, and even frustrating for the higher-education professor and student. With this software, student and educators alike can easily convert their glyphic mathematical thoughts into useable typesetting language. This will hopefully open doors to clearer and more specific teaching and learning throughout all levels of education.

5. PROJECT ROADMAP

5.1 Research

First and foremost, additional heavy research is required. Each small piece of information leads to additional sources of information and additional projects that have attempted to achieve (with varying levels of success) the same goal. There is much to be learned from each general field from which my project draws. It will be helpful to carefully analyze previous projects of this one's nature to find commonly-used methodologies when approaching handwriting recognition. Research may also help to steer my project from any abysses of hard work. The best implementation is frequently the simplest one, and as such I aim to avoid overly-complex solutions.

Research conducted will vary from understanding general text recognition, to developing a firm knowledge of mathematical representative languages, to bezier curve approximation, to distributed artificial intelligence implementation. Not all avenues of research will produce implementation, but it is my hope that most researched topics find their way into my design and implementation. The process will include careful reading of prior works, implementing rudimentary forms of algorithms investigated, and feasibility assessments of some of the more complex and optional components.

5.2 Design

A great deal of design must be done on such a big and complex project. The first task is to intelligently segment the project into smaller, possible-to-implement pieces that can be connected to create a whole system. This will require a strong knowledge of design patterns and general software engineering concepts. It is this project's intention to be comprised of several software components that may be useful on their own, and when combined properly, efficiently and elegantly solve the intended problem.

The pieces comprising the whole begin with the general application, which will fortunately be mostly complete on the creation of the project source in Xcode.

From there, building a rudimentary interface for drawing capture, then to a component that will be able to make some sort of sense of the series of pixels. The image processing component will hopefully do a lot of work to approximate drawn figures to create smoother-looking display of drawn glyphs, which should also increase their recognizability.

Once individual glyphs are identified, the biggest task: matching them with known glyphs. I hope this can eventually be tied in with a "cloud" solution that learns over time via global distributed input from all client software, much like Apple's Siri technology.

Once the glyphs are recognized, their relative positions and sizes will be assessed to create a best guess of what the mathematical expression is meant to be. This information will be stored in a data structure (most likely a tree of some sort) and then systematically converted into code for display.

5.3 Implementation

This project, as mentioned before, will need to be segmented, with each segment well-designed, in order for implementation to be manageable. The will be implemented on the Apple iOS platform using the Cocoa Touch framework. Cocoa Touch is developed primarily in Objective-C, which is compatible at compilation with C and C++. The intention is to compile and distribute for the iPad device, though the iPhone and iPod touch devices are equally capable, albeit slightly more difficult to use with such a small screen and digitizer. Apple provides a Development Program for institutions of higher education, which will negate any costs of development that Apple normally charges (namely the annual $99 membership) for developers to be able to compile their code to devices for testing and usage.

6. REFERENCES

[1] M. Levin. *CellWriter: Grid-Entry Handwriting Recognition*. PhD thesis, University of Minnesota, Minneapolis, MN, USA, 2007.

[2] N. E. Matsakis. *Recognition of Handwritten Mathematical Expressions*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1999.

[3] S. Smithies, K. Novins, and J. Arvo. A handwriting-based equation editor. Technical report, Department of Computer Science, University of Otago, Dunedin, New Zealand, 1999.