2015

Stroke Based Pen to Text

Kasey Powers

Dominique Wehner

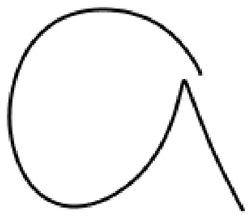
Stroke Based Pen to Text

Kasey Powers & Dominique Wehner

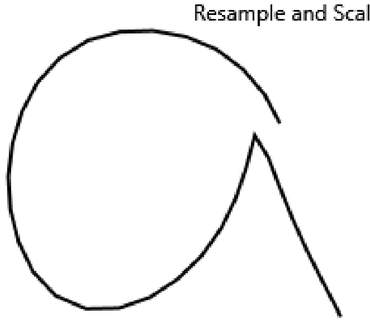
Faculty Advisor: Dr. Keith Gallagher, Dept. of Computer Science, Florida Institute of Technology

Clean Input

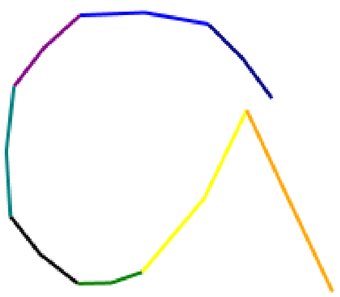
183 Original Data



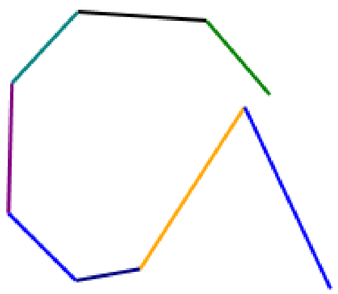
33 Resample and Scale



17 RDP Clean



9 Vector Sections



Overview

Using pen input with tablets still has its challenges, the majority of tools available are either too slow or inaccurate for daily use. Tasks like taking notes for class in a quick and legible way are impractical. Drawing based applications can make bad handwriting even harder to read. Pen input keyboards can type what is written, but are often too slow. Some tools convert drawn notes to text, but they are not reliable. Inspired to create a better option, we decided to try a different approach than we had seen in research, looking only at the point data making up the input, rather than the drawing itself.

Approach

Many handwriting recognition systems are built on finding major features, and comparing these to known letters, usually fonts, visually. These can read well printed books and documents, but handwritten inputs are often too widely varied to find an accurate match.

Simplifying the problem, we break down the letter's point data without using visual comparisons. As users write, the data is cleaned and line segments are created for each of 8 equidistant directions. We believe the order of these segments and their properties is unique enough to recognize most of the core Latin alphabet (a-z, A-Z, 0-9).

This process can be performed extremely fast, often in fractions of a millisecond, allowing it to be run faster than the user writes. As well as running quickly, we were able to exceed all goals with our cleaning step. We believe this functionality will allow the handling of input from those with active tremors. In the two inputs shown, the messy starting point has the same series of slopes outputted.

One drawback that we share with other recognition methods is recognizing characters with similar shapes such as o, 0, and O. We would try to negate this by adding context awareness of factors such as adjacent characters and their relative size. With those other probabilities calculated, the final prediction would be more accurate.

Future Goals

This project was primarily a proof of concept and has many improvements we would like to add if we were to continue working on it. First as a library, then creating applications that utilize it.

- Context probability of characters to pair with current system. Several independent probabilities should increase accuracy
- Working with blocks of input containing words by recognizing when the user has ended a character and started the next one
- Rewriting the library to be portable while also increasing speed without the need to output intermediate steps
- Further development and improvements on decision tree used for character identification
- Cleaning algorithm improvements, with feedback from active tremor groups

Math Steps

Resampling and Scaling

Starting with the original recorded data, scale the minimum value to 0 and maximum value to 1. Made more efficient by continuously adding new points on the same scale and checking them to adjust scale as needed.

Next taking the scaled data, resample so that points have a distance of 0.1 units between them. This removes a large amount of the redundant data, keeping the main shape unchanged. On very messy data some minor cleaning occurs due to the noisy points being close together. (As seen in image)

Ramer-Douglas-Peucker with Moving Average Cleaning

Using the Ramer-Douglas-Peucker algorithm with a relatively large epsilon value on the output of the previous step, major sections are found then smoothed using a moving average.

Finding major sections of the input was a massive improvement. Previous use of moving average smoothed the entire input, including smoothing out intentional sharp points. RDP allows smoothing the entire input while features large enough to be considered intentional are unchanged.

Vector Based Sections

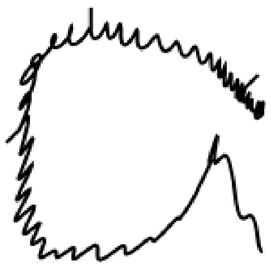
Lastly the cleaned input is broken down into a series of slopes. Determining slope by treating two points as a vector and comparing to the 8 directions using dot product. By then combining adjacent points with the same slope, the final output is found.

These sections, making up the building blocks of the character, are then able to be inputted into a decision tree built from a database of sample inputs.

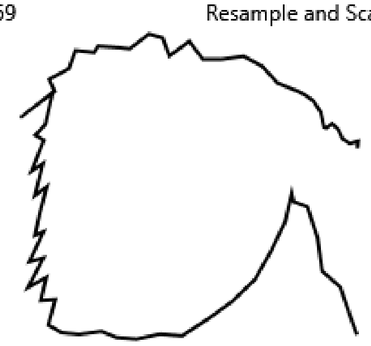
Image Legend		
# in top left corner: Data points at that step	RDP clean colors:	Sections colored with pattern
	V-Sections colors:	Slopes colored base on direction

Messy Input

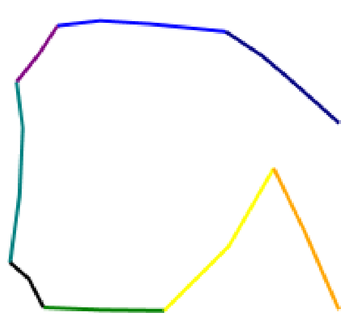
981 Original Data



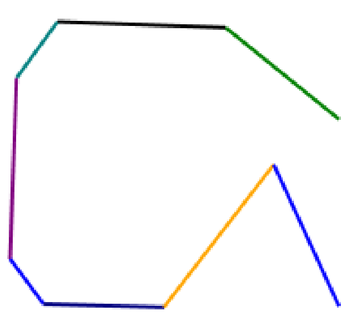
69 Resample and Scale



21 RDP Clean



9 Vector Sections



NORTHROP GRUMMAN

Engineering & Science Student Design Showcase at Florida Institute of Technology