The Applications of Image Processing Techniques to Sign Language Recognition Through a Web Camera Interface

Byron Hood

November 2, 2007

Abstract

Sign language recognition is the first step in a long road towards natural language processing, or the ability for a computer to "understand" naturally spoken language. Such an invention would drastically lessen the amount of time require for computer input, maybe even by a factor of two. This project explores using image recognition techniques such as edge detection and line detection to identify sign language in real time, using input from an average web camera ("webcam"). When research is complete, it is expected that the program will be able to identify most, if not all alphanumeric characters with a high degree of accuracy.

1 Introduction

1.1 Purpose

The purpose of this project is to provide an interface for people who speak sign language (whom I call "sign speakers") to input into a computer using their native form of communication. Such input should theoretically increase their input speed twofold, as explained in the Background section. In addition, such interpretation of sign language gestures and hand positions is a first step towards fulfilling programmers' dream of human-computer interaction nearly as old as the machines themselves, natural language processing, or the concept of a computer "understanding" naturally spoken (or in this case, signed) language.

1.2 Scope

This project will require some research into sign language hand positions and gestures, but more specific and deeper research is necessary into "image parsing" techniques (such as edge detection, line-finding, and so on). Additionally, the program will attempt to insert characters into the computer's input system, and this will require some heavy digging into either windowing system input code (so that the program can feed into X windowing system input) or operating system-level input (so that the program can emulate a keyboard). Data on hand positions will be stored in XML files read and parsed during the running time to ease the process of editing hand positions and also to make reading the said data simple when starting up; yet the project will also need some basic research into XML parsing schemes and methods of storing the XML data in memory.

The final application will recognize and interpret only alphanumeric characters for reasons of complexity. The further the program goes, the higher the complexity, which follows an exponential path. With the addition of positions, the distinctions between different positions (and therefore different letters) become smaller and less easily made. Therefore, the first step is to detect a reasonable number of different characters, and then to extend this basic program later on.

2 Background

In today's society, people with auditory and locutory disorders usually opt to commincate using sign language, a silent variation on the local language which uses body language: gestures, mouthing, and hand/finger positions, instead of spoken words. Through extensive practice and use (as normal people might gain extensive practice speaking their native language), many sign speakers are capable of "speaking" as fast as non-impaired people speak orally, about 200 words per minute in normal conversation[2].

If the average word is taken to be six letters long and one accounts for a slight speedup due to the short amount of time required to communicate a single letter, spelling a word out letter by letter will likely reduce speed by a factor of four for both sign and oral speakers. Nonetheless, this is a hefty 50 words per minute, and compares quite favorably to average typing speeds. According to Karat, et. al. the average computer user can type 33 words per minute while copying text, and this drops to a mere 19 when composing[3]. Therefore, an average computer user will spend from two to three seconds typing any given word, whilst a sign speaker (could he or she sign into a computer), would spend half of that time. Finally, the "QWERTY" keyboard was designed expressly for the purpose of *slowing* typists down (this is a throwback to the days of typewriters, to help prevent jams). Therefore, a person signing has a double advantage over a person typing: first of all, they are not inhibited by the popular keyboard, and secondly, they sign faster than the average person types.

While extensive and highly specific research has been done in the field of computer vision (as shown by the sheer number of books available on the subject), little has been devoted to the recognition of sign language, and only one study[7] has considered using a webcam-computer setup (most other modern research explores using a specialized glove to transmit data back to the computer). This is for a combination of reasons: first of all, processor power was formerly far too expensive and not powerful to process a multitude of images (with a high enough resolution to distinguish sophisticated shapes such as the human hand) in anything close to real time. Additionally, the keyboard has been—and remains—an effective, flexible, cheap, and easily extensible tool for computer input. Finally there is as of yet little demand for such a novel mechanism of input.

I have reviewed some background material, especially in image processing, much of which explained techniques such as Hough's transform for finding shapes[4], and the Robert's Cross edge detection algorithm (where R is the result and P_t, P_b, P_l, and P_r are the pixels above, below, to the left, and to the right, respectively, of the pixel on which the edge detection is being performed):

$$R = \sqrt{(P_t - P_b)^2 + (P_r - P_l)^2} \tag{1}$$

As very little substantial work (outside of Kraiss and Zieren's research[7]) has been done in this field, I am pretty much a pioneer and I must decide what path to follow on my own with little outside guidance from previous products and plans. This adds a new element of interest for me: success means that my program is one of the first of its kind in the world.

3 Development

3.1 Requirements and Limitations

A part of this project is to provide a relatively portable interface for human-computer interaction with a webcam. Therefore, the requirements of this program are rather basic. All that is necessary is a webcam—the basis of the application—and the associated drivers, a computer with Linux installed, and finally Video 4 Linux. To compile from source, a C compiler is also necessary.

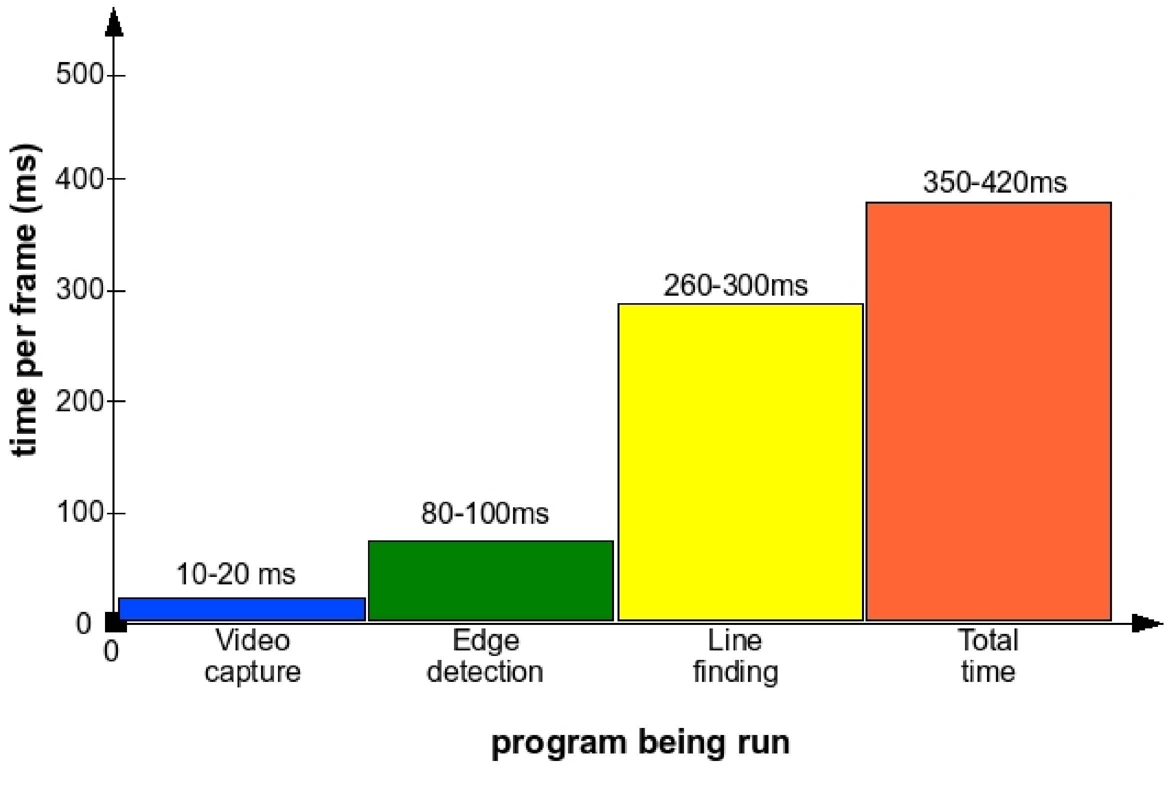
In terms of sign language recognition, the boundaries of this program will exist in terms of letters "understood" and accuracy. To simplify matters, the first program will only deal with alphanumeric characters, to provide a large enough distinction between letters to minimize some of the factors that might otherwise impede position recognition. In addition, the program will also have a limited quantity of time in which to analyze each frame, ranging from $\frac{1}{4}$ to $\frac{1}{2}$ of a second. The goal of acting in real time precludes any deep analysis of each image, and so therefore the program will inherently be somewhat inaccurate (although this may not be as much of a disadvantage as it seems; people make many mistakes at their keyboards as well).

3.2 Plan for Development

Originally, my plan was to program in the order of most testable programs first: first edge detection, then line-finding, then a line-interpreting AI, and finally an image-capture program to "grab" frames from a webcam. I soon found out, however, that this was impractical because the only good way to test a line-finding AI is to use a variety of images representing a variety of letters (otherwise I might just end up fine-tuning to a specific letter or image thereof and breaking compatibility with other letters). The best way to generate a multitude of realistic images is to capture them from a webcam; therefore my plan changed. Instead, I am working on perfecting a capture mechanism first, and then line finding and interpretation (I have already completed the edge detection phase). In brief, my plan is to program in the order of execution in the planned final product.

3.3 Testing and Analysis

The plan for testing my program(s) is rather straightforward: I will use a Python script to run each program several times and report the results and timing. Afterwards, I will inspect the image results from each portion (except the line-interpreting AI) to ensure that it is correct. In each circumstance, I will test ordinary conditions/images, boundary conditions/images, and images or conditions which should be discarded. For example, some very basic testing of four algorithms yielded these times:



This graph shows that the total processing time per frame is currently around 400ms, very much higher than the ceiling of 250ms per frame so that I can interpret sign language in real time at an acceptable pace.

References

[1] Brown, C. M. (1988). Human-computer interface design guidelines. Norwood, NJ: Ablex Publishing

[2] Omoigui, N., He, L., Gupta A., Grudin, J. and Sanocki, E. (1999). "Time-compression: Systems concerns, usage, and benefits." *Chicago 1999 Conference Proceedings*, 136-143.

[3] Karat, C. M., et. al. "Patterns of entry and correction in large vocabulary continuous speech recognition systems." *Chicago 1999 Conference Proceedings*, 568-575.

[4] Foregger, Thomas. "Hough Transform." 06 Aug 2006. PlanetMath. 28 Sept 2007. <http://planetmath.org/encyclopedia/HoughTransform.html>

[5] <http://paginas.terra.com.br/informatica/gleicon/video4linux/videodog.html>

[6] "Typewriter." *Wikipedia*, <http://en.wikipedia.org/wiki/Typewriter>

[7] Zieren, Jörg and Kraiss, Karl-Freidrich. "Robust Person-Independent Visual Sign Language Recognition." <http://www.springerlink.com/content/u5bhmbkldruml981/>