The Application of Image Processing Techniques to Sign Language Recognition Using a Web Camera

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.

by Byron Hood
Nov. 2, 2007

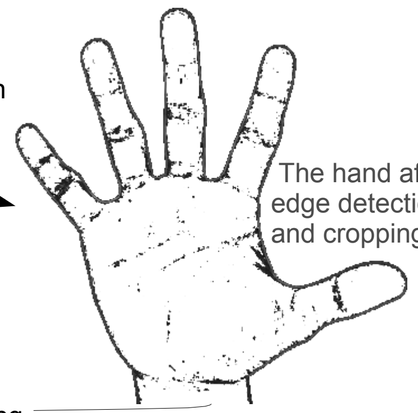
background & introduction

In today's society, people with hearing and speaking disorders communicate using sign language. Through extensive practice and use (as people gain extensive practice speaking their native language), sign speakers are capable of "speaking" as fast as others speak orally, from 200-220 words per minute.
The average computer user types 33 words per minute when transcribing and a mere 19 when composing. If the average sign speaker can communicate using finger spelling at as little as ¼ the pace of regular sign language, they sign 50 words per minute. If they could sign into a computer, this would be a significant speedup in computer input.

program procedures

The image of a hand, to be captured from a webcam. Here we use the sign for "5."

Edge detection



The hand after edge detection and cropping.

Final result:

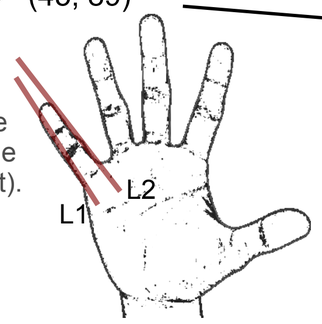
```
> ./main
  ...
Detected '5'
```

Line finding

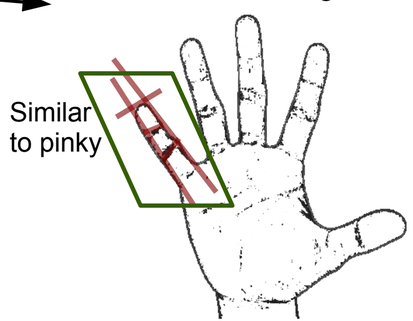
Parameterized lines:

L1: (0, 50) → (30, 95)
L2: (10, 45) → (45, 89)
...

Endpoints of the lines found in the picture (ex. right).



Line-interpreting AI:
Analyzes all of the lines in a given region and tries to match them to a finger.



Testing & timing

The program will be tested manually because automated testing would be highly impractical and would require complex image analysis. An example test would be running a program five times, and recording the time taken after each iteration, then manually viewing the results after all execution is complete.

Sample timing (done by the program):

```
> ./main
  ...
Edge detect time: 384ms
```

The Application of Image Processing Techniques to Sign Language Recognition Using a Web Camera

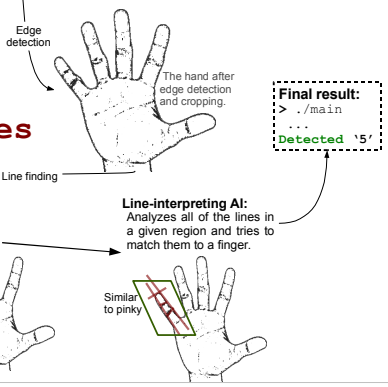
by Byron Hood
Nov. 2, 2007

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.

The image of a hand, to be captured from a webcam. Here we use the sign for "5."

program procedures



background & introduction

In today's society, people with hearing and speaking disorders communicate using sign language. Through extensive practice and use (as people gain extensive practice speaking their native language), sign speakers are capable of "speaking" as fast as others speak orally, from 200-220 words per minute.
The average computer user types 33 words per minute when transcribing and a mere 19 when composing. If the average sign speaker can communicate using finger spelling at as little as ¼ the pace of regular sign language, they sign 50 words per minute. If they could sign into a computer, this would be a significant speedup in computer input.

Testing & timing

The program will be tested manually because automated testing would be highly impractical and would require complex image analysis. An example test would be running a program five times, and recording the time taken after each iteration, then manually viewing the results after all execution is complete.

Sample timing (done by the program):

```
> ./main
 ...
Edge detect time: 384ms
```