Visual Calculator

Angus Ding (ad3180)
Ayaka Kume (ak3682)

1. Introduction

In this experiment, we will be implementing a calculator that is operated with visual input. We choose this topic for two reasons. First, calculator is one of the oldest applications of a computer. The arithmetic operations of a simple four-function calculator is fairly easy to implement, so long as the expression to be evaluated is properly received. Thus, the main effort of this project will be focused on how to design a good visual interface for the user to comfortably, accurately, and swiftly input the arithmetic expressions to be evaluated. As such, this is an application well-suited for exploring different aspects of visual interface designing.

The second reason for choosing this topic is because of the convenience of access to an established system. One of the team members has already implemented a simple version of the visual calculator as the turn in of assignment 1. By developing based on the existing system, we can focus on the improvement of the system, and thus explore deeper topics in visual interface designing.

2. Program Features

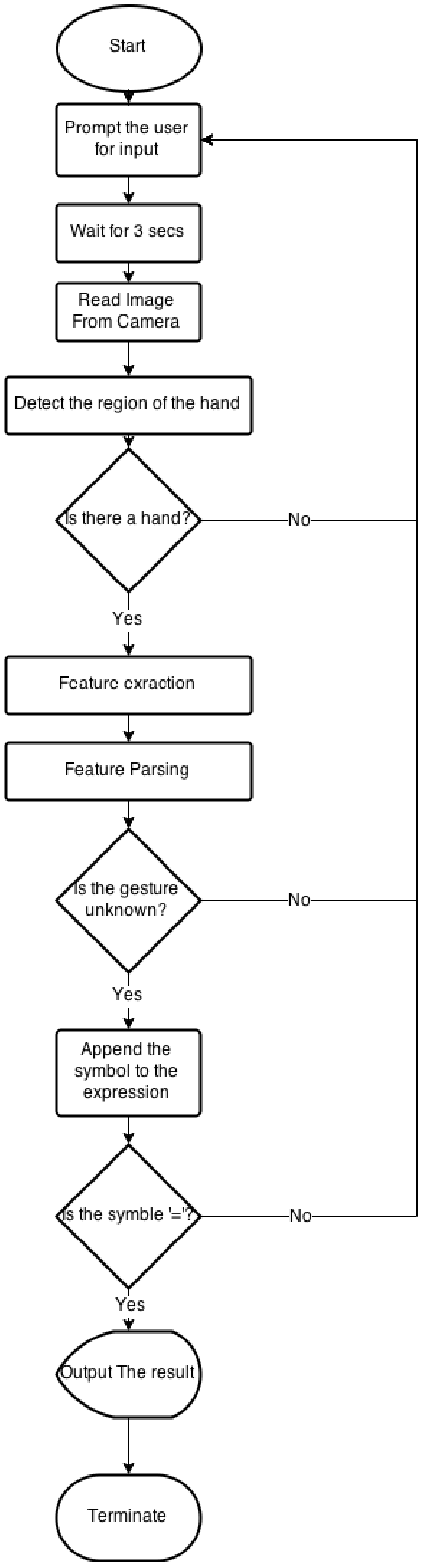
The program allows the user to input a mathematical expression through a series of predefined gestures, shows the received input in real-time, and upon receiving the evaluation command, evaluates the expression and show the result to the user.

We have some assumption about the working environment of the program. Ideally, the user wears non-skin color clothes, sits in front of the camera, with non-skin color background. The user make gestures in front of his/her clothes and not in front of his/her face.There should be no other human present in the screen. There should be only a single light source which comes from the ceiling. At first we assume the user wear long sleeve shirt so that the region of the arms are small compared to the hand region. If we have enough time we will detect a hand region from the arm region. We will be investigating techniques that enables the detection of hand under these settings. We will also explore the impact of the skin color on the detection of hand.

The list of operations we plan to implement includes addition, subtraction, multiplication, and division. The numbers will be in binary base, thus the program will only recognize numbers expressed with ‘0’s and ‘1’s. We choose the binary number system because it is easy to implement and learn, without sacrificing too much expression power. i.e. to express a large number, it doesn’t take too much more effort than using a decimal number system. The number of bits needed is only ~ 3.3219 times many as the digits needed in a decimal system. Finally, the program will also recognize a ‘evaluate’ command, equivalent to ‘=’ in a traditional hand-held calculator.

We will implement this project using python.

The system will communicate with the users using the command line. The visual input devices are the built-in cameras of the laptops, which are assumed to be located above the screen. Since different laptops have different cameras, the system will be tested on at least two different machines. The operating system supported are Windows and Linux.



3. Program Flow

The flowchart on the left illustrates the flow of the whole program. Upon starting, the program immediately prompts the user for a visual input via a message displayed on the screen. Then the program waits for three seconds, allowing the user to pose for the input. Then the program takes a picture with the camera, and load the image into memory. It then detects the region of the hand.

At first we detect the hand region. We will detect the skin color region, then extract only the hand region and make binary image of the hand. We plan to use two method, context detector and skin-based detector in combination [1]. Unlike the condition of assignment 1, there is not only a hand but also arms and a face which are also skin-colored. In [1], Mittal, et al. use the fact that the end of the arm tend to be more visible than the hand region and use machine learning. We would like to use deterministic and non-learning algorithm using some information like: the face has many non skin color regions, the hands region are smaller than face region if there is more than two separate regions.

For color detection, we would like to detect various user's skin color. In [1], Mittal, et al. detect the skin color using the information of the face in the image, which is not available for us. There are many ways to detect skin color [2], so we will try several ways and find a reasonable method with high speed and accuracy.

After we retrieve the region of the hand, we then proceed to recognize the gesture(or posture, as suggested by [4]). There are several ways to do this. In the existing system, we use hand-crafted feature mappings for different symbols. Some features used include the number of 'holes'(non-hand area surrounded by hand area), the size of the greatest 'hole', and the maximum values of the convolution of some hand-crafted filters and the image. Although the accuracy of the classification by this method is good, there are several issues. First of all, the running time of this method is not satisfying. It takes ~10s to compute the maximum value of convolution for each filter, which leads to a very frustrating user experience. To implement the new system, we need to implement more symbols. If we add a filter for each symbol, the running time will grow linearly, and will become unacceptable for an interactive application. Another issue is that this method requires a hand-crafted filter for each posture. If designed poorly, the filter may not precisely recognize the ideal posture.

To overcome these problems, there are several ways. To address the second issue, it is necessary to incorporate some machine learning techniques to train the classifier instead of hand crafting the features from scratch. An intuitive way to do this will be to have a training data set for each posture, and get the binary images of the training data, and use the average of the them as the filter. Besides this method, we will also explore the method suggested by [3] and [4]. [3] uses orientation histograms as the feature, and [4] uses haar-like filters as features. Both reports real-time performances.

After the classification of the posture, the remaining task is easy. If the posture is not recognized, then the program prompts the user to input again, without making any change to the expression so far. Otherwise, it appends the recognized symbol to the current expression. Then, if the recognized symbol is '=', representing the evaluation command, the program evaluates the expression according to arithmetic rules, and then displays the result to the user.

Since the part of the program after the posture recognition is trivial, we will focus on the evaluation of the posture recognition part, which includes the detection of hand and the recognition of postures. We will have two metrics for the overall performances of these tasks: speed and accuracy.

4. Evaluation Metrics

As an interactive application, speed is an important metric for the performance because it largely affects the user experience. The bottleneck of the program is the posture recognition, which includes the detection of hand and the recognition of postures. Since the program will wait for at least three seconds before taking the next image, the time for processing the current image should be no more than 0.3 seconds. We thus will use the average processing time as a metric for the evaluation of speed. The processing time is defined as the time elapsed since the program loads the image until the program outputs a symbol (or ‘unknown’). We will be testing with at least 120 training data, 15 for each defined posture and 15 for undefined posture.

As a calculator, correctness of the computation result is also important. Again, since the evaluation of the expression is trivial, the correctness basically relies on the accuracy of the recognition of the posture. Thus, we will use the accuracy of the posture recognition as the metric for the correctness. We will have at least 120 trials, 15 for each of the defined postures and 15 undefined postures. In each trial for the defined postures, the user tries to make the posture that represents the corresponding symbol, and the program gets 1 score if it outputs the correct symbol. The accuracy is calculated as the score divided by the number of trials for that symbol. For the trial for unknown gestures, the program gets 1 score if it outputs ‘unknown’, and the accuracy is computed similarly. Thus we will have 8 accuracies along with 1 overall accuracy. To make the system robust, all accuracies should be at least 90%.

Given that there exist many real time visual interface systems that utilizes hand gestures(instead of just static postures), we assume that it should be possible to fulfil speed requirements. However, if not, then it probably implies that there are better methods than the ones we explore.

There could be multiple reasons if the accuracy requirement is not fulfilled. Besides the lack of training data or the deficiencies of the methods, the failure may also be resulted from the nature of the problem. That is, because different users have slightly different criteria for the postures, it is impossible to get 100 % accuracy for every user with the same system. As a result, there is a natural limit on the accuracy of the system.

5. Tasks Assignment

Task	Assignee	Deadline
Project Proposal	Angus	3/26
Hand Detection	Ayaka	4/13
Posture Recognition	Angus	4/13
First System Test	Both	4/15

System improvement	Both	4/17
Final Evaluation of System	Both	4/17
Documentation	Both	5/5

Reference

[1] Mittal, Arpit, Andrew Zisserman, and Philip HS Torr. "Hand detection using multiple proposals." *BMVC*. 2011.
[2]Vezhnevets, Vladimir, Vassili Sazonov, and Alla Andreeva. "A survey on pixel-based skin color detection techniques." *Proc. Graphicon*. Vol. 3. 2003.
[3]Freeman, William T., and Michal Roth. "Orientation histograms for hand gesture recognition." *International workshop on automatic face and gesture recognition*. Vol. 12. 1995.
[4]Chen, Qing, Nicolas D. Georganas, and Emil M. Petriu. "Real-time vision-based hand gesture recognition using haar-like features." *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*. IEEE, 2007.