Assignment 3 – Symbol Recognizer
CAP5937

Due: 10/08/07 11:59pm

The focus of this third assignment is to learn the intricacies of creating a machine learning-based symbol recognizer. This is actually the first part of a two part assignment where you will be creating a simple pen-based calculator. In this assignment you are going to create a symbol recognition engine based on Rubine's 1991 SIGGRAPH paper, "Specifying Gestures by Example".

Requirements

Your symbol recognizer must be able to recognize the following symbols:

0,1,2,3,4,5,6,7,8,9,+,-,*,t,a,n,s,c,i, and the square root symbol.

You should also be able to use your scribble gesture to erase symbols.

You will also need to perform an experiment to evaluate your recognizer's accuracy. The experiment will explore how the number of training samples used per symbol affects recognition. I would suggest testing your recognizer with 5, 10, 15, and 20 samples per symbol. For the test itself, I would write each symbol 5 or 10 times which should give you a good accuracy number. Please put the results of your experiment in the README file.

Strategy

To implement your symbol recognizer, read the Rubine paper. It is fairly straightforward once you understand the mathematics.

Things you should keep in mind.

1. You need to find a way to invoke the recognizer. You can have it run in real time or in batch mode (for ex. lassoing the symbol or symbols and taping to invoke the recognizer).

2. Regardless of the invocation method, you will need some form of ink segmentation since you must be able to detect when a symbol has 2 or more strokes. Simple line segment intersection should suffice here since it is relatively easy to determine if you have a multi-stroke symbol in our alphabet.

3. Rubine's algorithm uses matrices. So make use of the Matrix library found on the course webpage.

4. Rubine's algorithm is designed to deal with only single stroke symbols. To recognize multi-stroke symbols, simply compute the features for each stroke and take the average.

5. You will need to show recognition results to the user. A simple text box is fine but if you want to be more elaborate feel free to do so.

Deliverables

You must submit a zip file containing your source and any relevant files needed to compile and run your application. Also include a README file describing what works and what does not in your application, the results of your accuracy experiment, any known bugs, and any problems you encountered. Please include a file I can open in your application that has all of the symbols written down in ink. This will show me how you wrote your symbols for testing purposes. To submit, you can email me your zip file.

Grading

Grading will be loosely based on the following:

80% correct functionality
20% documentation

Specifying Gestures by Example

Dean Rubine
Information Technology Center
Carnegie Mellon University
Pittsburgh, PA
Dean.Rubine@cs.cmu.edu

Abstract

Gesture-based interfaces offer an alternative to traditional keyboard, menu, and direct manipulation interfaces. The ability to specify objects, an operation, and additional parameters with a single intuitive gesture appeals to both novice and experienced users. Unfortunately, gesture-based interfaces have not been extensively researched, partly because they are difficult to create. This paper describes GRANDMA, a toolkit for rapidly adding gestures to direct manipulation interfaces. The trainable single-stroke gesture recognizer used by GRANDMA is also described.

Keywords — gesture, interaction techniques, user interface toolkits, statistical pattern recognition

1 Introduction

Gesture, as the term is used here, refers to hand markings, entered with a stylus or mouse, that indicate scope and commands [18]. Buxton gives the example of a proofreader's mark for moving text [1]. A single stroke indicates the operation (move text), the operand (the text to be moved), and additional parameters (the new location of the text). The intuitiveness and power of this gesture hints at the great potential of gestural interfaces for improving input from people to machines, historically the bottleneck in human-computer interaction. Additional motivation for gestural input is given by Rhyne [18] and Buxton [1].

A variety of gesture-based applications have been created. Coleman implemented a text editor based on proofreader's marks [3]. Minsky built a gestural interface to the LOGO programming language [13]. A group at IBM constructed a spreadsheet application that combines gesture and handwriting [18]. Buxton's group produced a musical score editor that uses gestures for entering notes [2] and more recently a graphical editor [9]. In these gesture-based applications (and many others) the module that distinguishes between the gestures expected by the system, known as the *gesture recognizer*, is hand coded. This code is usually complicated, making the systems (and the set of gestures accepted) difficult to create, maintain, and modify.

Creating hand-coded recognizers is difficult. This is one reason why gestural input has not received greater attention. This paper describes how gesture recognizers may be created automatically from example gestures, removing the need for hand coding. The recognition technology is incorporated into GRANDMA (Gesture Recognizers Automated in a Novel Direct Manipulation Architecture), a toolkit that enables an implementor to create gestural interfaces for applications with direct manipulation ("click-and-drag") interfaces. In the current work, such applications must themselves be built using GRANDMA. Hopefully, this paper will stimulate the integration of gesture recognition into other user interface construction tools.

Very few tools have been built to aid development of gesture-based applications. Artkit [7] provides architectural support for gestural interfaces, but no support for creating recognizers. Existing trainable character recognizers, such as those built from neural networks [6] or dictionary lookup [15], have significant shortcomings when applied to gestures, due to the different requirements gesture recognition places on a recognizer. In response, Lipscomb [11] has built a trainable recognizer specialized toward gestures, as has this author.

The recognition technology described here produces a small, fast, and accurate recognizers. Each recognizer is rapidly trained from a small number of examples of each gesture. Some gestures may vary in size and/or orientation while others depend on size and/or orientation for discrimination. Dynamic attributes (left-to-right or right-to-left, fast or slow) may be considered in classification. The gestural attributes used for classification are generally meaningful, and may be used as parameters to application routines.

The remainder of the paper describes various facets of GRANDMA. GDP, a gesture-based drawing program built using GRANDMA, is used as an example. First GDP's

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1991 ACM-0-89791-436-8/91/007/0329 $00.75