Generic and HMM based approaches to freehand sketch recognition

Tevfik Metin Sezgin MTSEZGIN@CSAIL.MIT.EDU
MIT Computer Science and Artificial Intelligence Laboratory, 200 Technology Square, Cambridge MA, 02139 USA

1. The Problem

We use sketches as a medium for expressing ideas and saving thoughts. Sketching is especially common in early design as a means of communication, documentation and as a tool for stimulating thought. Despite the increasing availability of pen based PDAs and PCs, we still can't interact with our devices via sketching as we do with people. As a group, we are building a generic multi-domain sketch recognition architecture to make computers sketch literate. This sketch recognition system will differ from existing architectures in many aspects, including a language for describing shapes, mechanisms for learning new shapes, and a blackboard based recognition architecture with top-down and bottom-up recognizers. Here we describe a part of this system that generates efficient bottom-up recognizers by compiling object descriptions.

2. Motivation

As described in (Davis, 2002), current sketch recognition systems require users to hand-code individual recognizers as well as data structures for each object to be recognized. Hand-coding individual recognizers has a number of drawbacks: *(i)* writing recognizers and data structures is labor intensive and error prone, *(ii)* extending or modifying existing recognizers requires knowing how they work, *(iii)* because recognizers may be written by different programmers and may have different recognition algorithms, they lack a unified approach to recognition, *(iv)* users usually sketch parts of objects in a certain order and style, but current systems don't have a systemic way of exploiting this information to improve recognition accuracy and speed.

3. Previous Work

Current sketch recognition systems generally either have very limited recognition, sidestep recognition to avoid problems induced by poor recognition ((Gross & Do, 1996), (Landay & Myers, 2001)), or depend heavily on other modalities such as speech (McGee et al., 2001).

Previous work in our group has focused on sketch recognition (Alvarado & Davis, 2001). One drawback of this first approach was that adding new recognizable objects required writing new recognizers and data structures. This approach also suffered from the problems of hand coding mentioned above.

4. Approach

We aim to solve the problems mentioned above by automatically generating recognizers from object descriptions. Objects are described in an object description language that includes information about components that form the object and constraints that must be satisfied in order to have a legal instance of an object (Hammond, 2002). Given an object description, our system generates the Java code for a recognizer that functions as a knowledge source in the blackboard based architecture. This automatic code generation scheme removes the need to write a separate recognizer for each object in the domain.

Our system reads object descriptions with a parser written using javacc (Sun's Java compiler compiler) and builds an abstract syntax tree (AST). The AST contains information about components, about constraints of different types, and about declarations such as renaming statements that provide a mechanism for referring to objects with a different name. The AST is processed to generate Java code for individual recognizers. The generated code includes a series of nested for-loops and conditional statements that cycle through the strokes currently on the sketching surface, looking for a permutation of the strokes that satisfy the constraints. If a particular subset of the strokes in the sketching surface satisfies all constraints, a recognition is signalled to the blackboard. If only a subset of the constraints is satisfied, the blackboard is notified about a partial recognition.

The recognition algorithm outlined above has worst-case exponential complexity. The exponential nature of sketch recognition task is also mentioned in (Mahoney & Fromherz, 2002). The root cause of this exponential complexity is the assumption that the strokes forming an object can be drawn in any order and even in an interspersed fashion, requiring that we be able to test all possible orders. But this is not typical behavior: As we found out in a user study, people sketch in a fairly predictable order (e.g., when drawing a stick figure, head is drawn first, left arm/leg is drawn before the right arm/leg). We also observed that people typically finish one object before they start drawing another. These key observations enable us to create a polynomial time recognition algorithm by building hidden markov models (HMMs) to model different sketching styles. Training data for the HMMs is generated by encoding the output of the early sketch processing toolkit described in (Sezgin, 2001). We train a separate HMM for each object class. We use these models to group strokes forming the same objects (segmentation) and find what kind of object they form (recognition) at the same time. This recognition scheme is more efficient than the brute force search method described above. Some of the results obtained using this method can be seen in Fig. 1.

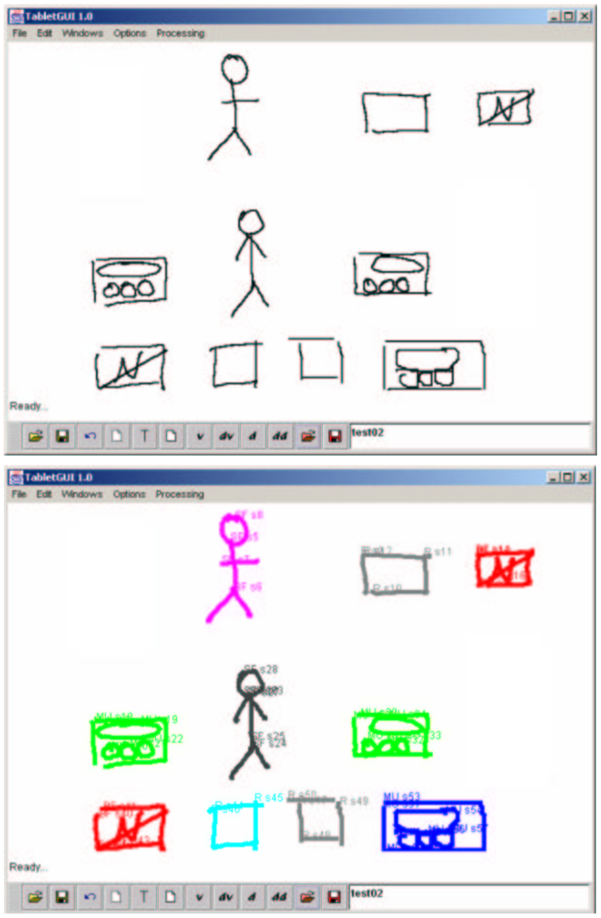
TabletGUI 1.0
File Edit Windows Options Processing
Ready...
test02
TabletGUI 1.0
File Edit Windows Options Processing
Ready...
test02

Figure 1. A number of hand-sketched objects and the recognition results (on the right). Colors indicate object class. Note that our method can also detect different sketching styles within the same object class.

Our system, along with the language described in (Hammond, 2002), helps separate the task of describing an object from how the task of recognizing it, and brings a unified approach to recognition. We also address the four problems mentioned before. We have introduced a systematic way of exploiting stroke order and drawing styles to produce robust and fast recognition.

5. Future Work

We described two sketch recognition methods. The code generation method exhaustively checks all possible stroke orderings and makes no assumptions about drawing order. On the other hand, the HMM based method utilizes users' sketching styles and runs in polynomial time. These two methods are complementary in that if the user sketches in a style captured by the HMMs, the HMM based method is ideal. If the user sketches in a style that the HMMs haven't been trained on before, the generated code can handle these cases. We are working on smart ways of combining these methods (as opposed to running them in separation one after the other), as well as improving the speed and accuracy of these methods individually. We are also investigating how hierarchical HMMs can be used in recognition and segmentation.

References

Alvarado, C., & Davis, R. (2001). Resolving ambiguities to create a natural sketch based interface. *Proceedings of IJCAI-2001*.

Davis, R. (2002). Designs for the future. *MIT Artificial Intelligence Laboratory Annual Abstract*.

Gross, M., & Do, E. (1996). Ambiguous intentions: a paper-like interface for creative design. *Proceedings of UIST 96* (pp. 183–192).

Hammond, T. (2002). A domain description language for sketch recognition. *MIT Artificial Intelligence Laboratory Annual Abstract*.

Landay, J. A., & Myers, B. A. (2001). Sketching interfaces: Toward more human interface design. *IEEE Computer, vol. 34, no. 3, March 2001, pp. 56-64*.

Mahoney, J. V., & Fromherz, M. P. (2002). Three main concerns in sketch recognition and an approach to addressing them. *2002 AAAI SSS – Sketch Understanding*.

McGee, D. R., Pavel, M., Adami, A., Wang, G., & Cohen, P. R. (2001). A visual modality for the augmentation of paper. *Workshop on Perceptive User Interfaces*.

Sezgin, T. M. (2001). Feature point detection and curve approximation for early processing of f ree-hand sketches. Master's thesis, Massachusetts Institute of Technology.