Recognition efficiency issues for freehand sketches

Tevfik Metin Sezgin MTSEZGIN@AI.MIT.EDU
MIT Artificial Intelligence Laboratory, 200 Technology Square, Cambridge MA, 02139 USA

1. Introduction

Sketch understanding has received attention as an enabling technology for natural human-computer interaction (Thomas Stahovich & Randall Davis, 2002). With the widespread availability of pen based PDAs, and more recently with the emergence of Tablet PCs, there is an increasing interest in sketch recognition. Current approaches to sketch recognition treat sketches as static images and apply structural or syntactic recognition techniques commonly used in computer vision. In this paper, we characterize sketching as an interactive, incremental process, and argue that sketch recognition algorithms should be tailored to take advantage of these properties of sketches that separate them from images. We report experimental results showing how the order in which strokes are drawn affects the recognition speed and propose possible approaches for achieving algorithms with better memory and speed requirements.

2. The problem

One property of sketches that is not exploited as much is that they are created in an incremental fashion. On the other hand, most existing sketch recognition algorithms are variants of computer vision algorithms, designed to deal with the free-hand and articulated nature of sketches despite the fact that computer vision algorithms have been developed to deal with static pictures. Starting with a blank sheet of paper to the end of the sketching process, the sketching surface sees a number of plausible scenes formed of completed objects even if the scene is not semantically meaningful. In many circumstances, the recognition system may be required to recognize such valid completed objects in the scene even if the sketch is not completed. One obvious scenario where object recognition is needed as the sketch is being constructed is when the designer of the sketch based interface wants the system to show its understanding by displaying iconic descriptions or neatened versions of the objects that it recognizes[1]. Another instance where it is required to recognize objects before a sketch is completed occurs in the case of editing. For example, in the domain of digital logic circuit sketches, if the user is sketching an RS flip-flop circuit composed of two $NAND$ gates and wires, the editing behavior of the sketching interface may depend on its operand (e.g., when the eraser part of the stylus is used on wires, it deletes parts that it touches; when used on the gates it deletes the whole all at once). Because, widely used computer vision algorithms such as interpretation tree search and subgraph isomorphism perform were not developed with this "recognize as we go" requirement in mind, they either result in poor performance or are simply not applicable.

[1] Whether or not this is appropriate for all domains or tasks is itself an interesting research question.

Model based object recognition methods in the literature perform a search either in the correspondence space, the transformation space or use a combination of two. Because sketches have a high degree of variability (e.g., non-affine scaling properties), transformation space search becomes inappropriate for recognition, thus we focused on correspondence space methods which try to find correspondences between image features and model features subject to some constraints. In the next section, we will describe how a popular correspondece space search algorithm – namely a variant of the interpretation tree (IT) algorithm – performs for continuous sketch recognition.

3. Continuous sketch recognition with the IT algotithm

The details of how the IT algorithm works is described in (Grimson, 1989). In our experimentes, we used a variant of the IT algorithm modified for continuous sketch recognition. The basic idea is to instantiate plausable partial interpretations of a given scene. The list of plausable partial interpretations is extended as new strokes are drawn. After each stroke is drawn, it is classified as a geometric primitive (line, polyline, oval, curves) using the early sketch processing toolkit described in (Tevfik Metin Sezgin, 2001). Partial interpretations are created and updated as follows. For each type object to be recognized,

- If there are no partial interpretations of the given type, and if the geometric primitive derived from the latest stroke fits into a slot[2] without violating any constraints, create a new partial template with that particular slot assigned to the primitive.

- If there are existing partial interpretations which can be extended with the latest primitive without violating any constraints, these interpretations are cloned and extended.

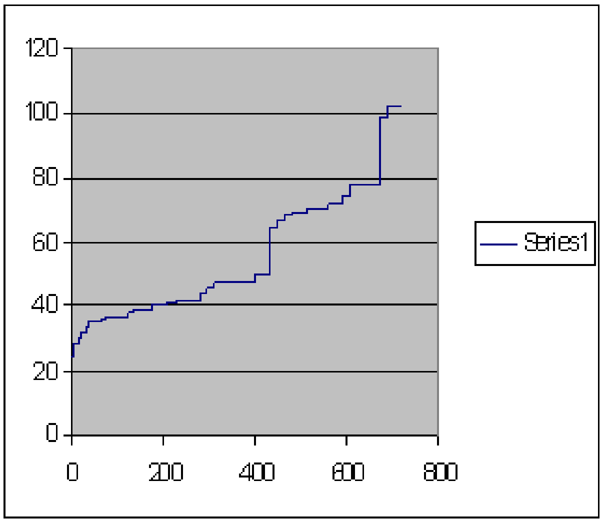

Figure 1. Number of partial interpretations generated during recognition of stickfigures drawn with different stroke orderings, sorted in ascending order. The y-axis shows the cost.

4. Experiments

In order to test how the simple recognition strategy described above performs for continuous sketch recognition, we implemented a stickfigure recognizer. Because sketches are created incrementaly and the drawing order for parts of a stickfigure can change, we tested how many partial interpretations we get for different drawing orders. We recorded raw strokes for a stickfigure and added the strokes to the sketching surface in different drawing orders to simulate different ways in which an object can be drawn. To measure the cost of recognition for a particular ordering, we used the number of partial interpretations that were instantiated at the completion of the sketch. Fig. 1 shows the costs for different orders sorted in an ascending order.

[2]By a slot, we refer to a component of a particular object model. For example, a plus sign has two slots which refer to the intersecting horizontal and vertical lines. This is referred to as object feature in the computer vision literature.

5. Results and future work

As seen in the figure, costs for different orders range from 24 to 121. The reason for this difference is that depending on how an object is drawn, the combinatoric explosion in the number of interpretations will vary. In other words, the branching factor of the corresponding IT will be different for different drawing orders. An example illustrates why this happes: For a stick figure, if we start with two touching lines, they could potentially be a pairs of arms, legs, or the body along with any of the other limbs. On the other hand, if we have an oval touching a line, these strokes can only be the head and the body. So, in one case there are multiple ways in which two strokes can be labeled and in the other case the labeling is unique. In the computer vision literature, these kinds of combinatorial explosion in search is controlled by actively searching highly constraining feature sets to initiate the recognition (the concept of *key* or *anchor* components). However, in continuous sketch recognition, there is no guarantee that the key components will be sketched first. We believe that a more appropriate recognition strategy would be to delay the labeling of strokes until the *key components* of a particular object are drawn. This is indeed a hard problem because it requires knowing the current state of a drawing. In this case, determining the exact state would be as costly as enumerating all plausable interpretations so we propose to use an *estimate* of the state. The idea is to use an estimate of the current state along with the knowledge of the search space (learned offline from object descriptions) to guide the search to minimize combinatoric explosion. This approach is closely realted to decision theoretic and non-myopic approaches to search, and literature on (PO)MDPs and planning. We are currently investigating this proposed framework as a more CPU and memory efficient approach to sketch recognition.

Acknowledgements

I would like to thank my thesis advisor Prof. Randall Davis for his supervision.

References

Grimson, W. E. L. (1989). The combinatorics of heuristic search termination. *AI Lab Memo 1111*.

Tevfik Metin Sezgin, Thomas Stahovich, R. D. (2001). Sketch based interfaces:early processing for sketch understanding. *Proceedings of PUI-2001, November 2001*.

Thomas Stahovich, J. L., & Randall Davis, C. (2002). A framework for multi-domain sketch recognition. *AAAI Spring Symposium: Sketch Understanding March 25-27, Stanford CA*.