Abstract:
This article presents a knowledge-based framework for sketch understanding. Compared to traditional sketch understanding systems, it has two significant features: (1) knowledge-based. This makes the system independent of applied domain. Users can easily design an ontology-based domain knowledge base to guide the understanding process. (2) Hierarchical approach. The sketch understanding process is divided into three phases: token recognition, salient symbol recognition and integral recognition. During all these phases, domain knowledge is used as a guider. In addition, a domain knowledge base of UML Class Diagram is presented as a case study of this framework, in which the details of how to design ontology-based knowledge base and how to implement a hierarchical recognition are given. Some UML Class Diagram sketches are used as experiments, which produce satisfactory results.

Keywords:
Sketch understanding; ontology; Knowledge base

1. Introduction

During the past years, as a result of fast development of computer hardware techniques, the computing capability of PC grows more and more powerful, which makes intelligent human-computer interaction become widely available, such as handwriting recognition, voice recognition etc. Among those, sketching is a natural, efficient, and convenient way for human-computer interaction. The fundamental challenge in sketch-based human-computer interaction that distinguishes it from other types of interaction mechanisms has something to do with the difficulty in interpreting hand drawings, which is called as sketch understanding.

Research in automatic understanding of freehand drawings has produced a wide variety of representation and recognition techniques, such as graph-based methods [1], parameter-based methods [2] and machine learning methods [3]. Nevertheless, even the latest experimental systems are typically limited to basic shapes, such as rectangles and circles, etc. Generally speaking, sketch understanding is a difficult task and the underlying factors that determine the true identity of freehand sketches remain to be intractable. It is probably safe to say that no simple scheme is likely to achieve high recognition and reliability rates without the support of visual knowledge about sketching.

However, almost none of the previous works uses visual knowledge as a key factor in sketch understanding. As a result, they cannot understand complicated freehand drawing, such as a UML design diagram, which contains a number of classes, interfaces and other kinds of composite symbols. In order to solve this problem, we propose a novel knowledge-based framework for sketch understanding, in which we use visual concept ontology [4] to build a domain knowledge base to support the recognition process.

Compared to the previous works in sketch understanding, our framework has two significant features. The first one is knowledge-based. Domain related visual knowledge base is designed using ontology which originates from the ontological engineering community. We use visual concept ontology to hide the low-level vision layer complexity of sketches, and users can easily adapt this framework to any other domain by designing a high-level knowledge base. The second feature is hierarchical approach, which results from the high complexity of low-level information in sketch. Usually, a sketch contains dozens of strokes, which is defined by the sequence of points between successive "pen-down" and "pen-up" events. Grouping these strokes into certain composite symbols (such as classes, interfaces in UML Class Diagram) cannot be solved in one step. Therefore, we divide recognition process into three phases: token recognition, salient symbol recognition and integral recognition.

The rest of this article is arranged as follows: In section 2, some previous works in sketch understanding are reviewed as a background of our work. In section 3, we introduce the architecture of our novel knowledge-based framework for sketch understanding. In section 4, UML Class Diagram is used as a case study to demonstrate that our framework does work. In section 5, we give a conclusion to this article and present some future works.
2. Related Works

Sketch-based interaction and sketch understanding draw a lot of attentions in both academies and industries in the past ten years. For user interface design, Landay and Myers [5] present an interactive sketching tool called SILK that allows designers to quickly sketch out a user interface and transform it into a fully operational system.

Sezgin et al. [6] focus on the very first step in sketch understanding: interpreting the pixels produced by the user’s strokes and producing low level geometric descriptions such as lines, curves, ovals and their combinations. Kara and Stahovich [7] address the issues of parsing and recognition of hand-drawn sketches in the domain of network diagrams. But it is too dependent on domain because they assume that non-arrow symbols are connected by means of arrows in diagrams.

Some of other works utilize the domain knowledge, but only as an assistant. Gennari et al. [8] present a sketch understanding system for network-like diagrams consisting of symbols linked together. Domain knowledge and context are used to correct parsing and recognition errors in the last step.

However, these works do not emphasize the importance of domain-related visual knowledge. As a result, their approach could only apply to specific domain.

3. Framework of sketch understanding

The framework of our sketch understanding system is shown in Figure 1. It consists of five components: Input, Recognizer, Knowledge Base, Ontology, and Output. The Input component converts user’s sketch into computer understandable form. This can be done by scanner or freehand-writing pad. The Output component displays the final understanding result to users in visual form. The Knowledge Base contains domain visual knowledge needed for sketch understanding. Ontology is used to design and build knowledge base in our framework. The details of knowledge base and ontology are given in section 3.1 and 3.2. The Recognizer component processes input sketch data using a hierarchical approach with the support of knowledge base, details are given in section 3.3.

3.1. Knowledge base

In our framework, one of the most important components is Knowledge Base. It contains several kinds of domain knowledge needed for the understanding process, including: token knowledge, symbol knowledge, context knowledge, and symbol recognition routines.

First, we are going to introduce some basic concepts that will be frequently used:

Concept 1: Stroke is a sequence of sampled points between successive pen-down and pen-up events. The point can be parameterized as \((x, y, p)\), in which \(x\) and \(y\) refer to the coordinates of a point, \(p\) is additional parameter that could be drawing speed or pen pressure etc.

Concept 2: Token is a predefined geometric unit (such as line segment, oval, curve etc.) that is extracted from stroke by some sketch recognition algorithm adopted from [6].

Concept 3: Symbol is a domain-related visual object (such as a resistance in circuit diagram, or a class in UML diagram). It has certain semantics that is meaningful to human.
case, these relationships are described in geometrical form. Take two lines \( l_1, l_2 \) as example, the relationships between them can be described as \( l_1 \) is orthogonal to \( l_2 \), and they cross each other. (2) The axioms of the sketch domain. In any given domain, when a user tries to sketch something, he must follow some rules, e.g., when sketching an inheritance symbol in UML Class Diagram, the user must draw it between two class symbols. These rules are axioms. 

*Symbol recognition routines* indicate the knowledge about how to extract one salient symbol from tokens, and how to infer regular symbols with context knowledge.

### 3.2. Ontology

We see above that visual domain knowledge contains lots of concepts, such as tokens and symbols; even the context knowledge can be described in relational concepts such as orthogonal. Because “an ontology is an explicit specification of a conceptualization” [9], we adapt ontology to design and build knowledge base. An ontology is composed of the following entities [4]:

- a set of concepts \( (C) \);
- a set of relations \( (R) \);
- a set of axioms \( (A) \).

Two partial orders \( \prec_c \) and \( \prec_r \) define the concept hierarchy and relation hierarchy, respectively, while \( A \) is supposed to be the support of reasoning mechanisms. In knowledge base, the tokens knowledge and symbol knowledge are designed with \( C \), the context knowledge is designed with \( R \) and \( A \), symbol recognition routines are designed with \( A \). (You can find these in Figure 1)

When we use ontology to build a knowledge base, the process must be done in four distinct phases [10]:

1. **Specification**: states why the ontology is built and who the end users are.
2. **Conceptualization**: leads to a structured domain knowledge.
3. **Formalization**: transforms the conceptual model into a formal model.
4. **Implementation**: transforms the formal model into a computational model.

### 3.3. Recognizer

Recognizer’s major work is to accept user’s input sketch and finally give the understanding result the user. From Figure 1, we can see that recognizer is composed of three sub recognizers, which are:

**Token Recognizer**: token recognizer firstly read the digital samples of input sketch (represented as strokes) from Input component, and then some recognition techniques are used to extract tokens from these strokes. Tokens that will be recognized are defined in token knowledge as part of \( C \) (set of concepts) in knowledge base. In this paper, we adopt token recognition techniques from [6]: using curvature to detect vertices, using Bezier curves to approximate curve regions.

**Salient Symbol Recognizer**: Usually, the number of tokens in one sketch is from 20 to 50, it will reach hundreds in some special case. Due to this reason, recognizing all the symbols directly from input tokens is time-consumed and dangerous. To make it clear: firstly, recognizing symbols from tokens is similar to sub-graph isomorphism problem, which is known as a NPC problem. When the number of tokens grows, the time complexity of symbol-finding process grows exponentially. Secondly, when number of tokens grows, the relations between tokens will become more complex, which will make the error rate of symbol-finding process increase, since many of the relations are redundant, they may interfere with the recognition process.

Therefore, in order to improve the speed of understanding process and the correct ratio of understanding results, we propose to recognize salient symbol first. Salient symbol recognizer does these jobs: (1) acquiring salient symbols information from knowledge base; (2) invoking salient-finding routine in knowledge base to find salient symbols; (3) updating context (e.g. removing the used tokens from context, adding recognized symbols in context).

**Integral Recognizer**: After salient symbols’ recognition, integral recognizer uses recognized symbols, unused tokens and the context of sketch to get a total understanding of the input sketch. Integral recognizer does two jobs: (1) recognizing the rest regular symbols (that is \( S - S_S \) with the definition of formula 1); (2) eliminating some useless tokens. These jobs need the support of context knowledge in knowledge base. The recognition process here is more like an inference. Since we can not directly use tokens and their relations to find out what symbols they are composing, we should use recognized salient symbols and domain-related context knowledge to infer or guess what symbols these tokens are composing, and then use tokens’ information to verify the inference or guess.

### 4. Case study: UML Class Diagram

In this section, we take UML Class Diagram [11] as an example to show how to apply our framework to realize
4.1. Designing knowledge base

To design an ontology-based knowledge base for UML Class Diagram domain, we should first define the visual concept ontology ($VO$):

$$VO = \{C, R, A\}$$

(2)

Where $C = \{\text{Stroke, Symbol}\}$, $R = \{\text{Ang, Dir, Pos}\}$, and $A$ is a set of axioms that consists of inference rules. The concept hierarchy is shown in Figure 2.

When reading a sketch, people usually get its meaning by using geometrical relations between tokens. Therefore, we adapt three important geometrical concepts to describe the relations in ontology, which are:

- $\text{Ang}$ is the angle between two lines' vector direction.
- $\text{Dir}$ is the direction of a vector which starts from the center of first line, ends at the center of second line.
- $\text{Pos}$ is the positional relation between two lines.

The relation hierarchy is shown in Figure 3.

After defining concept and relation, we can now build inference engine (set of axioms) for knowledge base. The content in inference engine includes: (1) what are salient symbols and what are regular symbols; (2) how to recognize the symbol. The rules should be organized with first-order logic using predicates defined in visual concept ontology (Figure 2 and Figure 3). Here is a general example (in Prolog style):

```prolog
% salient symbols recognition routine
salient([class, interface]).
class :- (rules for finding a class).
interface :- (rules for finding a interface).

% regular symbols recognition routine
regular([inheritance, realization]).
inheritance :- (rules for finding a inheritance).
realization :- (rules for finding a realization).
```

4.2. Hierarchical approach of sketch understanding

Figure 4 shows an example of hierarchical approach of sketch understanding:

Figure 4(a) is the input sketch; Figure 4(b) shows recognized tokens that is extract from Figure 4(a); In Figure 4(c), three salient symbols are recognized, which are two classes and one interface; With recognized salient symbols, we can easily recognize two regular symbols (inheritance and realization) in Figure 4(d).
4.3. Experimental results

Using JAVA, SWT [12] and SWI-Prolog [13], we have implemented a prototype system of sketch understanding that is called MagicSketch. Till now, we have only implemented a knowledge base for UML Class Diagram domain. We use hand drawing pad as input tool, and draw several UML diagrams from simple to complex to test MagicSketch’s function. The experimental results are shown in Figure 5, where inputting sketchy diagrams are shown on left side and understanding results on right side: (a) is a simple example of UML sketch; (b) is an example with three symbols (two classes, one interface); (c) is an example in which useless tokens can be eliminated by recognizer; (d) is a complex example with five symbols (about 40 tokens).

5. Conclusion

This article introduces a framework of knowledge-based hierarchical sketch understanding. The major features of our framework are: (1) knowledge-based, which makes our framework be used independent of domain. User can easily adapt it to any domain by designing high-level knowledge base with visual concept ontology; (2) hierarchical approach, which is a fountainhead that our framework can handle any complex input sketch with dozens of tokens.

In addition, we have implemented a knowledge base of UML Class Diagram to enable our framework to understand UML sketches. We test our system with various experiments from simple case to complex one (Figure 5). The experiments produce very good and satisfactory results. We are now working on designing a domain knowledge base of circuit diagram, which will demonstrate that MagicSketch is a domain independent sketch understanding system.

Acknowledgements

The work described in this paper was supported by a grant from National Natural Science Foundation of China (Project No. 69903006 and 60373065) and the Program for New Century Excellent Talents in University of China.

References


