A Multi-Agent System for Hand-drawn Diagram Recognition

G. Casella[1,2], V. Deufemia[1], V. Mascardi[2]

1 DMI – Università di Salerno
Via Ponte don Melillo
84084, Fisciano (SA), Italy
deufemia@unisa.it

2 DISI – Università di Genova
Via Dodecaneso 35
16146, Genova, Italy
mascardi@disi.unige.it
casella@disi.unige.it

Abstract

In this paper we present AgentSketch, an agent-based system for on-line recognition of hand-drawn diagrams. Agents are used for managing the activity of symbol recognizers and for providing efficient interpretations of the sketch to the user thanks to the use of contextual information for ambiguity resolution. The system can be applied to a variety of domains by providing recognizers of the symbols in that domain. A first experimental evaluation has been performed on the domain of UML use case diagrams to verify the effectiveness of the proposed approach.

1. Introduction

In the last years sketch recognition algorithms have obtained a growing importance mostly thanks to the diffusion of tablet PCs and intelligent pen-based interfaces. In particular, sketch-based user interfaces have been developed for a variety of different disciplines, including engineering design, user interface design and architectures.

The problem of sketch recognition is critical to the success of such interfaces, since the recognition accuracy should make them useful and predictable to end-users. The sketchy graphic objects drawn on a tablet using a digital pen are usually not easy for machines to understand and process. This problem is particularly difficult since sketches are almost never drawn perfectly, they contain internal inconsistencies even when drawn by the same person.

In this paper we present AgentSketch, an agent-based system for the on-line recognition of symbols in diagrammatic sketches. An intelligent agent is a software or hardware entity aware of its dynamic and unpredictable environment, able to react to the changes that take place in it, driven by long-term goals, autonomous, and social . Exploiting agents for sketch recognition represents a suitable solution since the activity of drawing with sketch-based interfaces represents a dynamic and unpredictable environment, and the recognizers of diagrammatic symbols have to possess the distinctive features of agents previously describe.

At the lowest level of our system, the symbols of the domain language are recognized by applying suitable symbol recognizers to the interpretations of elementary strokes. Several symbol recognizers , , , can be integrated into our approach in spite of the fact that they differ one from another under several aspects. Their execution is coordinated by Symbol Recognition Agents, which also cooperate with other agents to compute contextual information for the symbols they are recognizing. The interpretation of the whole sketch is accomplished by the Sketch Interpretation Agent, which disambiguates the recognized symbols exploiting the domain context.

We have applied AgentSketch to the domain of UML use case diagrams , and performed a preliminary evaluation study. The obtained results have highlighted the effectiveness of the proposed context-based recognition approach.

The paper is organized as follows. In Section 2 we describe the proposed multi-agent recognition system and a preliminary evaluation is presented in Section 3. Finally, we discuss the related work in Section 4 and give the conclusion in Section 5.

2. AgentSketch

AgentSketch is a system for recognizing freely hand-drawn diagrammatic sketches exploiting the multi-agent framework presented in []. It can be applied to a variety of domains by providing recognizers of the symbols in that domain. Its architecture for recognizing UML use case diagrams is shown in Fig. 1. The system is composed by four kinds of agents:

Interface Agent. It represents an interface between the agent-based system and the generic "Input Suppliers".

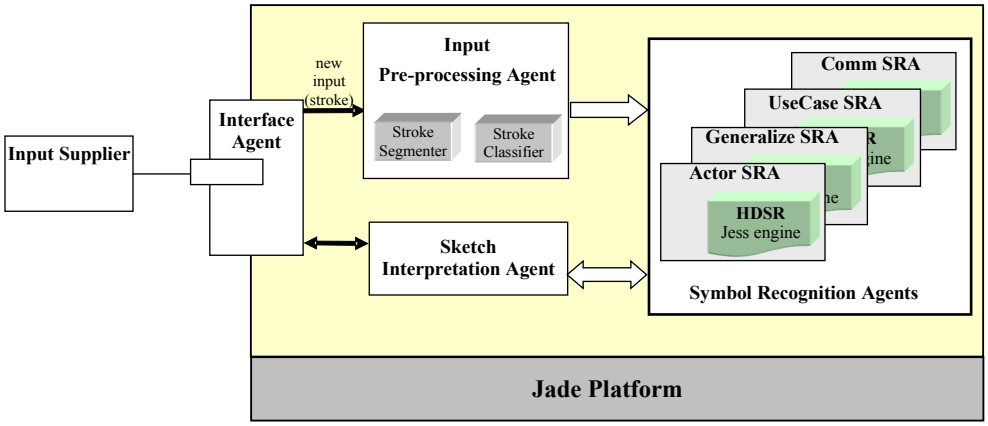

Figure 1. The AgentSketch architecture

The nature of these input suppliers may vary according to the type of sketch to be interpreted and to the drawing process (on-line vs. off-line). For example, they might be editors suitable for on-line drawing of diagrammatic sketch, as well as interfaces that allow the user to select an existing image to be interpreted (useful for an off-line recognition process). The Interface Agent is responsible for converting the information produced by the input suppliers into a format compliant with that used by the "Input Pre-Processing Agent" and by the "Sketch Interpretation Agent" (SIA), and vice versa. It sends each new available piece of input (or the whole input, in case of an off-line recognition process) to the Input Pre-Processing Agent, and interacts with the SIA for sending the user's interpretation requests to it, and for delivering its answer to the user. For any new input supplier to be linked to the framework, a new "stub" inside the Interface Agent must be created ad-hoc.

Input Pre-Processing Agent. The main activity of this agent is to classify the user strokes into a sequence of domain independent primitive shapes. It is composed of two modules, the Stroke Segmenter and the Stroke Classifier. Both modules have been implemented using third parties software and, for their highly modular implementation, might be easily replaceable with other pieces of software with the same purpose. The Stroke Segmenter segments strokes into substrokes by identifying key points. In this way, symbols can be drawn with multiple pen strokes, and a single pen stroke can contain multiple symbols. Each substroke may be classified by the Stroke Classifier as either a line, an arc or an ellipse. However, if the user stroke is imprecise, then the Stroke Classifier may classify it as more than one primitive shape (for example, both as a line and as an arc).

Symbol Recognition Agents. Each SRA is devoted to recognize a particular symbol of the domain and collaborates with the SIA that deals with the sketch interpretation activity. Moreover if the graphical language in which the sketch is drawn allows to exploit contextual information for interpreting each symbol in the right way, SRAs may collaborate with other SRAs in order to apply context knowledge to the symbols they are recognizing. As an example in UML use case diagrams , there will be a SRA devoted to recognize *generalize* symbols by managing the execution of Hand-Drawn Symbol Recognizers - HDSRs for short - for the *generalize* symbol, and by collaborating with other SRAs to obtain contextual feedback. SRAs try to recognize domain symbol by applying suitable HDSRs to the stroke classifications produced by the Input Pre-Processing Agent. The Input Pre-Processing Agent sends a stroke classification to an SRA only if the SRA may use it to recognize new domain symbols (i.e. strokes classified as ellipses will not be sent to the SRA devoted to recognize the generalize symbol, which does not include ellipses).

When a new symbol has been recognized by a HDSR, the corresponding SRA starts the collaboration

process with other SRAs for obtaining contextual information. The collaboration consists of sending a feedback request message containing the attributes about the recognized symbol to all the SRAs that recognize related symbols (and that are known "a priori" by each SRA). When an SRA receives a feedback request message, it checks its set of recognized symbols to give a response. If it finds a symbol that satisfies the language relationship with the symbol in the feedback request, it sends a positive response to the requester; otherwise, it sends a negative response. For example, due to the rules that govern the definition of well-formed UML use case diagrams, an SRA that recognizes an *actor* symbol should ask a feedback to the SRA that recognizes *communication* symbols, since *actors* participate to *use cases*, to which they are connected via a *communication* symbol.

Sketch Interpretation Agent. The SIA provides the correct interpretation either of the sketch drawn so far (in case of an on-line drawing process) or of the entire sketch (in case of an off-line recognition process) to the Interface Agent. In particular, it analyzes the information received from SRAs and solves conflicts that might arise. When all the conflicts have been solved, the SIA provides the sketch interpretation to the user, interacting with the Interface Agent.

In case symbols are decomposed into strokes, the SIA looks for conflicts by checking if there are symbols that share one or more strokes. Conflicts may take place either because a stroke is classified as two different shapes (for example, as a line and as an arc) due to the sketch inaccuracy, or because the same stroke, although correctly classified, is used by two SRAs to recognize two different symbols. By using the feedback of the recognized symbols and their accuracy, the SIA solves the conflicts starting by those where it is almost easy to identify the right interpretation. In this way, the SIA is able to free some symbols by the conflicts they were involved in. These symbols are used for solving other conflicts, and the process goes on until there are no more conflicts left

Finally, the SIA selects and communicates to the SRAs the active HDSRs that can be pruned. Heuristics are useful to avoid deleting symbols considered as mistaken due to the incompleteness of the input sketch. As an example, pruning could be applied to symbols without feedback involved in conflicts with symbols having feedback and to recognizing symbols whose constituent strokes all belong to another symbol with more positive feedback, and so on. The choice of the heuristics to be applied also depends from the domain language. It is important to select accurately symbols to be pruned because if the system decides to prune a symbol but it is not a mistaken recognized symbol then the symbol will not further considered for the next sketch interpretation requests. On the other hand, if symbols are correctly pruned, the system will be able to quickly reply to the user sketch interpretation requests.

AgentSketch has been implemented on top of the Jade agent-based platform using Java 1.5.

The Interface Agent implemented in AgentSketch provides a stub for an on-line editor that takes advantage of Java Swing components and Satin .

Agents in AgentSketch communicate by exchanging messages encoded in FIPA-ACL messages , a communication language natively supported by Jade. Instead, the content language of the message is not fixed by the FIPA-ACL specification, and may be chosen by the MAS developer. We express it in XML according to a set of XML-Schemas that we have defined. The messages exchanged within AgentSketch have various purposes, such as communicating the availability of new stroke classifications, requesting/providing feedbacks, requesting/providing the sketch interpretation, and so on. Jade offers the means for sending and receiving messages, also to and from remote computers, in a way that is transparent to the agent developer: for each agent, it maintains a private queue of incoming ACL messages that our agents access in a blocking mode.

The HDSRs included in AgentSketch have been implemented, following the approach presented in , as a set of Jess rules able to recognize the symbols belonging to the UML Use Case language. In our future work we want to exploit Web Services Technology to implement replaceable, independent, self-consistent HDSRs. This would allow to reach a complete decoupling between agents and HDSR, decoupling which is not fully achieved in our current architecture. In fact, by publishing their functionalities as Web Services, HDSRs might be implemented by anyone and in any language, might be physically stored anywhere, might run on any platform, and might be replaced with a minimal effort. In order to access them, agents should only know their physical address and the specification of the offered services. These services would consist of operations that given a set of stroke classifications, return a recognized symbol, if any.

In our current implementation of AgentSketch, each SRA checks symbol relations for feedback exchange purposes. In particular exploiting the attributes of symbols recognized by HDSRs, SRAs are able to check a set of relations between symbols (or symbol's portions/points) such as *near*, *intersect*, *touch*, *parallel*, and so on.

3. Experimental evaluation

An experiment was designed to evaluate the effectiveness of the recognition system for use case diagrams domain .

Such diagrams are composed of *use cases* (represented by ovals), *actors* (represented by stick figures), and connectors among them. There is only one relationship that may occur between *actors* and *use cases*; it is visualized as a solid line, named *communication* link, and means that an *actor* participates to a *use case*. Instead, four types of relationships between *use cases* are supported by UML: *communication* links, *inclusion* (visualized as a dashed arrow from the including to the included use case, with label «include»), *extension* (a dashed arrow from the extending to the extended *use case*, with label «extend»), and *generalization* (a solid line ending in a hollow triangle drawn from the specialized to the more general *use case*). The only relationship that may hold among *actors* is generalization.

We recruited twenty subjects with basic knowledge of use case diagrams, to which we briefly introduced AgentSketch, included the multi-stroke symbol recognition capabilities.

When they felt comfortable with the system we asked them to draw two diagrams, containing all use case symbols, representing real software systems. The two diagrams were formed by 16 and 23 symbols, respectively. Fig. 2 shows one of the diagrams sketched by the subjects with AgentSketch.

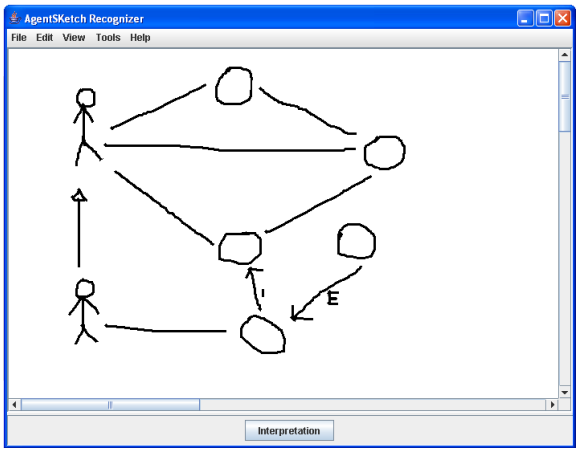

Figure 2. A use case diagram sketched by a subject with AgentSketch

To analyze the importance of the contextual information for the interpretation of sketched symbols, we compared the results obtained by the context-based recognizer, indicated with CR, with those obtained deactivating the SIA contextual reasoning module, indicated with BR (baseline recognizer). In the BR approach the conflicts are solved by considering the shape's accuracy only. Table 1 contains some statistics on the recognition performances of both systems. In particular, for each domain symbol we reported: the number of instances drawn by the subjects, those correctly recognized by BR and CR (in percentages) and the percentage of instances not recognized by any HDSR (%US).

Table 1. Recognition statistics by symbol

	#Instances	%correct BR	%correct CR	%U.S
Actor	100	73	84	0
Use Case	260	79.23	92.69	7.31
Comm.	280	92.50	97.86	1.43
Include	60	61.67	66.67	11.67
Extend	40	60	67.5	10
Generalize	40	65	87.5	7.5
Total	**780**	**80.13**	**89.87**	**4.74**

% correct BR = symbols correctly interpreted without using contextual information

% correct CR = percentage of symbol instances correctly interpreted using contextual information

% US = percentage of symbol instances unrecognized

The results show that the context-based conflict resolution technique implemented by the SIA considerably improves the overall precision in the recognition. On average, the baseline system correctly identified 80% of the symbols, while CR correctly identified 90%.

4. Related Work

Very few approaches for the recognition of freehand drawings are based on agent technology. When we compare our proposal with those using the agent technology, we find that the main differences lie in the intended usage domain of the system, which is very specific for all the implemented systems

QuickSet is a suite of agents for multimodal human-computer communication . Underneath the QuickSet suite of agents lies a distributed, blackboard-based, multi-agent architecture. The gesture recognition agent recognizes gestures from strokes drawn on the map. Along with the coordinate values, each stroke from the user interface also provides contextual information about objects touched or encircled by the stroke. Recognition results are an *n*-best list of interpretations and an associated probability estimate for each interpretation. This list is then passed to the multimodal integrator that accepts typed feature structures from both the gesture and the parser agents (that interpret natural language sentences), and unifies them.

In , Mackenzie and Alechina propose an agent-based technique for the classification and understanding of child-like sketches of animals, using a live pen-based input device. Once the segmentation stage is completed, an agent-based search of the resultant data begins in an attempt to isolate recognizable high-level features within the sketch. Newly recognized strokes are inserted into an "arena" (a sort of blackboard shared by all the agents, where agents can move), and every single agent is in charge of recognizing a specific high-level feature. Agents wander around the arena looking for strokes that they can use for recognizing a symbol. The mobile agents gradually become more and more restless over time if they are unsuccessful in their task. The more restless an agent is, the less reputable it is considered by other agents.

In Juchmes et al. describe EsQUIsE, an interactive tool for free-hand sketches to support early architectural design. The EsQUIsE environment uses pen computer technologies featuring the "virtual blank sheet". Lines drawn on the screen are captured and interpreted in real time thanks to the activity and collaboration of different types of agents. The system also involves agents that recognize individual chars, and a dictionary agent. When many different interpretations of the drawing are possible, the agents must work together to propose a pertinent interpretation of the sketch. In Azar *et al.* extend the previous multi-agent architecture with the possibility of interpreting vocal information also. The graphical inputs are interpreted by either rule-based agents or model-based agents, while the spoken inputs are interpreted by model-based vocal agents.

With respect to these systems, our proposal aims at creating a general sketch recognition system that does not rely on any assumption on the drawing style, and is not tailored to any specific domain.

4. Conclusions

In this paper we have presented AgentSketch, an agent-based system for interpreting hand-sketched symbols in a context-driven fashion. The recognition process is supported by intelligent agents (SRAs) that manage the activity of hand-drawn symbol recognizers, and coordinate themselves in order to provide efficient and precise interpretations of the sketch to the user. We have performed a first experimental evaluation of the system on the use case diagrams domain, and the results have indicated that the use of context to disambiguate symbol significantly reduced recognition error over a baseline system that did not consider contextual information.

References

[1] A. Apte, V. Vo, and T.D. Kimura, "Recognizing Multistroke Geometric Shapes: An Experimental Evaluation", in Proc. of *User Interface and Software Technology*, 1993, ACM Press, pp. 121-128.

[2] S. Azar, L. Couvreury, V. Delfosse, B. Jaspartz and C. Boulanger, "An Agent-Based Multimodal Interface for Sketch Interpretation", *in Proc. of International Workshop on Multimedia Signal Processing*, British Columbia, Canada.

[3] F. Bellifemine, A. Poggi, and G. Rimassa. "Developing Multi-Agent Systems with JADE", *in 7th International Workshop Agent Theories Architectures and Languages*, Lecture Notes in Computer Science, Springer, 2000, vol. 1986, pp. 89-103.

[4] P. R. Cohen, M. Johnston, D. McGee, I. Smith, J. Pittman, L. Chen, and J. Clow, "Multimodal Interaction for Distributed Interactive Simulation", *in Proc.of Innovative Applications of Artificial Intelligence Conference*, 1997, AAAI Press, pp. 978-985.

[5] FIPA ORG, FIPA ACL Message Structure Specification, Document no. SC00061G, 2002.

[6] L. Gennari, L.B. Kara, and T.F. Stahovich, "Combining Geometry and Domain Knowledge to Interpret Hand-Drawn Diagrams", *Computers & Graphics,* 29(4), 2005, pp. 547-562.

[7] T. Hammond and R. Davis, "LADDER, A Sketching Language for User Interface Developers", *Computers & Graphics*, 29(4), 2005, pp. 518-532.

[8] J.I. Hong, and J.A. Landay, "SATIN: A Toolkit for Informal Ink-based Applications", in *Proc. of User Interface and Software Technology* (*UIST'00*), 2000, ACM Press, pp. 63-72.

[9] R. Juchmes, P. Leclercq, and S. Azar, "A Freehand-Sketch Environment for Architectural Design Supported by a Multi-Agent System", *Computers & Graphics,* 29(6), 2005, pp. 905-915.

[10] B. Krishnapuram, C. Bishop, and M. Szummer, "Generative Bayesian Models for Shape Recognition", in *Proc.of 10th International Workshop on Frontiers in Handwriting Recognition* (*IWFHR-9*), 2004, IEEE CS Press, pp. 20-25.

[11] M. Luck, P. McBurney, O. Shehory, S. Willmott, and the AgentLink Community, "Agent Technology: Computing as Interaction – A Roadmap for Agent-Based Computing", *AgentLink III*, 2005.

[12] G. Mackenzie and N. Alechina, "Classifying Sketches of Animals using an Agent-Based System", *in Proc.of 10th International Conference on Computer Analysis of Images and Patterns*, Springer, Lecture Notes in Computer Science, vol. 2756 , 2003, pp. 521-529

[13] Object Management Group. UML Specification ver. 2.0. http://www.omg.org/technology/documents/formal/uml.htm, 2005.