A Novel Recognition Approach for Sketch-Based Interfaces

Danilo Avola, Andrea Del Buono, Giorgio Gianforme, and Stefano Paolozzi

CSKLab National Research Center, Department of Advanced Research
Via Savoia 84, 00198 Rome, Italy
{danilo.avola,andrea.delbuono}@csklab.it
http://www.csklab.it/index.html.en
University of "Rome 3", Department of Computer Science and Automation
Viale della Vasca Navale 79, 00146 Rome, Italy
{gianforme,paolozzi}@dia.uniroma3.it
http://web.dia.uniroma3.it/

Abstract. Multimodal interfaces can be profitably used to support the more and more complex applications and services which support human activities in everyday life. In particular, sketch-based interfaces offer users an effortless and powerful communication way to represent concepts and/or commands on different devices. Developing a sketch-based interface for a specific application or service is a time-consuming operation that requires the re-engineering and/or the re-designing of the whole recognizer framework. This paper describes a definitive framework that allows users to define each kind of sketch-based interface, using freehand drawing only. The definition of the interface and its recognition process are performed using our developed Sketch Modeling Language (SketchML).

Keywords: Sketch-based interfaces, sketch-based interaction, sketch recognition, multi-domain, vectorization, segmentation, XML, SVG.

1 Introduction

Multimodal interfaces allow users to interact naturally with any desktop or mobile devices using multiple different modalities (e.g. sketch, gesture, speech, gaze). This kind of interaction provides a powerful tool that allows users to manage the more and more complex applications and services surrounding their activities in everyday life. In particular, sketch-based interfaces enable users to express concepts, to provide commands and to represent ideas in an immediate, intuitive and simple way.

Unlike the other modalities, sketch-based interaction has two advantageous features. The first one is the "robustness" respect outdoor and indoor environments. This means that both behavior and error level of other interfaces (e.g. speech-based, eye-based) are strongly tied to the specific environment in which these interfaces are used. In fact, several factors (e.g. noise, amount of light) can

P. Foggia, C. Sansone, and M. Vento (Eds.): ICIAP 2009, LNCS 5716, pp. 1015–1024, 2009.

influence the interpretation process of these interfaces. The second one is the quick "customization" of the concepts and/or commands and/or ideas used to manage a specific application or service. In fact, it is always possible to create a specific set of graphical symbols in order to manage several applications and/or services according to both specific graphical standards and user needs.

The main effort in building sketch-based interfaces regards the implementation of the recognizer framework able to distinguish every symbol allowed in the system (the set of those symbols is called application domain or library). The application domains are potentially unbounded, since they just depend on the customization. Developing a sketch-based interface (that is a specific set of symbols) for a specific application and/or service is a time-consuming operation that requires the re-engineering and/or the re-designing of the whole recognizer framework.

This paper describes a definitive framework that allows users to define each kind of sketch-based interface, using freehand drawing only. Interface definition and its recognition process are performed using our developed Sketch Modeling Language (SketchML). Compared to the actual adopted methodologies in sketch recognition, our developed approach (and related prototypes) has three advantageous main aspects that jointly put our results in the vanguard inside the sketch-based interfaces area. The first one regards the possibility to define each kind of application domain. Actual frameworks are focused recognize only a specific set of symbols. They can also recognize other kinds of application domains but this step, differently from our approach, involves the management of the recognition logic inside the framework itself. The second one regards the possibility to quickly define each kind of application domain simply using free hand-drawing. In fact, there are several frameworks based on multi-domain concept, but in order to build a new application domain it is necessary an expert user with a deep knowledge about standards and/or vectorial applications and/or programming languages and so on. The last one concerns the interface definition and its recognition process which are performed by using SketchML, a language based on XML (eXtended Markup Language) standard. This consolidated W3C (World Wide Web Consortium) standard ensures both compatibility on different devices and cooperation between different applications/services.

The paper is structured as follows. Section 2 proposes some remarkable related works in multi-domain sketch-based interfaces. Section 3 introduces the novel approach and the related developed framework. Section 4 shows experimental results on a wide application domain. Finally, Section 5 concludes the paper.

2 Related Works

As mentioned in the previous section, our approach has three advantages main aspects respect the actual methodologies (define each kind of library without manage the framework, use sketch to build any libraries, represent sketch and its recognition by SketchML). However, our results have been assisted by the study of some remarkable works about the multi-domain sketch recognition frameworks. For example, an interesting methodology is presented in [1]. In this work the authors show a framework that allows users to draw shapes in natural way. In particular, the framework is able to recognize a variety of defaults domains by exploiting several consolidate techniques able to catch information about: geometric, features, temporal, contextual, multi-modal and domain. A different approach is provided in [2]. In this work the authors present a novel form of dynamically constructed Bayesian network which uses a sophisticated recognition engine able to integrate information and domain knowledge to improve recognition accuracy across a variety of domains. Another interesting work is shown in [3], in this work the authors present a remarkable agent-based framework for sketched symbol interpretation that heavily exploits contextual information for ambiguity resolution. In particular, in this work the agents manage both the activity of low-level hand-drawn symbol recognizers and contextual information to obtain an efficient and precise interpretation of sketches. Another approach, features based, is shown in [4]. In this work the authors present an advanced heuristic based framework that offers the user more freedom for: free-style sketching, grouping the strokes, and reducing the complexity of recognition activity. Unlike previous approaches the authors in [5] introduce a primitive-based engine. Their prototype is designed to infer designers' intention and interprets the input sketch into more exact geometric primitives: straight lines, poly-lines, circles, circular arcs, and so on. An important work, tied to our approach, is explained in [6]. The authors introduce LADDER, the first language to describe how sketched diagrams in a domain are drawn, displayed, and edited. In this work the difficulties about a suitable choice of a set of predefined entities that is broad enough to support a wide range of domains are faced. This language consists of predefined shapes, constraints, editing behaviors, and display methods, as well as a syntax for specifying a domain description sketch grammar and extending the language, ensuring that shapes and shape groups from many domains can be described. The just mentioned work has been a real source of inspiration. Our proposed SketchML has several common points with LADDER language, but SketchML is more general and it can be used to represent any application domain. A different approach is shown in [7], where a framework for recognition of composite graphic objects, topological spatial relationships of their components play an important role is given. In this work the authors introduce the ternary relationship, which is a complement to the binary relationship, to describe composite graphic objects. Another remarkable approach is shown in [9] where the authors describe a statistical framework based on dynamic Bayesian networks that explicitly models the fact that objects can be drawn interspersed. We conclude this section showing an approach, among those available, based on HMMs (Hidden Markov Models). In fact, new interesting tendency on sketch recognition approaches is to use HMM to overcome several critical duties, such as: customization, shape learning, and so on. An innovative approach is proposed in [8] where the authors show that it is possible to view sketching as an interactive process in order to use HMM for sketch recognition process. All the described approaches depend on set of symbols and/or primitives that are tied to specific application domains. Our novel approach overcomes this kind of problems by using real general primitives.

3 Approach and Developed Framework

The developed framework provides two different environments. The first, shown in Figure 1-a, is the Library Editor Environment (LEE), where the user can define, by freehand drawing, the libraries. The second, shown in Figure 1-b, is the Sketch Recognition Environment (SRE), where the user can perform the recognition activity, on a drawing schema, after having loaded a previously created library. Both environments allow users to perform several useful activities, such as: deletions, restyling, and so on.

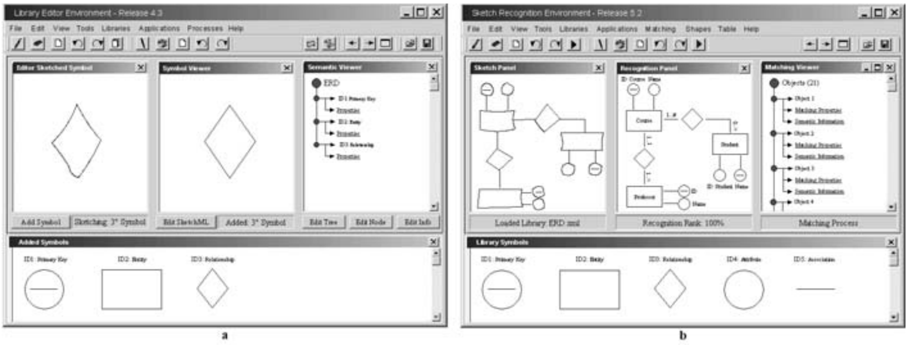


Fig. 1. a) Library Editor Environment b) Sketch Recognition Environment

In order to explain both the approach and the related framework functionalities a well known application domain, Entity Relationship Diagram (ERD), is introduced. In Figure 1-a the user has drawn the third symbol regarding the ERD domain, and the LEE has built the related vectorial representation. In Figure 1-b the user has performed a simple ERD schema, and the SRE has built the related vectorial schema. Both LEE and SRE work in off-line way, that is the user decides when the environments have to perform the recognition (and related vectorization) activity.

The approach and the algorithms designed to obtain the vectorial symbol/schema from the sketched symbol/schema is the main aspect of our paper. In both environments our XML-based developed language (SketchML) is used to describe the vectorial representations.

The SRE exploits the knowledge of the symbols that make up the loaded library to improve the recognition process. In order to explain this concept in Figure 2 an example based on the five symbols belonging to the ERD library is shown. In Figure 2-a the user has correctly sketched an ERD instance. More precisely, the user has sketched the following sequence of ERD symbols: three primary keys, three entities, two relationships, three attributes and ten associations (twenty one objects). In Figure 2-b, all the drawn objects have been recognized. Indeed, the situation is more complex than the just mentioned one. In fact, taking into account the symbols belonging to the loaded library, it is possible to observe that the sketched symbol of association (shown in Figure 2-a

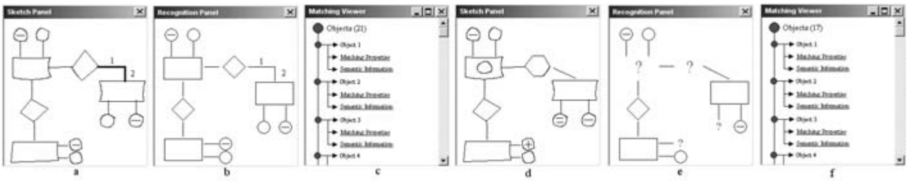


Fig. 2. Correct/Incorrect ERD Instance

by labels 1 and 2) is not a library symbol. It is recognized as an association symbol because it results as the composition of two symbols of the ERD library with a particular constraint (i.e. the two association symbols with an angle of ninety degrees). In Figure 2-c is shown the matching process of each symbol (explained in the relative section). This process allows the framework to map every object sketched by the user (in SRE) with a symbols belonging to the related library (defined in LEE).

Unlike the previous example, in Figure 2-d the user has incorrectly sketched an ERD instance. In fact, as it is possible to observe in Figure 2-e, seventeen objects have been recognized and four objects have not been recognized. In Figure 2-f is detailed each one of the matching process between the objects drawn by the user and the loaded library. This example highlights that four objects have not satisfied the matching requirements (in fact the matched objects are seventeen), this means that these four objects have not matched a respective symbol in the loaded library. The next two subsections describe the main aspects (approaches and algorithms) of the LEE and SRE.

3.1 Library Editor Environment

In order to explain the subsection content, it is necessary to introduce the following three definitions: stroke (set of pixels obtained during the pen action: pen down, pen drawing, pen up), sub-stroke (subpart of a stroke), generalized stroke (two, or more, joined sub-strokes coming from different strokes).

The LEE activity is performed by four sequential modules: the first module (First Layer Segmentation Algorithm, FLSA) is used to obtain, from the strokes drawn by user, two kind of objects: closed regions and poly-lines. The second module (Second Layer Segmentation Algorithm, SLSA) is used to obtain the set of primitives (ovals and lines) that make up each obtained object. The third module (Constraints Establishment, CE) is used to evaluate the relationships between the found primitives. The fourth module (SketchML Descriptor) is used "to translate" all the found information in a XML/SVG structure. In this subsection each module is detailed.

The aim of FLSA (first module) is to provide two different sets of objects: closed regions and poly-lines. This concept is not so simple as it might appear. In fact, there is not always a unique interpretation about two sketched strokes. For example, the strokes (S_1 and S_2) drawn in Figure 3-a-a can be interpreted as two overlapped closed regions (A_1 and A_2, Figure 3-a-b), otherwise the strokes can

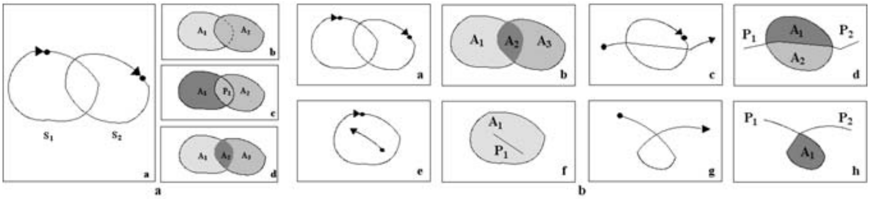


Fig. 3. a) Ambiguity Examples. b) Algorithm Interpretation.

be interpreted as two closed-regions (A_1 and A_2) and a poly-line (P_1), Figure 3-a-c. In another interpretation, Figure 3-a-d, the two strokes can be interpreted as three closed regions (A_1, A_2, A_3). Moreover, there are other available interpretations. This kind of issues, defined as ambiguity problems, is a crucial point of every sketch recognition system. To overcome the ambiguity problems, the developed FLSA works taking into account the following two definitions: closed region (the smallest area confined by a set of strokes and/or sub-Strokes and/or generalized strokes), poly-line (the smallest stroke or sub-Stroke). Following the introduced definitions every sketch can be interpreted in a unique way.

In order to highlight the algorithm approach in Figure 3-b some examples of developed algorithm interpretation are given. The two strokes drawn in Figure 3-b-a are interpreted, as shown in Figure 3-b-b, as three closed regions (A_1, A_2 and A_3). The two strokes drawn in Figure 3-b-c are interpreted, as shown in Figure 3-b-d, as two closed regions and two poly-lines (A_1, A_2 and P_1, P_1). The two strokes drawn in Figure 3-b-e are interpreted, as shown in Figure 3-b-d, as a closed region and a poly-line (A_1, P_1). Finally, the stroke drawn in Figure 3-b-g is interpreted, as shown in Figure 3-b-h, as a closed region and two poly-lines (A_1, P_1 and P_2).

In order to accomplish as just mentioned the FLSA works according to the following sequential two main steps. During the first step every stroke, without intersection points, is analyzed to check if it is closed or not. During the second step every stroke, with intersection points, is analyzed to detect every intersection point which is used to find the maximum number of disjoined areas. This last step is performed by an exhaustive sub-algorithm which, from every intersection point, searches every possible closed path. During the process the sub-algorithm considers, step by step, only the smallest areas that do not overlap (entirely or in part) other found areas.

Starting from the set of objects (closed regions and poly-lines) identified by the previous algorithm, the aim of the SLSA (second module) is to recognize, on each object, the set of primitives that makes it up. In our context, the term primitive means only two specific geometrical primitives: line and oval (and as particular arc and circle). The concept behind the developed algorithm is that every "segment" of a shape can be approximated by the mentioned geometrical primitives. In order to explain the just introduced concept in Figure 4 an example

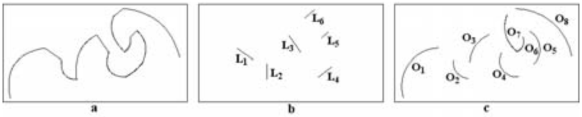


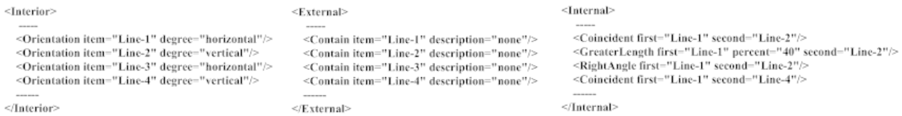
Fig. 4. Searching: Primitives

is shown. Deliberately the example regards a complex poly-line, our goal is to show that the proposed approach is very general, and it can be used on every kind of user sketch.

As shown in Figure 4-b, the algorithm first detects the lines ($L_1..L_6$), the remaining searched primitives are the ovals or, as shown in this example (Figure 4-c), the sub-parts (arcs) of different ovals ($O_1..O_8$). In order to accomplish this task on every closed-region and/or poly-line the algorithm works according to the following two main steps. During the first step (line detection) on every object the longest set points having a linearity property are searched. The term linearity property is used to indicate a contiguous set of points (having a prefixed minimum length, called L_{fix}) that, independently from the orientation, can be considered belonging to a same prefixed enclosing rectangle. During the second step (oval/arc, detection) every set of points discarded by the previous step is an oval (or arc) candidate. For this reason a curvature property is verified. The term curvature property is used to indicate a contiguous set of points in which, independently from the orientation, the linearity property is true but only on contiguous sub-set points having a length lower than L_{fix}. In order to obtain a suitable collocation of the found set of points as line and/or oval (or arc), the two steps are cooperatively applied several times. At the end of the developed algorithm it is possible to perform the vectorization step by replacing the set of points representing each detected primitive with the following basic SVG (Scalable Vector Graphics, which is an XML based language) shapes: primitive line with line SVG basic shape, primitive oval with ellipse SVG basic shape, primitive part of oval with path SVG basic shape.

Starting from the vectorial set of primitives (SVG basic shapes: lines, ellipses, and paths) that make up every object, the aim of the CE (third module) is to detect the spatial relationships between them. Our recognizer engine takes into account the following constraints classification: interiors (constraints evaluated on each single vectorial primitive, e.g. orientation, closure), internals (constraints evaluated between primitives of a same object, e.g. coincident, intersect), and externals (constraints evaluated between primitives of different objects, e.g. contain, vertical alignment). Every constraint has several attributes in order to specify the information needed to define the related spatial relationship.

When the recognition process reaches SketchML Descriptor (fourth module) every hard task has been performed. In fact, this module has only to include the vectorial information obtained previously about the SVG basic shapes (in SLSA), with the new information (suitably "translated" in XML) related to the found



```
<Interior>
  -----
  <Orientation item="Line-1" degree="horizontal"/>
  <Orientation item="Line-2" degree="vertical"/>
  <Orientation item="Line-3" degree="horizontal"/>
  <Orientation item="Line-4" degree="vertical"/>
  ------
</Interior>
```

```
<External>
  -----
  <Contain item="Line-1" description="none"/>
  <Contain item="Line-2" description="none"/>
  <Contain item="Line-3" description="none"/>
  <Contain item="Line-4" description="none"/>
  ------
</External>
```

```
<Internal>
  -----
  <Coincident first="Line-1" second="Line-2"/>
  <GreaterLength first="Line-1" percent="40" second="Line-2"/>
  <RightAngle first="Line-1" second="Line-2"/>
  <Coincident first="Line-1" second="Line-4"/>
  ------
</Internal>
```

Fig. 5. Vectorial Square Represented By SketchML

constraints. This enriched XML makes up the main constructs of our developed SketchML. In Figure 5 is shown a SketchML representation of a vectorial square. The amount of information needed to represent a simple and single symbol is very large, for this reason in Figure 5 only some constraints are shown.

In particular, the example shows that the square is considered as composed from four lines (Line-1, Line-2, Line-3, Line-4). These lines have several interior, internal and external constrains. More precisely, the main interior constraints explain the orientation of the lines, there is not main external constraints (because there are not others objects) and there are several internal constraints that explain the relationships between the mentioned four lines (for example, the GreaterLength constraint applied respectively to the arguments Line-1 and Line-2, describes that the second line (Line-2) is greater than the first line (Line-1) by about 40 percent.

3.2 Sketch Recognition Environment

This environment is used to recognize one or more than one library symbols. In the first case (i.e. when a user expresses a command) the environment works exactly as the LEE. In the second case (i.e. when a user expresses a concept) it is necessary a previous elaboration (based on context and spatial relationships) to recognize, first of all, how many possible symbols the user has drawn. To accomplish this task all the intersection points (of the sketch performed by the user) are evaluated. After this step the system detects the intersection points that do not have "correspondence" with the intersection points used to build the library, in this way a first evaluation about the number of the sketched objects can be performed. In both cases the next step is the matching process to map the performed "candidate" sketched symbols with the symbols contained in the loaded library.

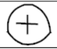
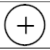
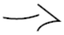
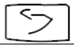
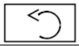
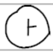
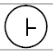
3.3 Matching Process

The final purpose of matching between symbols sketched by the user in the SRE and the symbols represented in a specific library (made up by LEE) is to choose the right element inside the library that provides the suitable interpretation of the sketched symbols. As mentioned, every symbol of the loaded library inside the framework is described through an XML/SVG file. Also the user's sketch has been represented using XML/SVG file that contains information comparable to the symbols of the library. The matching algorithm performs an exhaustive search of the particular patterns created by the user during sketch activity. Every SVG basic primitive (that is, Line, Ellipse and Path), coming from library or user's sketch, is represented by both a node and a set of relationships (the constraints) between the nodes. Moreover, two or more related basic symbols (for example the containment constraint) are represented by two macro-nodes between which exist a macro-relation (in this case, the *containment*). The matching approach starts from a generic node (of the sketch representation), it performs a matching action with every node of the same kind (such as: line, ellipse or path) presents in the symbol of the loaded library. Obviously, the process appears exponentially complex specially when the first nodes are examined. The process ends when all the nodes are examined. When this happens several possible configuration will be identified by the system. Each one will have a rank depending on the quality of the accomplished matching. Several configuration will have a very low rank, others will have a suitable ranking profitable to identify the right pattern (i.e. the right symbol).

4 Experimental Results

This section shows the results obtained on a wide application domain designed to meet the needs of heterogeneous environments. As shown in Table 1 the application domain is made up from 178 different symbols. In this context it has to be taken into account that several symbols (e.g. arrow) can have up to eight different orientations (North, East, South, West, Northeast, Southeast, Southwest, Northwest) each of which is considered a different symbol. For each symbol has been performed an average of 75 tracking for a total of 11.570 tests. The percentage of recognition of each symbol is almost total. Indeed, this percentage tends to be slightly lower according to the complexity of the symbol. In fact, for symbols made up of more than 7 strokes some constraints may be ambiguous and misinterpreted. In order to overcome this problem, it is possible to introduce in the framework a more complex personalization concept. Moreover, it is possible to consider a new class of constraints (i.e. control constraints) able to oversee ambiguous potentially situations. The framework has been also tested on more traditional domains (i.e. entity relationship diagram, data flow diagram, electrical schemes, and so on) providing always a total level of reliability.

Table 1. Experimental Application Domain

N° Symbols 178 (Examples)	N° Tracing	Related Vectorization	% of Recognition	% Misinterpretation	N° Symbols 178 (Examples)	N° Tracing	Related Vectorization	% of Recognition	% Misinterpretation
	72		100%	0%		100		95%	0%
	80		96%	0%		67		99%	0%
----------	----------	----------	----------	----------	Tot: 11.570	Averange: 75	Tot: 11.570	Averange: 98%	0.5%

5 Conclusions

The proposed approach provides a concrete framework able to build and to recognize every kind of sketched symbol/schema. The key point is that a user, without special knowledge, can create a personalized library simply sketching the desired symbols. The simple and versatile language used to represent the sketch activity (SketchML) allows the cooperation of the proposed framework with a large number of applications, devices and services. The framework can be used on 2D and 3D library (adding another category of constraints 3D oriented). Moreover, the proposed algorithms allow the user to perform sketches with an high inaccuracy level (without ambiguity problems and/or unrecognized events).

References

1. Hammond, T., et al.: Free-sketch Recognition: Putting the CHI in Sketching. In: Proc. of CHI 2008 Extended Abstracts on Human Factors in Computing Systems, CHI 2008, pp. 3027–3032. ACM Press, Florence (2008)
2. Alvarado, C., Davis, R.: Dynamically Constructed Bayes Nets for Multi-Domain Sketch Understanding. In: Proc. of ACM SIGGRAPH 2007 courses, SIGGRAPH 2007, p. 33. ACM Press, California (2007)
3. Casella, G., Deufemia, V., Mascardi, V., Costagliola, G., Martelli, M.: An Agent-Based Framework for Sketched Symbol Interpretation. Journal of Visual Languages and Computing 19(2), 225–257 (2008)
4. Sahoo, G., Singh, B.K.: A New Approach to Sketch Recognition Using Heuristic. JCSNS International Journal of Computer Science and Network Security 8(2), 102–108 (2008)
5. Shu-Xia, W., Man-Tun, G., Le-Hua, Q.: Freehand Sketching Interfaces: Early Processing for Sketch Recognition. In: Jacko, J.A. (ed.) HCI 2007. LNCS, vol. 4551, pp. 161–170. Springer, Heidelberg (2007)
6. Hammond, T., Davis, R.: LADDER: a Language to Describe Drawing, Display, and Editing in Sketch Recognition. In: Proceedings of SIGGRAPH 2006: ACM SIGGRAPH 2006 Courses, p. 27. ACM Press, New York (2006)
7. Peng, B., Liu, Y., Wenyin, L., Huang, G.: Sketch Recognition Based on Topological Spatial Relationship. In: Fred, A., Caelli, T.M., Duin, R.P.W., Campilho, A.C., de Ridder, D. (eds.) SSPR&SPR 2004. LNCS, vol. 3138, pp. 434–443. Springer, Heidelberg (2004)
8. Sezgin, T.M., Davis, R.: HMM-Based Efficient Sketch Recognition. In: Proceedings of the 10th International Conference on Intelligent User Interfaces, IUI 2005, pp. 281–283. ACM Press, San Diego (2005)
9. Sezgin, T.M., Davis, R.: Sketch Recognition in Interspersed Drawings Using Time-Based Graphical Models. In: Proceedings of Computer Graphics, vol. 32(5), pp. 500–510. Pergamon Press, Inc., Elmsford (2008)