Graphical Knowledge Management in Graphics Recognition Systems

Mathieu Delalandre[1], Eric Trupin[1], Jacques Labiche[1], Jean-Marc Ogier[2] ♣

[1] PSI Laboratory, University of Rouen, 76 821 Mont Saint Aignan, France
{first name, last name}@univ-rouen.fr
[2] L3i Laboratory, University of La Rochelle, 17042 La Rochelle, France
jean-marc.ogier@univ-lr.fr

Abstract. This paper deals with the problem of graphical knowledge management (formalization, modelling, representation and operationalization) in graphics recognition systems. We present here a "generic" formalism for graphical knowledge, allowing various modellings for a given graphical shape. We use a modelling library based on this formalism for the management of our graphical knowledge. The use of this library allows to request graphical knowledge databases, according to the processings' requirements on graphical primitives. Like this, this approach allows interoperability between processings, especially for their combination. We present a "short" system use-case of our approach to illustrate the interoperability between processings.

1 Introduction

This paper deals with the problem of graphical knowledge management in graphics recognition systems. This knowledge corresponds to graphical primitives used by systems during the recognition process. We present here a "generic" formalism for graphical knowledge. Indeed, this formalism allows various modellings of a given graphical shape. Based on this formalism, we have developed a modelling library for the representation and the operationalization of our graphical knowledge. We use this library in graphics recognition systems to request graphical knowledge databases, according to the processings' requirements on graphical primitives. Like this, this approach allows the interoperability between processings, especially for their combination. In the paper's follow-up, we present in section (2) an overview on graphical knowledge management in graphics recognition systems. In section (3), we present our approach for graphical knowledge management with our formalism, its representation and operationalization through our modelling library. In section (4), we present a "short" use-case of our approach with a graphics recognition system and its application. Finally, in section (5) we conclude and give some perspectives.

♣ The authors wish to thank Sébastien Perin and Mustapha Hamidou (Rouen University, France) for their contribution to this work.

2 Graphical Knowledge Management: Overview

Graphics recognition [19] is a stage of document image interpretation that is used for different purposes like: technical document interpretation [1], symbol recognition [10], handwriting recognition (especially Asian handwriting [18]), and so on. It is a well-known problem and several commercial applications exist [1]. A graphics recognition process can be decomposed into two parts [14]: the extraction part of graphical primitives, and the system part.

The system part uses various approaches in order to supervise the extraction process [4]. These approaches come from pattern recognition and artificial intelligence domains. This paper deals especially with the extraction step of graphical primitives [22] [5]. This one extracts graphical primitives from document images corresponding to graphical shapes of documents. It employs many methods in order to extract different primitive types from images. In a previous work [5], we have proposed a classification of these methods in some families (Fig. 1). The methods are based on skeletonization (a), contouring (b), tracking (c), run (d), region (e), mesh (f), object segmentation (g), connected component grouping (h).

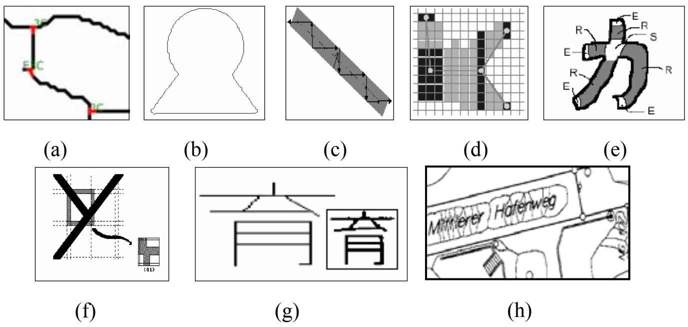


Fig. 1. (a) skeletonization (b) contouring (c) tracking (d) run (e) region (f) mesh (g) object segmentation (h) component grouping

We do not discuss in this paper about the presentation and the comparison of these methods[1], but about common graphical primitives extracted between these methods as we show in Table 1. These graphical primitives can be grouped in four primitive classes: pixel, vectorial (vector, arc, and curve), region (subset of connected pixels on image), and symbol (a symbolic label). In the same way, some methods can be used to extract different types of primitive [5]. For example, the skeletonization and contouring are often used with a polygonisation method to extract vectorial data (in the "*two steps*" vectorisation systems [15]). Also, the run decomposition methods can be used to extract the skeleton and contours [24], and in this way used with polygonisation method, and so on.

[1] It is not the purpose of this paper to do this, we report the reader to [1] [5] [10] and [22].

Table 1. Comparison of methods for graphical primitive extraction

Graphical primitives	Methods
Pixel	skeletonization (a), contouring (b), run (d)
Vectorial	skeletonization (a), contouring (b), tracking (c), run (d), object segmentation (g)
Region	Run (d), region (e), component grouping (h)
Symbol	*all*

So, a system can use different methods in order to extract some given graphical primitives. In a previous work [5], we have presented the drawbacks and advantages of all these methods (Fig. 1). So, their combination can help a system for the graphical primitive extraction. From our point of view, it is an important research perspective of graphics recognition. However, this perspective raises the problem of *graphical knowledge exchange* between the system's processings based on these methods.

In computer science systems "in the large" [20], the *knowledge* corresponds to semantic data (example, data: "37.5", semantic: "a temperature") with their exploitation processes (the system parts based on knowledge use). The use of these exploitation processes corresponds to the *operationalization* of systems' knowledge [4]. In these systems, the knowledge is used [20] in an internal way (in the algorithms) or in external way (outside of algorithms). The external knowledge is based on knowledge *representation* methods [16] like: representation languages, databases, and formats. In the internal and external cases, the knowledge used is based on a *formalism* [16]. Several formalisms exist[2] like: algebraic (list, matrix, number, and so on.), rule, graph, frame, and so on. Based on these formalisms, the systems use *modellings* of their knowledge. A modelling corresponds to a possible use of a given formalism. In the literature [16], we talk about knowledge *management* for the formalization, the modelling, the representation, and the operationalization of knowledge.

Different types of knowledge are used in graphics recognition systems [14]. This paper deals only with graphical knowledge [11]. This knowledge corresponds to graphical primitives used in systems. We resume on Table 2 formalisms commonly used for the graphical knowledge in some research systems, and standard formats of vector graphics.

Table 2. Formalisms of graphical knowledge

Systems	Formalisms	Formats	Formalisms
ADIK [12]	vectorial, rule, symbol	CGM [8]	vectorial, graph
ANON [1]	vectorial, rule, symbol	DXF [2]	vectorial, list
DMOS [3]	region, vectorial, rule	SVG [21]	vectorial, list
OOPSV [15]	vectorial, graph, symbol		
QGAR [9]	vectorial, graph		

[2] We don't present here these formalisms, and report the reader to [16] and [20].

From our point of view, the graphical knowledge in graphics recognition systems is based on two formalism levels (Table 2). A low level is used to describe the graphical primitives. It is based on general formalisms used in graphic file formats [11]: vectorial, and raster (for the region representation). A high level is used to structure these graphical primitives. Different formalisms are then used. Among them the most used are the lists and the graphs [17], the rule formalism is often used too. However this one is more adapted to recognition problem than modelling problem [1] [3] [12]. The graph and list formalisms correspond to structural descriptions of graphical primitives. Indeed, the graphical shapes of documents represent themselves, in a natural way, according to a structural description [1] [10] [19].

However, based on these structural formalisms, the graphics recognition systems use fixed modellings of their knowledge [3] [9] [12] [15]. The modellings are chosen according to the recognition approaches of these systems. Based on these fixed modelling, these systems can't deal with an adaptable combination of processings for the graphical primitive extraction. To solve "a part" of this problem, some systems perform their combinations through a low level formalism (image) [15], or a high level formalism (rule) [3].

In the following section (3), we present a "generic" formalism and a modelling library for the graphical knowledge management. This library allows to request a graphical knowledge databases, according to the processings' requirements on graphical primitives. Like this, this approach allows the interoperability between processings, especially for their combination.

3 Our Approach for Graphical Knowledge Management

We present here our approach for graphical knowledge management. We first present in subsection (3.1) our formalism. Next, in subsection (3.2), we present a modelling use-case of a given graphical shape. In subsection (3.3), we present the knowledge representation and operationalization through our modelling library.

3.1 Used Formalism

Our formalism is based on object-oriented concepts for knowledge formalization [13]. We have based our approach especially on works described in [23]. Our graphical knowledge (k_g) is represented (1) by a single graphical object (o). This graphical object is an instance (i) of a generic (and abstract) graphical object class (o_g) which is specialized in several graphical object classes ($\{1,.,u\}$) according to an inheritance (I) relationship. In this way, this representation exploits some important properties of inheritance [23], polymorphism and extensibility. The graphical objects are composed (2) of a set of data (D) and methods (M). These data (D) can be composed (2) of specific data (d_i), or other graphical objects (o_i) through a composition (or aggregation) relationship.

In our approach (as we have concluded in our overview of section (2)) we have decomposed the different graphical object classes, in an implicit way, into two formalism levels (3). The first one is a low level formalism for the description of graphical primitive (*p*). So, this description is based on vectorial and raster formalisms [11]. We have considered some standard graphical primitives like: *point*, *junction*, *line*, *arc*, *curve*, *region*, and *quadrilateral*. However, these standard graphical primitives can be easily extended thanks to the polymorphism and extensibility properties [23] of our approach (1). The second one is a high level formalism, based on list (*l*) and graph (*g*) formalisms, to structure the graphical objects corresponding to graphical primitives.

$$\exists k_g = \{o\} \quad o \xrightarrow{i} o_g \xleftarrow{I} \{o_{g_1},..,o_{g_u}\} \tag{1}$$

$$o = \{D, M\} \quad D = \{\{d_0,..,d_u\},\{o_0,..,o_v\}\} \quad M = \{P,\{r,w\}\} \tag{2}$$

$$o_{g_i} \in \{\{p_1,..,p_u\},\{l,g\}\} \tag{3}$$

The list formalism (*l*) defines (4) sets of ordered graphical object (*O*) and ordered attribute object (*A*). A given attribute object (a_i) describes a relationship (5) between two successive graphical objects (o_i) and (o_j) of the list. These attribute objects (*a*) are defined in the same way (6) (7) than the graphical objects (*o*) (1) (2). In the same way, these ones exploit the polymorphism and extensibility properties [23]. According to the list's looping (4), the (*A*) size may be of (*u*) or (*u-1*). We have considered some standard attributes like the *labelling*, *angle*, *length*, and so on.

$$l = \{O, A\} = \{\{o_0,..,o_u\},\{a_0,..,a_v\}\} \qquad v = ((u-1) \vee u) \tag{4}$$

$$\forall\, a_i \quad \exists \{o_i, o_j\} \quad f(o_i, o_j) = a_i \tag{5}$$

$$a \xrightarrow{i} a_g \xleftarrow{I} \{a_{g_1},..,a_{g_u}\} \tag{6}$$

$$a = \{D, M\} \quad D = \{\{d_0,..,d_u\},\{a_0,..,a_v\}\} \quad M = \{P,\{r,w\}\} \tag{7}$$

The graph formalism (*g*) defines (8) sets of graphical object (*O*) and edge object (*E*). A given edge object (e_q) describes a directed (or undirected) relationship (9) between any graphical objects (o_i) and (o_j). This relationship is defined (9) according to a given attribute object (a_q).

$$g = \{O, E\} = \{\{o_0,..,o_u\},\{e_0,..,e_v\}\} \tag{8}$$

$$\forall\ e_q = \{\{i, j\}, a_q\} \quad \exists\ \{o_i, o_j\} \quad f(o_i, o_j) = a_q \qquad \textbf{(9)}$$

Through these object-oriented concepts for knowledge formalization, our graphical knowledge (k_g) (1) is structured according to a *hierarchical relational graph* [17]. Indeed, our graphical objects (o) (1) included into the list (l) (4) and graph (g) (8) objects can be other list (l) and graph (g) objects. The list (l) objects are used here to reduce the size of graph (g) objects. Indeed, the list formalism can be considered as graph formalism [17], this one is commonly used in graph representation models of document shapes [24].

3.2 Modelling Use-Case

Our formalism (subsection (3.1)) allows various modellings of a given graphical shape. In order to illustrate this "generic" aspect, we present here a modelling use-case of linked-squares (Fig. 2 (a)).

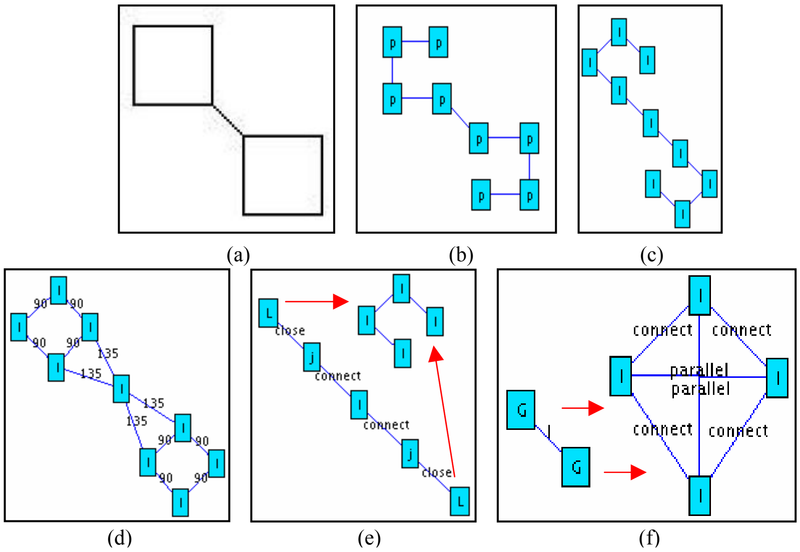


Fig. 2. (a) linked-squares (b) point list (c) line list
(d) line graph (e) hierarchical lists (f) hierarchical graphs

The Fig. 2 (b), (c), and (d) present three no-hierarchical modellings of linked-squares. The (b) (c) modellings are based on the list formalism, for the point (p) (b) and line (l) (c) objects. Indeed, it is possible to describe the linked-squares according to successive points or lines. The (d) modelling is based on graph formalism for the line (l) objects. In this graph, the connected lines (l) are linked by angular attributes.

The Fig. 2 (e) and (f) present two hierarchical modellings of linked-squares. The (e) modelling is based on list formalism for the line (*l*), junction (*j*), and (*L*) objects. In this modelling, the (*L*) objects represent sub-lists of line (*l*) object. Each sub-list corresponds to a square object. The (*L*), (*j*), and (*l*) objects are linked by the labelling attributes (*close*) and (*connect*). The (f) modelling is based on graph formalism for the line (*l*) and (*G*) object. In this modelling, the (*G*) objects represent sub-graphs of line (*l*) objects. Each sub-graph corresponds to a square object. In these sub-graphs, the line (*l*) objects are linked by labelling attributes (*connect*) and (*parallel*). The (*l*) and (*G*) objects are linked by an attribute corresponding to a graphical primitive (*l*).

Like this, the Fig. 2 presents some of possible modellings (b-f) of linked-squares (a). Based on our formalism (subsection (3.1)), it is still possible to define several modellings. From our point of view, it doesn't exist a best modelling to describe a given graphical shape. Indeed, an adopted modelling by a graphics recognition system depends of its process aims [14]. Then, it is important to allow the exchange of "similar" graphical knowledge between graphics recognition systems, in spite of differences between adopted modellings.

3.3 Graphical Knowledge Representation and Operationalization

We present in this subsection the representation and the operationalization of our graphical knowledge through our *Graphical Object Modelling Library*[3].

In our formalism (subsection (3.1)) each graphical and attribute object is composed of a set of method (*M*) (2) (7). This set of method is composed of process methods (*P*) on object's data, and read (*r*) write (*w*) methods (2) (7). In this way, each object supports its outsourcing. The Fig. 3 (a) gives an example of representation in XML used in our library of point object. The outsourcing properties of objects can be then used by other objects, like the (*l*) (4) and (*g*) (8) objects, or any other graphical or attribute objects using a composition relationship (2) (7). The Fig. 3 (b) gives an outsourcing example of point object used through a composition relationship into the line object.

```
<OPoint x= "10" y= "10" />
```

(a)

```
<OLine length="10" direction="0">
      <OPoint x="10" y="10" />
      <OPoint x="20" y="10" />
</OLine>
```

(b)

Fig. 3. XML representation: point (a) line (b)

We use our library in the graphics recognition processings. Like this, our library allows the graphical knowledge operationalization into the processings for, the graphical primitive management, and their read/write into graphical knowledge databases represented in XML (Fig. 3).

[3] GOMLib, available on http://site.voila.fr/mdhws/

The aim of our approach is to allow the interoperability between processings. We have developed a request based approach, in order to extract graphical knowledge from XML databases according to the processings' requirements on the graphical primitives. This approach exploits request methods, based on list or/and graph search algorithms. So, these requests are "content based" like the FLoWeR[4] requests. Indeed, our requests do not allow to structure search like sub-lists or/and sub-graphs.

The Fig. 4 presents our requests based approach through an example. In this example, a processing (*Processing*) performs a read request method (R_r) on a graphical knowledge database (k_g). This request uses a set of "content constraints" corresponding to the request (r_r): list of point (l_p) and size ($s_{\geq 2}$). So, we can translate this request into natural language like this: "**for** graphical knowledge (k_g) **return** lists **where** a list is only composed of point object (l_p) **and** the list's size is upper than one point ($s_{\geq 2}$)". Then, a read graphical object (o_r) is extracted corresponding to request's result. Following the execution of processing (*Processing*), an object to write (o_w) is obtained. The processing (*Processing*) performs then a write request method (R_w) in order to update the graphical knowledge database (k_g) with this result object (o_w). During the (R_w) execution, (r_r) is used like trace to locate the objects to update, in this example the line list objects (l_l) update the point list object (l_p).

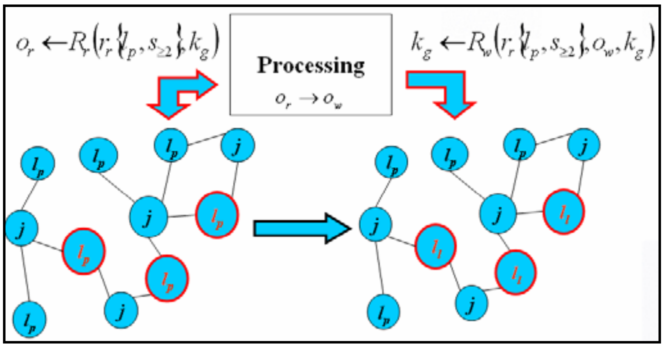


Fig. 4. example of request process on a graphical knowledge database

4 System Use-Case

In order to illustrate our approach for graphical knowledge management (section (3)), we present here a "short" system use-case of graphics recognition. The Fig. 5 (a-high) gives a network's part extracted from an utility map [6]. For our graphics recognition system, we have developed the well-known contouring/skeletonisation approach [1]. We don't discuss here about the processing abilities[5], but about the interoperability between processings through the graphical knowledge database.

---

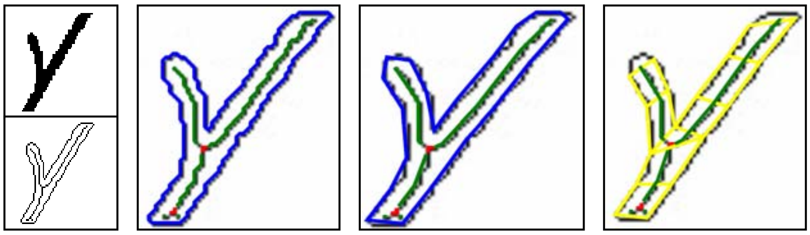
[4] For Let Where Return

[5] It is not the purpose of this paper to do this, we report the reader to [6].

In a first step (Fig. 5), our system performs a chaining processing (b) on the resulting image of skeletonisation/contouring processing (a-low). So, our graphical knowledge database (k_g) is updated (Table 3) from a raster object (r) to a graph (g) object composed of junction (j) and point list (l_p) objects. In a second step, our system performs a polygonisation processing (Fig. 5 (c)). A request on point list (l_p) objects is then used to extract these (l_p) objects form graph (g) object. These (l_p) objects are then updated (Table 3) in (k_g) by line list (l_l) objects. In the final step, our system performs a contour matching processing (Fig. 5 (d)). A request on closed line list (l_l) objects is then used to extract these (l_l) objects form graph (g) object. These (l_l) objects are updated (Table 3) in (k_g) by quadrilateral list (l_q) objects. The result graphical knowledge database (k_g) is then composed of (g), (j), (l_l), and (l_q) objects.

Table 3. update of graphical knowledge database for processings interoperability

skeletonisation/contouring	**chaining**	**polygonisation**	**matching**
$k_g = \{r\}$	$k_g = \{g,j,l_p\}$	$k_g = \{g,j,l_l\}$	$k_g = \{g,j,l_l,l_q\}$



(a) (b) (c) (d)

Fig. 5. (a) skeletonisation/contouring (b) chaining (c) polygonisation (d) matching

5 Conclusion and Perspectives

In this paper we have presented an approach for graphical knowledge management in graphics recognition systems. This approach is based on a "generic" formalism allowing various modellings of a given graphical shape. This formalism is based on object-oriented concepts, especially for the inheritance, polymorphism and extensibility properties. We represent and operationalize this formalism through our modelling library. We use this library into graphics recognition systems to request graphical knowledge databases, according to the processings' requirements on graphical primitives. Like this, this approach allows the interoperability between processings, especially for their combination. For the perspectives, in a first step we wish to develop a complete platform of graphics recognition processing based on our formalism. We would like to exploit the interoperability between processings to develop some strategic approaches [6]. Next, we wish to extend our request based approach with request language to extract graphical object structures, through graph request [7].

References

1. S. Ablameyko and T. Pridmore. Machine Interpretation of Line Drawing Images. Springer Verlag Publisher, 2000.
2. Autodesk. Drawing Interchange and File Formats, Release 12. Autodesk Inc, 1992.
3. B. Coüasnon. Dmos : a generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems. In International Conference on Document Analysis And Recognition (ICDAR), 2001.
4. D. Crevier and R. Lepage. Knowledge-based image understanding systems: A survey. Computer Vision and Image Understanding (CVIU), 67(2):161-185, 1997.
5 M. Delalandre, E. Trupin, and J. Ogier. Local structural analysis: A primer. Lecture Notes in Computer Sciences (LNCS), 3088:220-231, 2004.
6. M. Delalandre, Y. Saidali, J. Ogier, and E. Trupin. Adaptable vectorisation system based on strategic knowledge and xml representation use. Lecture Notes in Computer Sciences (LNCS), 3088:196-207, 2004.
7. R. Giugno and D. Shasha. Graphgrep : A fast and universal method for querying graphs. In International Conference on Pattern Recognition (ICPR), 2002.
8. L. Henderson and A. Mumford. The CGM Handbook. Academic Press, 1993.
9. X. Hilaire and K. Tombre. Improving the accuracy of skeleton-based vectorisation. In Workshop on Graphics Recognition (GREC), 2001.
10. J. Lladós, E. Valveny, G. Sánchez, and E. Martí. Symbol recognition : Current advances and perspectives. In Workshop on Graphics Recognition (GREC), 2001.
11. J. Murray and W.V. Ryper. Encyclopedia of Graphic File Formats. Editions O'Reilly, 1996.
12. B. Pasternak and B. Neumann. The role of taxonomy in drawing interpretation. In International Conference on Document Analysis And Recognition (ICDAR), 1995.
13. J. Pesonen. Concepts and object-oriented knowledge representation. Master's thesis, Department of Cognitive Science, University of Helsinki, Finland, 2002.
14. Y. Saidali, S. Adam, J. Ogier, E. Trupin, and J. Labiche. Knowledge representation and acquisition for engineering document analysis. Lecture Notes in Computer Science (LNCS), 3088:25-36, 2004.
15. J. Song, F. Su, C. Tai, and S. Cai. An object-oriented progressive-simplification based vectorisation system for engineering drawings: Model, algorithm and performance. Pattern Analysis and Machine Intelligence (PAMI), 24(8):1048-1060, 2002.
16. J. Sowa. Knowledge Representation: Logical, Philosophical, and Computational Foundations. Cole Publishing Co, 1999.
17. J. Spinrad. Efficient graph representations. In Fields Institute Monographs, volume 19. American Mathematical Society, 2003.
18. C. Suen, S. Mori, S. Kim, and C. Leung. Analysis and recognition of asian scripts - the state of the art. In International Conference on Document Analysis and Recognition (ICDAR), pages 866- 878, 2003.
19. K. Tombre and B. Lamiroy. Graphics recognition - from re-engineering to retrieval. In International Conference on Document Analysis and Recognition (ICDAR), 2003.
20. J. Ullman. Principles of Data-Base and Knowledge Base Systems, volume 1-2. Computer Sciences Press, 1989.
21. W3C. Scalar Vector Graphics (SVG) 1.0 Specification. 2001.
22. L. Wenyin and D. Dori. From raster to vectors : Extracting visual information from line drawings. Pattern Analysis and Applications (PAA), 2(2):10-21, 1999.
23. T. Williams. Object architecture dealing with the unknown - or - type safety in a dynamically extensible class library. Technical report, Microsoft Corporation, 1988.
24. H. Xue. Building skeletal graphs for structural feature extraction on handwriting images. In International Conference on Document Analysis And Recognition (ICDAR), 2001.