Symbol Detection Using Region Adjacency Graphs and Integer Linear Programming

Pierre Le Bodic, Hervé Locteau, Sébastien Adam, Pierre Héroux, Yves Lecourtier, Arnaud Knippel

Symbol Detection Using Region Adjacency Graphs and Integer Linear Programming

Pierre Le Bodic
LRI UMR 8623
Using Université Paris-Sud
91405 Orsay Cedex France

Hervé Locteau
LORIA UMR 7503
Campus Scientifique BP 239
54506 Vandoeuvre-lés-Nancy

Sébastien Adam
LITIS EA 4108
Université de Rouen
76800 Saint Etienne du Rouvray

Pierre Héroux
LITIS EA 4108
Université de Rouen
76800 Saint Etienne du Rouvray

Yves Lecourtier
LITIS EA 4108
Université de Rouen
76800 Saint Etienne du Rouvray

Arnaud Knippel
LMI EA 3226
INSA de Rouen
76130 Mt St Aignan

Abstract

In this paper, we tackle the problem of localizing graphical symbols on complex technical document images by using an original approach to solve the subgraph isomorphism problem. In the proposed system, document and symbol images are represented by vector-attributed Region Adjacency Graphs (RAG) which are extracted by a segmentation process and feature extractors. Vertices representing regions are labeled with shape descriptors whereas edges are labeled with feature vector representing topological relations between the regions. Then, in order to search the instances of a model graph describing a particular symbol in a large graph corresponding to a whole document, we model the subgraph isomorphism problem as an Integer Linear Program (ILP) which enables to be error-tolerant on vectorial labels. The problem is then solved using a free efficient solver called SYMPHONY. The whole system is evaluated on a set of synthetic documents.

1 Introduction

Labeled graphs are powerful data structure for the representation of complex entities. In a graph-based representation, nodes and their labels describe objects (or parts of objects) while labeled edges represent interrelationships between the objects. Due to the inherent genericity of graph-based representation, and thanks to the improvement of computer capacities, structural representations have become more and more popular in many application domains such as biology, chemistry, text processing, document analysis or symbol recognition. As a consequence of the emergence of graph-based representation, many computing issues such as graph mining, graph clustering, graph classification or subgraph isomorphism are gaining a growing popularity in the recent years.

This paper deals with the approximate subgraph isomorphism problem applied to a symbol detection application. Using a vector-attributed Region Adjacency Graph (RAG) for representing symbols and documents, we propose a new framework to tackle the subgraph isomorphism problem in order to find symbol instances on a whole document. This new subgraph isomorphism framework relies on a modeling of the problem as an Integer Linear Program (ILP). The proposed formulation aims at finding exact matching from the graph topology point of view but is error-tolerant concerning the labels of vertices and edges. The problem is then solved using a free efficient solver called SYMPHONY[1] proposed under a common public licence in the operational research community. The whole system is evaluated on a set of 15 synthetic documents built using a free system described in [2].

The paper is organized as follows: in section 2 the process which is used to transform images into graphs is briefly described. In section 3, we present the modeling of the subgraph isomorphism problem as an Interger Linear Problem. In section 4, we describe our experimental protocol, the datasets used and the obtained results. We finally draw some conclusions and we discuss future works in the last section.

[1] http://www.coin-or.org/SYMPHONY/index.htm

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.202

2 Symbol detection using RAG

Symbol detection is a problem related to document analysis which consists in searching some symbols in an input document image. Such a problem is much more difficult than recognizing isolated symbols since it requires to solve the well known Sayres paradigm by simultaneously segmenting the symbols and recognizing them. Consequently, if the literature concerning symbol recognition is very abundant, one can see that very few approaches are proposed in the literature to solve the symbol detection problem. In this paper, we present a whole system that is able to solve this problem in a particular context. The proposed approach relies on two levels. The first level consists in approximating both model symbols and document images as attributed Region Adjacency Graphs and the second consists in finding model graphs that represent the model symbol as a subgraph of an input graph representing the document. We describe the first step in the following of this section.

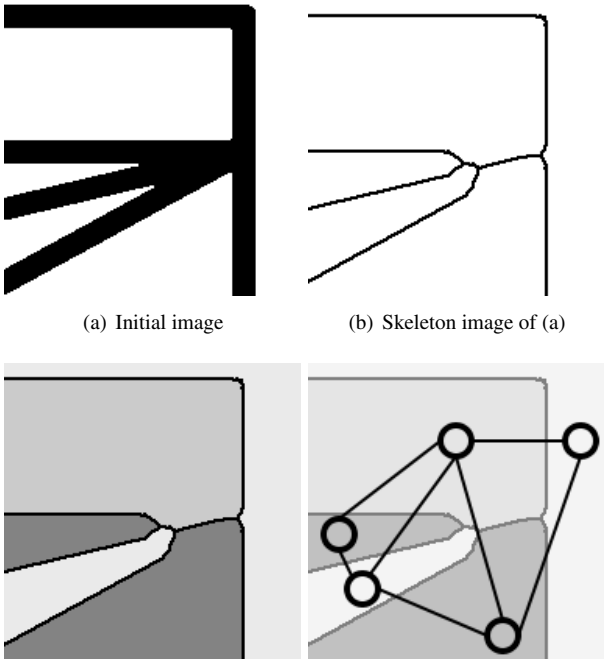
Region Adjacency Graphs are well suited for representing symbols since they enable to model the topological relations between the regions extracted by a segmentation process. Working on technical documents, the digital images are mainly binary images where white components denote the background while black components stand for the drawings. Segmenting such kind of images can be achieved using component labelling [4]. However, aiming at finely modeling adjacency relations between pairs of regions, a binary image can be firstly thinned [3]. The obtained image is then morphologically the same than the initial image of the document but the thickness of the drawing components is reduced to a single pixel. Using this image, each white component is mapped to a node of the graph. Then, scanning the skeleton branches, an adjacency relation, an edge, is build according to the neighborhood of each pixel between two regions, two nodes. The figure 1 illustrates the overall process on a document crop.

To discriminate regions and their relations, attributes have do be assigned to each node and edge. Many features have been proposed in the past to characterize shapes and spatial relations [7]. Among them, Zernike Moments [6] yield interesting results for pattern recognition tasks when invariance to affine transforms and robustness to degradations are required. Hence, a feature vector corresponding to a set of ZM is assigned to each node in order to characterize shapes. To obtained a rich representation of the data, our graph is made directed and relative attributes of the edge $(source \rightarrow target)$ are defined :

- *relative scale* of the target region w.r.t. to the source region : $\sqrt{\frac{A(target)}{A(target)+A(source)}}$,
- *relative distance* between gravity centers of a pair of regions is computed w.r.t. the overall area of the two regions : $\frac{d_e(g_{source},g_{target})}{\sqrt{A(source)+A(target)}}$.

We finally get a graph-based representation $G = (V, E, L_V, L_E)$ of a document where V and $E \subseteq V \times V$ stand for the set of nodes (region) / of edges (adjacency relations) and where L_V and L_E are mapping functions :

- $L_V : V \mapsto \mathcal{R}^d$ describing the morphology of a region (a node),
- $L_E : E = (n_i, n_j) \mapsto \mathcal{R}^2$ expressing the geometrical properties of an adjacency relation (an edge).



(a) Initial image (b) Skeleton image of (a)

(c) Components labels of (b) (d) Corresponding RAG

Figure 1. Raster to Region Adjacency Graph.

Once RAG have been built from document image, the symbol detection problem turns into a subgraph isomorphism problem. In the following section, we present the main contribution of this paper *i.e.* a new subgraph isomorphism formalism.

3 Modeling the Subgraph Isomorphism Problem as an Integer Linear Program

3.1 Integer Linear Programming

To solve the Subgraph Isomorpism Problem, we have modeled it as an *Integer Linear Program (ILP)* [5, 1]. The

standard form of an ILP is as follows:

$$\min_x c^t x \tag{1a}$$

$$\text{subject to } Ax \leq b \tag{1b}$$

$$x \in C \subseteq \mathbb{Z}^n \tag{1c}$$

where $c \in \mathbb{R}^n, A \in \mathbb{R}^{n*m}, b \in \mathbb{R}^m$ are data of the problem. This *Mathematical Program* defines a set of solutions for the modeled problem. A solution is a vector (x) of n *variables*. In the case of integer programming, each variable belongs to a subset of $\mathbb{Z}$ (1c). These variables are used to express linear *constraints* (1b). A valid solution for the original problem is a vector x such that constraints (1b) and (1c) are respected. Such a solution is said to be *feasible*. To find an optimal solution, we minimize (or maximize) the *objective function* (1a) over the set of feasible solutions.

3.2 Modeling the Subgraph Isomorphism Problem

In order to model the problem as an ILP, we only use binary variables, i.e. the solution takes its value in $\{0,1\}^n$. We define two kind of variables (as depicted in figure 2):

- For every vertex $i \in V_\mathcal{S}$ and for every vertex $k \in V_\mathcal{G}$, there is a variable $x_{i,k}$, such that $x_{i,k} = 1$ if vertices i and k are matched together, 0 otherwise.
- For every edge $ij \in E_\mathcal{S}$ and for every edge $kl \in E_\mathcal{G}$, there is a variable $y_{ij,kl}$, such that $y_{ij,kl} = 1$ if edges ij and kl are matched together, 0 otherwise.

Given $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ and $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$, let us assume that we know a distance $d : V_\mathcal{S} \times V_\mathcal{G} \rightarrow \mathbb{R}^+$ as well as a distance (also refered to as d for convenience) $d : E_\mathcal{S} \times E_\mathcal{G} \rightarrow \mathbb{R}^+$. Let us consider two vertices $i \in V_\mathcal{S}$ and $k \in V_\mathcal{G}$. Pairing i and k (i.e. $x_{i,k} = 1$) costs $d(i,k)$, whereas not pairing them (i.e. $x_{i,k} = 0$) costs 0. The cost can thus be written $d(i,k) * x_{i,k}$. Similarly, the cost between two edges $ij \in E_\mathcal{S}$ and $kl \in E_\mathcal{G}$ is $d(ij,kl) * y_{ij,kl}$. Let us now write the objective function. We would like to minimize the sum of the distances between the elements (i.e. vertices and edges) that are matched together:

$$\min_{x,y} \sum_{i \in V_\mathcal{S}} \sum_{k \in V_\mathcal{G}} d(i,k) * x_{i,k} + \sum_{ij \in E_\mathcal{S}} \sum_{kl \in E_\mathcal{G}} d(ij,kl) * y_{ij,kl} \tag{2a}$$

Let us now consider the constraints of the ILP:
Every vertex of $V_\mathcal{S}$ must be matched to a unique vertex of $V_\mathcal{G}$:

$$\sum_{k \in V_\mathcal{G}} x_{i,k} = 1 \quad \forall i \in V_\mathcal{S} \tag{2b}$$

Every edge of $E_\mathcal{S}$ must be matched to a unique edge of $E_\mathcal{G}$:

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = 1 \quad \forall ij \in E_\mathcal{S} \tag{2c}$$

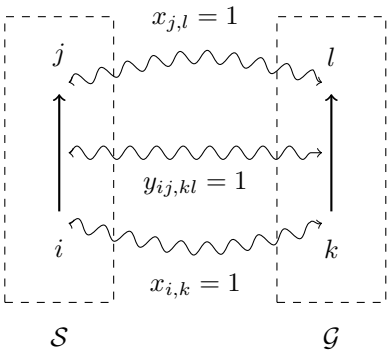


Figure 2. an example of matching. $\mathcal{S}$ and $\mathcal{G}$ both contain a single edge, respectively ij and kl. The following solution is represented on this figure : $x_{i,k} = 1$ (resp. $x_{j,l} = 1$, $y_{ij,kl} = 1$), i.e. i (resp. j, ij) is matched with k (resp. l, kl). Reversely, since i (resp. j) is not matched with l (resp. k), $x_{i,l} = 0$ (resp. $x_{j,k} = 0$).

Every vertex of $V_\mathcal{G}$ must be matched to at most an edge of $E_\mathcal{S}$:

$$\sum_{i \in V_\mathcal{S}} x_{i,k} \leq 1 \quad \forall k \in V_\mathcal{G} \tag{2d}$$

If two vertices are matched together, an edge originating one of these two vertices must be matched with an edge originating the other vertex:

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = x_{i,k} \quad \forall k \in V_\mathcal{G}, \forall ij \in E_\mathcal{S} \tag{2e}$$

If two vertices are matched together, an edge targeting one of these two vertices must be matched with an edge targeting the other vertex:

$$\sum_{kl \in E_\mathcal{G}} y_{ij,kl} = x_{j,l} \quad \forall l \in V_\mathcal{G}, \forall ij \in E_\mathcal{S} \tag{2f}$$

Finally, let us properly write the aforementioned domain constraints of the variables:

$$x_{i,k} \in \{0,1\} \quad \forall i \in V_\mathcal{S}, \forall k \in V_\mathcal{G} \tag{2g}$$

$$y_{ij,kl} \in \{0,1\} \quad \forall ij \in E_\mathcal{S}, \forall kl \in E_\mathcal{G} \tag{2h}$$

Equations (2a) to (2h) form the ILP used to solve the Subgraph Isomorphism Problem.

Once modeled, this problem can be solved using a mathematical solver. In this study, we use a free solver called SYMPHONY. In the following section, our experiments and results are described.

4 Experiments

In this section we report our experiments in the context of symbol detection in architectural floor plans. We first describe the databased we use. Then, we present our experimental protocol and finally we discuss the obtained results.

In our symbol detection experiments, we use a subset of 15 synthetic graphical documents from the database provided by Delalandre *et al.* [2]. Their system allows to build graphical documents for performance evaluation. The key idea is to position symbols, or even any graphical part, on an image according to a set of constraints. The system engine is in charge of randomly adding new symbols to the document, without overlapping, preserving the layer property described by the user rules. These involved predefined distributions of symbol models and affine parameter ranges. More details can be found in the former reference. An examples of architectural drawings, using a given *background layer* made of walls, is depicted in figure 3.

In order to evaluate the performance of the proposed system, we have applied the following procedure. For each existing model graph, we have queried the system on the 15 documents described above which contain 220 symbols.

Figure 4 illustrates the results of such a query corresponding to the 6^{th} symbol of table 1. The color code and the numerical values indicate the different symbols in the order they are found by the system. In this example, the system has been tuned in order to return the 10 best results (symbols) for each query. One can see that the three first answers correspond to the symbol which is searched for. Moreover, one can also notice that the following answers are quite logical since they correspond to more complex symbols which include the queried symbol.

The obtained performance on the whole dataset are synthetized in table 1. The global detection rate of the system reaches 87.5%.

symbol	# of occurences	Detection rate
	17	100%
	48	87.5%
	50	54%
	13	100%
	12	91.66%
	47	100%
	10	100%
	14	86.66%
	10	90%

Table 1. Obtained detection rates.

5. Conclusions

In this paper, we have proposed a complete system for symbol detection on technical document. The proposed approach relies on a representation of documents and symboles as vector-attributed region adjacency graphs. Our main contribution in this context is the proposition of a new framework for partially inexact subgraph isomorphism which exploits this vector labeling. This framework relies on a modeling of the problem as an Integer Linear Program (ILP) which is solved using a solver proposed in the operational research community. Preliminary obtained results prove that this algorithm is able to find 87.5% of 220 symbols on 15 whole architectural documents. Such results are encouraging since no optimization has been done on the system. In the future, we plan to extend the proposed mathematical program to inexact subgraph isomorphism. We also plan to associate a learning procedure to the existing algorithm in order to optimize the features which are used as labels on vertices and edges. We also plan to include an optimization of the number of symbols to be returned.

References

[1] J. W. Chinneck. Practical optimization: A gentle introduction.

[2] M. Delalandre, T. Pridmore, E. Valveny, H. Locteau, and E. Trupin. Building synthetic graphical documents for performance evaluation. In *Graphics Recognition. Recent Advances and New Opportunities*, volume 5046 of *Lecture Notes in Computer Science*, pages 288–298. 2008.

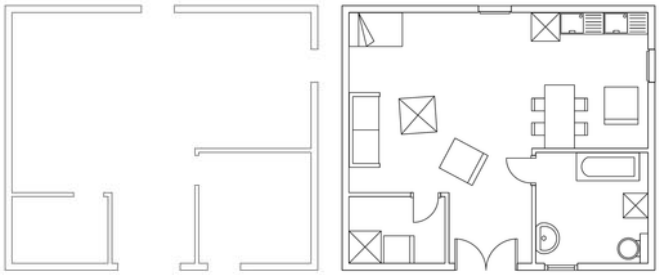
[3] G. S. di Baja and E. Thiel. Skeletonization algorithm running on path-based distance maps. *Image and Vision Computing*, 14:47–57, 1996.

[4] C.-J. C. Fu Chang and C.-J. Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93:206–220, 2004.

[5] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[6] M. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, 1980.

[7] O. R. Terrades, S. Tabbone, and E. Valveny. A review of shape descriptors for document analysis. In *International Conference on Document Analysis and Recognition*, pages 227–231, 2007.



(a) Background Layer (b) Example of synthetized architectural drawing

Figure 3. Examples of synthetic architectural drawings used for our experiments. Obtained by [2] from the same "background layer" (left).

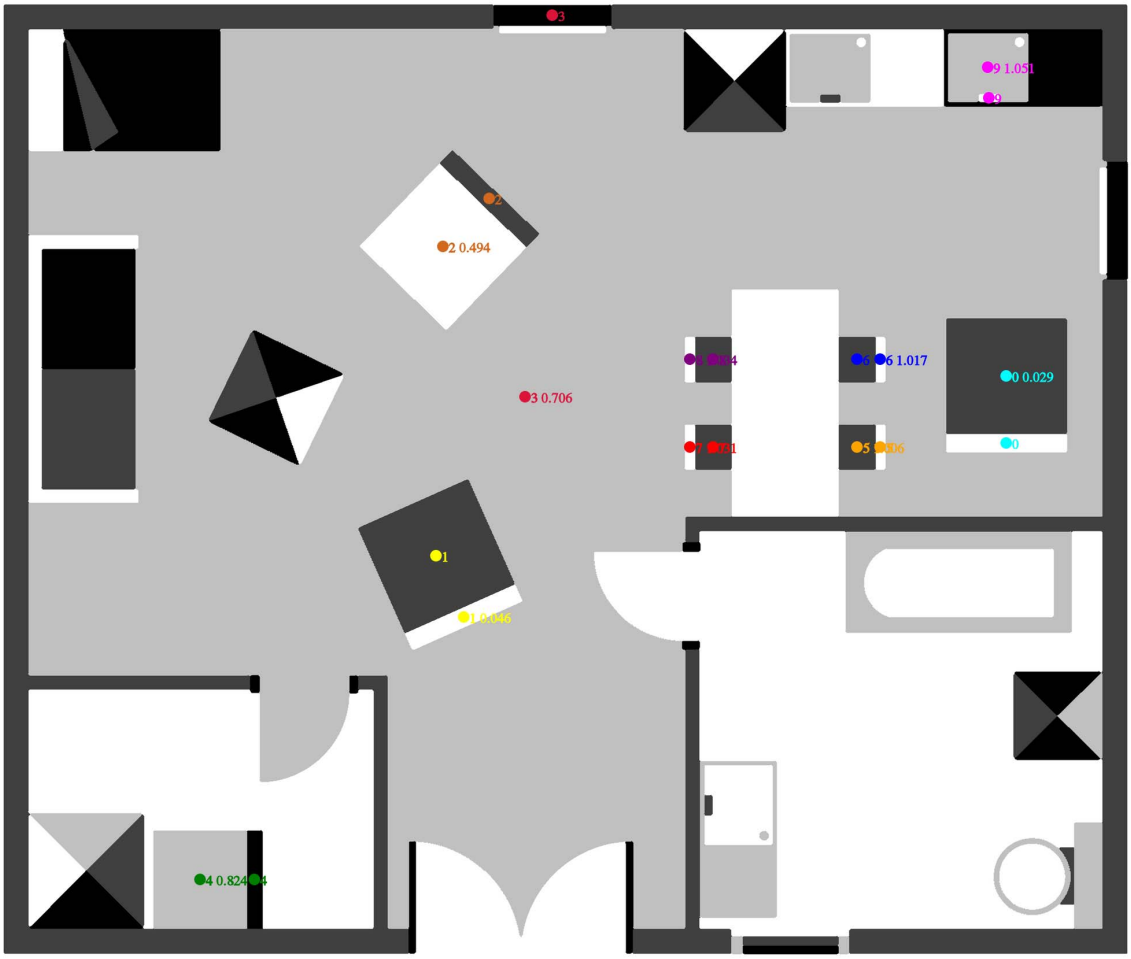


Figure 4. Examples of query results