A Method for Symbol Spotting in Graphical Documents

Daniel Zuwala and Salvatore Tabbone

Loria-UMR 7503,
Campus Scientifique, B.P. 239,
54506 Villers-les-Nancy Cedex, France
{dzuwala, tabbone}@loria.fr

Abstract. In this paper we propose a new approach to find symbols in graphical documents. The method is based on a representation of the document in chain points extracted from the skeleton. We merge successively these chain points into a dendrogram framework and according to a measure of density. From the dendrogram, we extract potential symbols which can be recognized after.

1 Introduction

Symbol recognition is more than ever a problem that is discussed in scientific community [1]. Several approaches have been proposed and one kind is based on the structural representation of documents [6–10]. These approaches are often very powerful to spot symbols but they suffer of large complexity and they are not robust against the noise. Another kind of approaches is based on features descriptors [2–5] . These descriptors are either considering the contour of the object or either considering the whole object. They are robust against noise and occlusion, but they need the document to be clearly segmented, which is a problem since symbols are often embedded with other graphics layers.

In our previous work [11], we have proposed a method to automatically detect potential symbols without knowledge. However this method was only working on special kind of symbols that had a loop contour. Moreover, it was time-consuming to detect the loop symbols since we were looking for loop in a graph. In this paper, we proposed a new method that do not have the loop structure constraint, but only the constraint that the symbol is defined by a single connected component. We have also overcome the time consuming problem, by developing a new algorithm to detect symbols. The method is based on a structural representation of the document which is decomposed into different chain points. Then, we merge successively these chain points in order to retrieve symbols. Finally we used the photometric information to discard or accept symbols with respect to symbol model. This method is fast, and can be seen as a first processing, in order to have a set of candidate symbols that may be interesting with respect to a symbol query.

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 518–528, 2006.

The paper is organized as follows. In section 2 we explain our approach. Examples and experimental results are given in section 3. Limit and future investigations are provided in section 4.

2 The Method

2.1 Low Level Processing

The documents are usually scanned in grey levels, and a binarization is done in order to be able to extract symbols from graphical documents (see Fig 1). Then, the skeleton is defined using the 3-4 distance transform ([12]) (see Fig 2). Finally, we extract from the skeleton image a set of chain points. These chains are composed of connected points that have only two neighbors, and the extremities are either junction points (ie more than two neighbors) or terminal points (ie only one neighbor) as indicated in Fig 3.

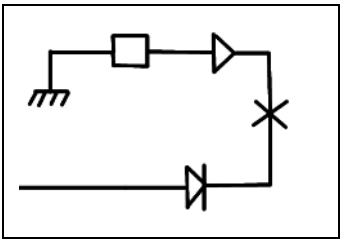

Fig. 1. Binarized document

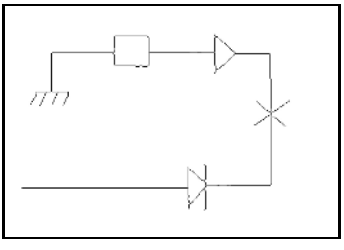

Fig. 2. Extracted skeleton

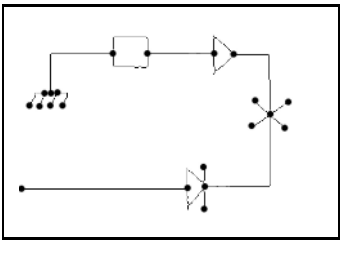

Fig. 3. Document with junctions and terminal points

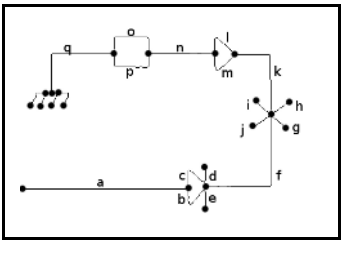

Fig. 4. Defined chain points (denoted a,b,c,...)

At the end of these low level processing, the whole document is decomposed into a set of chain points (Fig 4).

2.2 Main Processing

The idea is to merge these chain points in order to segment the symbols belonging to a document. We assume that the symbols we are looking for are defined by a set of compact connected chains (Fig 5).

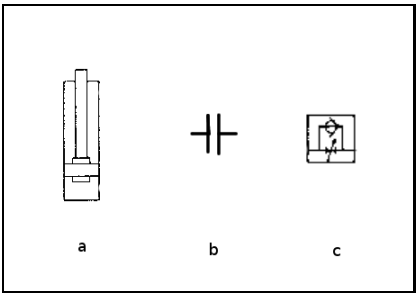

Fig. 5. Examples of symbols that can be recognized or not. Symbol (a), may not be recognized because it is not compact enough. Symbol (b), will not be recognized because it is composed of two disconnected components. Symbol (c), will be recognized.

So in order to find them, we have to define a measure that can tell us, how compact a set of chain points is. In this perspective, let C be a chain points, x_i a point of C, $i \in [1, n_C]$ where n_C is the total number of points in C. We define the density of C as:

$$density(C) = \frac{\sum_{i=1}^{n_C} d(x_i, x_C^G)}{n_C}$$

where, x_C^G is the barycenter of the chain point C defined by $x_C^G = \frac{1}{n_C} \sum_{i=1}^{n_C} x_i$, and $d(x_i, x_C^G)$ a distance (we have taken here, the Euclidean distance).

The idea is to merge successively these chains into a dendrogram structure that can be seen as a graph representation. The nodes of the graph are the chain points, and the edges describe the relation between the chain points. For example, if C_1 and C_2 are directly connected (ie they have common extremities), there is an edge between them. To each edge we assign a value to know how relevant is a merge between two corresponding nodes. The value that we assign to the edge (C_1, C_2) is the following:

$$value(C_1, C_2) = density(C_1) density(C_2) \frac{(n_{C_1} + n_{C_2}) density(C_1 C_2)}{n_{C_1} density(C_1) + n_{C_2} density(C_2)} \quad (1)$$

where n_{C_1}, n_{C_2} are weighted coefficients on the number of merges that already happened in clusters C_1 and C_2. The second part of the expression formula :

$$\frac{(n_{C_1} + n_{C_2}) density(C_1 C_2)}{n_{C_1} density(C_1) + n_{C_2} density(C_2)}$$

is here to take into account the evolution of the density. Before the merge, we have two nodes of density: $density(C_1)$ and $density(C_2)$. After the merge we will have one density : $density(C_1 C_2)$. We look at the ratio between them, in order to measure how the density has evolved before and after the merge. So it means that we are looking for the smallest evolution of the density. But this term

alone is not sufficient because if we take a large chain of points, the evolution with a neighbor chain will be very small. To avoid this behavior we add the first term:

$$density(C_1)density(C_2).$$

This term underlines small density chain points. So *value* (formula 1) is a measure that tells that we want a small evolution of the density but also that we want a small density chain point first.

```
for each node in the graph
    calculate the density of the node
end for
for each edge in the graph
    calculate the value of the edge
    insert the edge in a queue Q sorted by increasing value
end for
while Q not empty do
    pop the first edge of Q
    create a new node from this edge
    update the value of edges that have changed because of the new node
end while
```

Fig. 6. Algorithm to construct the dendrogram

We show a sample of a dendrogram in Fig 7 provided from the algorithm defined on Fig 6. At the beginning (level 0), we have small chain points. Then we successively merge them (as indicated by Fig 6). Finally, we obtain a binary tree.

Once we have constructed the dendrogram, we have to find how to extract the best partition (ie the partition that give us the decomposition of symbols). Fig 8 shows the evolution of the measure *value* during the dendrogram construction for the document in Fig 4.

The following figures 9, 10 and 11 show the partition we get when $n = 22, 26, 30$. n is a parameter on the number of merges that have been done since the beginning of the processing, it is related to the level in Fig 7). From these figures, we can remark that for $n = 22$, we find out that there is still some symbols that have not been merged. For $n = 26$ all the symbols are good and for $n = 30$ some symbols have merged with lines.

So it seems difficult to find the right value of n. Of course we don't have to select a single n, but a range. However we get a full partition of the document and we are not able to make a difference between candidate symbols and garbage symbols (like lines network). So many symbols are returned. That's why we have developed another method in order to filter the number of potential symbols returned.

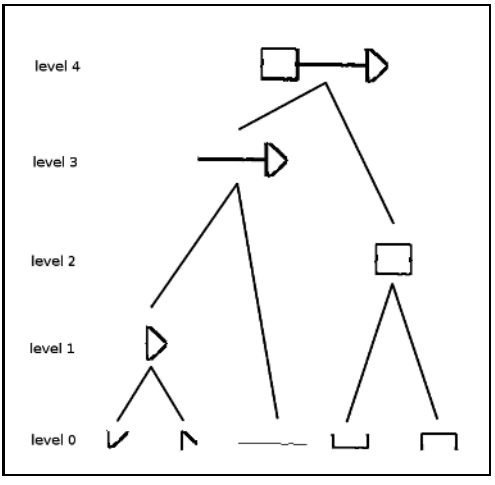

Fig. 7. A dendrogram

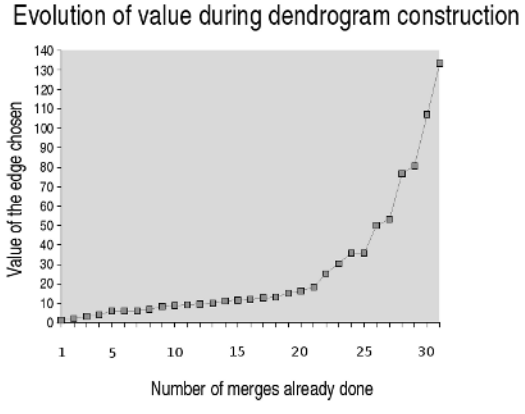

Fig. 8. Evolution of *value* during the dendrogram construction

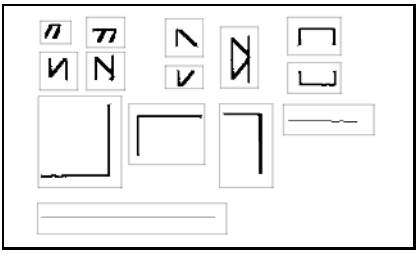

Fig. 9. Partition found with $n = 22$

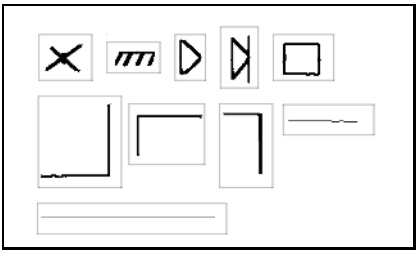

Fig. 10. Partition found with $n = 26$

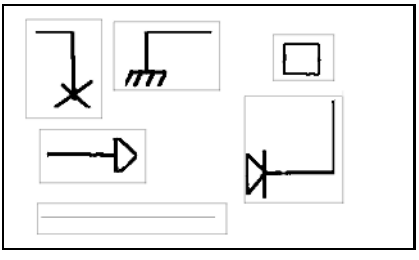

Fig. 11. Partition found with $n = 30$

2.3 Another Method to Extract the Symbols

Instead of having a global view of the dendrogram, let's have a local view of it. A dendrogram is a binary tree, where the leafs are the initial chain points that we have extracted from the skeleton image. If for each leaf, we start looking the evolution of *value* from the leaf to the top of the tree, we will have a local view. The algorithm is described in Fig 12.

We have two parameters to set. The first one, $BoundingBoxMax$ is only here to skip chain points that are inconsistent (ie we don't look the symbols that are too big). It represents the maximum size of potential symbols. If we have no clue on the size of the symbols in the documents we can set $BoundingBoxMax = BoundingBox(document)$. The second one, $ratioMax$, is here to limit the evolution of *value*. When *ratio* is above $ratioMax$ we suppose that the candidate symbol has merged with another chain points that do not belong to the symbol.

Let's look at the results we have with this method (see Figs 13, 14 and 15). For $ratioMax = 1.2$, we notice that some symbols are incomplete. For $ratioMax = 1.5$, we have all, and only the symbols. For $ratioMax = 1.8$, we have a symbol that have merged with other stuff. So we can observe that we have significantly reduce the number of symbols. However we have added a parameter $ratioMax$ that have some influences on the results. As said above, one solution would be to gather all the results return from $ratioMax$=1.2 to 1.8, with a step of 0.1. Of course, we increase the number of potential symbols returned but we are rather sure, we do not miss one.

```
for each leaf in tree
    – vertexPrevious = leaf
    – vertex = leaf
    if BoundingBox(vertex) < BoundingBoxMax do
        – ratio = 1
        while ((vertex has a parent)and(ratio < ratioMax)) do
            – vertexPrevious=vertex
            – vertex=parent(vertex)
            – ratio=value(vertex)/value(vertexPrevious)
        end while
        if (ratio < ratioMax) return vertex
        else return vertexPrevious
    end if
end for
```

Fig. 12. Algorithm to find out symbols from a local view

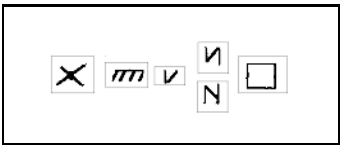

Fig. 13. Results with $ratioMax = 1.2$

Fig. 14. Results with $ratioMax = 1.5$

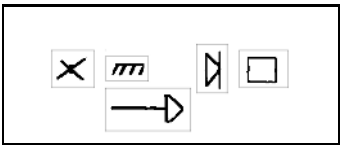

Fig. 15. Results with $ratioMax = 1.8$

2.4 Boosting the Search of Candidate Symbols

As shown before, we can have a same symbol that has been too much merged with other stuff, or that has been not enough merged. In order to detect symbols, a way to not check all the potential symbols is to organize them into a tree, like they were in the dendrogram. Figs 16 and 17 represent the symbols with a tree organization and without. That way, if we find a match with a symbol model, we don't have to look for a match with its children.

2.5 Descriptors

In order to find if the candidate symbols match the model, we are going to use a simple descriptor based on geometric moment[13]. The output of the descriptor

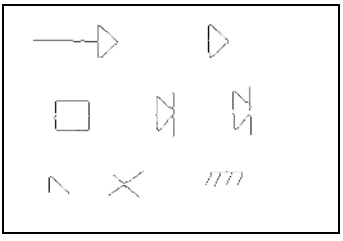

Fig. 16. Without tree organization. We have 9 matches to do.

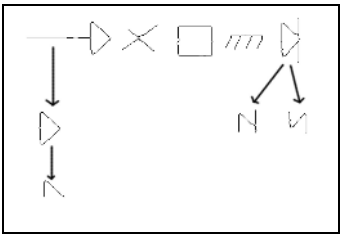

Fig. 17. With tree organization. Between 5 and 9 matches to do.

is a 4 dimensional vector. We compute the Euclidean distance between the two vectors to know if the symbols match together. That is, if this distance is below a fixed threshold, they match.

This descriptor is very simple, but it is fast. We may have to find better descriptors but it is not the subject of this paper. We will see in the experimentation section that the descriptor is not always discriminant. However, here the main idea is to have a system that can quickly display a set of candidate symbols.

3 Experimentation and Perspectives

Tests have been conducted on five different documents (Fig 19 and 20 are a sample of them). The queries symbols are presented in Fig. 18 and Table 1 shows the results obtained with our approach. We have reported on this table the number of symbols found related to the query compared to the real number belonging to the document. We also add the rank of the symbols retrieved.

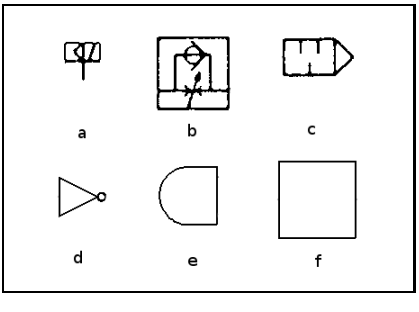

Fig. 18. Symbols to recognize

The results appeared promising. Considering all the experimentations we have made, we have found 35 symbols over 42. For the first image, everything has been recognized, but with variable ranks. We may reasonably hope that with a better descriptor, the ranks returned will be better. We notice that we have miss some symbols. There can be two reasons:

– The first one, is that after the dendrogram construction, we have failed to extract the correct symbols, and so we cannot recognized them. One can overcome this situation by relaxing the different parameters, or by simply returning all the symbols involved in the dendrogram construction. Of course it will be more time consuming to calculate a descriptor for all the symbols, but it is not impossible. For example, for the image in Fig 19, the dendrogram has decomposed the image around 1000 symbols. So it is still possible to calculate a descriptor for every symbols.
– The second reason, is that the method has failed to merge the correct chain points. This could be solved by finding a better definition of the density. Another improvement to overcome this situation should be to not only con-

Table 1. Results obtained with the proposed method. Number of symbols found / Number of symbols in the document. In parenthesis, the different rank of the symbols found.

Symbols	a	b	c	d	e	f
Test 1	4/4 (1;2;3;4)	4/4 (2;4;5;10)	4/4 (3;6;7;10)	0/0	0/0	0/0
Test 2	0/0	0/0	0/0	2/2 (2;4)	2/2 (1;2)	4/6 (1;2;3;5)
Test 3	0/0	0/0	0/0	2/3 (3;5)	4/4 (1;2;3;4)	0/0
Test 4	0/0	0/0	0/0	2/2 (1;2)	0/0	3/7 (3;4;5)
Test 5	0/0	0/0	0/0	4/4 (2;3;101;102)	0/0	0/0

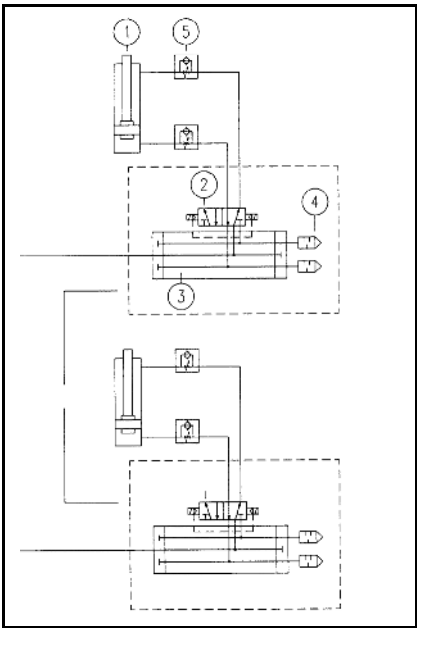

Fig. 19. Document test 1

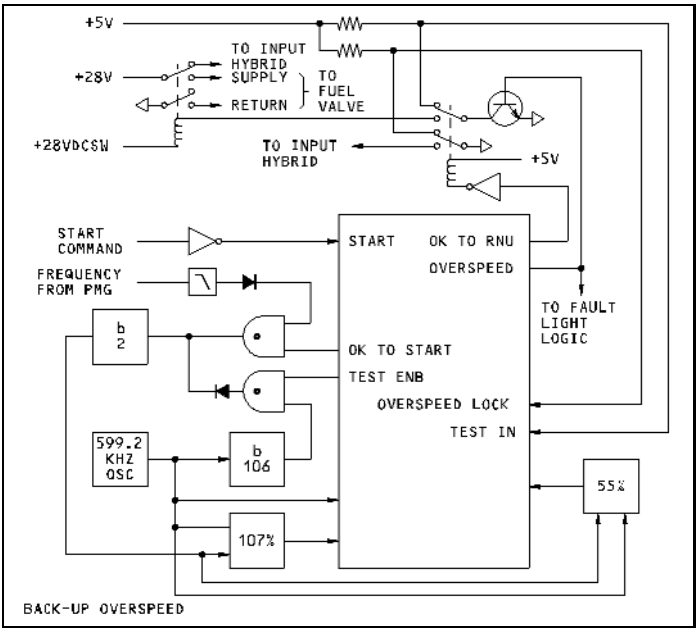

Fig. 20. Document test 2

sider the density of the chain points that are directly connected, but also the chain points that are connected n step later (with n integer greater than 1). Doing so, we will have a more global view of the document instead of the current local view during the construction of the dendrogram.

All the processing (from the skeletization to the output results) take about 5 seconds on Pentium IV (2.8GHz) on a document image of size 1200X2000 pixels.

4 Conclusion

We have proposed a new method to find symbols which do not need to be already segmented or disconnected from graphical documents. This method doesn't made any assumption on the symbols to find, and so can be applied to many different symbols. It assumes only that a symbol is a set of compact chain points, that is often the case. The results that we obtain are encouraging even if we have pointed out several drawbacks. In particular, a better descriptor is needed to improve immediately the performance of the approach. Futures works will be devoted to:

- define a better density measure. In particular the one we have used is more dedicated to concentric distribution. It will be possible to take into account elliptic distribution.
- extent the method to chain points that may be connected n step later (with n integer greater than 1).

References

1. Cordella, L.P., Vento, M.: Symbol recognition in documents: a collection of techniques? International Journal on Document Analysis and Recognition **3** (2000) 73–88
2. Belkasim, S.O., Shridar, M., Ahmadi, M.: Pattern Recognition with Moment Invariants: A Comparative Study and New Results. Pattern Recognition **24** (1991) 1117–1138
3. Belongie, S., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Transactions on PAMI **24** (2002) 509–522
4. Ghorbel, F.: A complete invariant description for gray level images by harmonic analysis approach. Pattern Recognition Letters **15** (1994) 1043–1051
5. Lin, B.C., Shen, J.: Fast Computation of Moment Invariants. Pattern Recognition **24** (1991) 807–813
6. Llados, J., Jose, J.: Symbol Recognition by Subgraph Matching Between Region Adjancy Graphs. IEEE Transactions on PAMI **23** (2001) 1137–1143
7. Messmer, B.: Automatic learning and recognition of graphical symbols in engineering drawings. In: Graphics Recognition: methods and applications (GREC'95), LNCS 1072. (1996) 123–134
8. Park, B.G, Lee, K.M., Lee, J.: Recognition of partially occluded objects using probabilistic ARG (attributed relational graph)-based matching. Computer Vision and Image Understanding **90** (2003) 217–241
9. Ramel, J.Y, Emptoz, H.: A structural representation adapted to handwritten symbol recognition. In: Graphics Recognition: methods and applications (GREC'99), Jaipur, India. (1999) 259–266
10. Yan, L., Wenyin, L.: Engineering drawings recognition using a case-based approach. In: International Conference on Document Analysis and Recognition, Edinburgh. Volume **2886** (2003) 190–194
11. Tabbone, S., Wendling L., Zuwala D.: A Hybrid Approach to Detect Graphical Symbols in Documents. In: Document Analysis Systems, Volume **3163** (2004),342–353
12. Sanniti di Baja, G.: Well-shaped, stable, and reversible skeletons from the (3,4)-distance transform, In:Journal of Visual Communication and Image Representation, **5** (1):107-115, March 1994.
13. Hu, M. K.: Visual pattern recognition by moment invariants, In: IEEE Trans. Inform. Theory, **8**, 1962.