

Symbol Spotting using Full Visibility Graph Representation

Hervé Locteau, Sébastien Adam, Eric Trupin, Jacques Labiche, Pierre Héroux

▶ **To cite this version:**

Hervé Locteau, Sébastien Adam, Eric Trupin, Jacques Labiche, Pierre Héroux. Symbol Spotting using Full Visibility Graph Representation. Workshop on Graphics Recognition, Jul 2007, Brazil. pp.49-50. hal-00671265

HAL Id: hal-00671265
https://hal.science/hal-00671265

Submitted on 17 Feb 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Symbol Spotting using Full Visibility Graph Representation

Hervé Locteau, Sébastien Adam, Éric Trupin,
Jacques Labiche, and Pierre Héroux

LITIS, EA 4051, University of Rouen,
St Étienne du Rouvray, France
Herve.Locteau@univ-rouen.fr

Abstract. In this paper, a method for matching symbols in line-drawings is presented. Facing both segmentation and recognition of symbols is a difficult challenge. Starting from the results of a vectorization procedure, a visibility graph is built to enhance the main geometric constraints which were specified during the construction of the initial document. The cliques detection, which correspond to a perceptual grouping of primitives, is used in the system to detect regions of particular interest. Both opened and perceptually closed curves are identified from aggregation of cliques. Finally, the recognition stage uses an attributed edit distance technique to match approximated curves within the host attributed relation graph and the ones from a collection of symbols.

Keywords: perceptual grouping, visibility graph, segmentation, symbol spotting, symbol recognition, cyclic string edit distance.

1 Introduction

Symbol recognition deals with identification of a given object but objects have also to be extracted in the input data. In specific cases, symbol localization can be triggered from the localization of the main white connected component (pixel based approaches) or the main minimal cycle within the junction graph (skeleton based approaches) leading to a framing window that enables the identification or rejection of the symbol. Aiming to achieve both localization and recognition of *a priori* known classes of symbols in a document requires to make those processes interacting. As an alternative to this proposal, particular patterns from a junction within the junction graph permit to dynamically define a region of interest from which a vectorial signature is extracted in [1]. This spotting technique postpone the recognition step restricting significantly the regions that have to be classified or rejected. It is both relevant for symbols built from a cycle and the ones that have no cycle in the junction graph. No matter localization and recognition are synchronized or not, although structural methods are generally sensitive to noise, they represent an appealing framework capturing the spatial and topological relationship between graphical primitives.

In this study, we investigate the use of visibility graphs for symbol representation as they look close to the perceptual grouping human vision may achieves. The relevancy of the final representation is twofold : it is suitable for (i.) symbols with or without any closed curve and (ii.) symbols built from disconnected components. This type of graph is introduced in the section 2. We present in section 3 how cliques within the visibility graph can be aggregated to build (nearly) closed curves that can then be indexed to provide regions of interest. Finally, our matching process based on an attributed edit distance is depicted in section 4.

2 Visibility Graph

Visibility representations are of particular interest in computer science such as graphics, VLSI layout, motion planning and computational geometry [2]. In this work, a visibility graph is defined from a set of n vectorial primitives such that each primitive is mapped to a node while edges represent visibilities between couples of primitives (u, v). The visibility relation used throughout between u and v corresponds to the existence of a point on u and a point on v defining a line segment that does not intersect any other primitive [3]. Such a relation has been called *weak visibilty* in some works and leads to a *full visibility graph*, FVG for short. As an example, we report the FVG build from a set of segments in the figure 1.

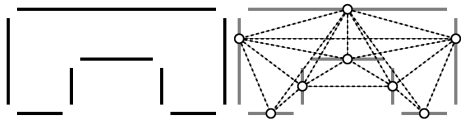


Fig. 1. Set of primitives and its Full Visibility Graph. Segments are separated from each other for clarity.

Algorithm 1 Building the full visibility graph of a set of n segments **Alg. 1.**

for all $i \in [0, n]$ **do**
  for all $Pe \in \{Begin(i) = B_i, End(i) = E_i\}$ **do**
    computes polar coordinates of extremities P_k of primitives j with respect to Pe
    sort the $(\theta(P_k), \rho(P_k))$ according to $\theta()$, then to $\rho()$
  end for
  computes the visibility polygon of Pe (see §2.1)
  extends the visibilities of primitives j around Pe (see §2.2)
end for

The algorithm used to extract the FVG of a segments' set is summarized in (Alg. 1). For a given primitive j, let S_j and F_j stand respectively for the first and the second, counterclockwise endpoint, thus :

- if it is hardly connected to primitive i (i.e. $B_j = Pe$ or $E_j = Pe$), the junction point's angle is set to the opposite one of $\overline{Pe}$ (the other endpoint of the primitive i),
- since $\widehat{S_j\ Pe\ F_j} < \pi$, if $\theta(F_j) < \theta(S_j)$, $\theta(F_j) = \theta(F_j) + 2\pi$.

The procedures corresponding to the extraction of the visibility polygon of a point Pe and its use to extend the visibility of primitives around this point are both based on line sweep algorithms.

2.1 Computing the visibility polygon of a point

Following the Wismath's idea, we first computes the visibility polygon of a point. The primitives are sorted within a stack by their distance between a reference point and the intersection point of a ray sweeping the plane. The algorithm (Alg. 2) permits to build the visibility polygon $VPoly(Pe)$ of a point Pe. Dealing with (i.) colinear points and (ii.) common points are taken into account when we update the stack (insertion and deletion) and when we evaluate whether a point is visible or not. Obviously, a first series of visibilities involving the primitive i can be directly extracted from $VPoly(Pe)$. Nevertheless, it is also necessary to seek for particular configurations within $VPoly(Pe)$ in order to specify a second series of visibilities involving primitives both referenced in $VPoly(Pe)$.

2.2 Extending the visibility of primitives around a point

From the visibility polygon's building procedure of a point Pe, we can note that visibilities between a couple of primitive (i, j) are extracted from at least an endpoint of either the primitive i or the primitive j. In order to detect visibilities involving banal points (points distinct from endpoints) both for a primitive u and a primitive v, further works should be done.

From a series of observations, Wismath highligthed in [3] the configuration around Pe from $VPoly(Pe)$ which permits to fully define the set's visibility graph of n non-intersecting line segments.

Given three points of $VPoly(Pe)$, saying q_u, q_a, q_b such as $\widehat{q_u\ Pe\ q_a} < \pi$ and $\widehat{q_u\ Pe\ q_b} > \pi$. If points q_a and q_b have a corresponding primitive v, then we can report the primitive u, corresponding to q_u, and v are visible around Pe. Indeed a perception can be issued from q_u from either $\widehat{q_a\ q_u\ Pe}$ or $\widehat{Pe\ q_u\ q_b}$, no matter the position of $\overline{Pe}$ (the other endpoint of the primitive i). Such case is illustrated by the figure 2.

However, due to the management of colinear and equal points, the procedure should include a verification step. We have to check that no radial primitive hides the perception issued from q_u. For example, if a primitive whom extremities are

Algorithm 2 Computing the visibility polygon of Pe **Alg. 2.**

```
stack ← ∅
VPoly(Pe) ← ∅
for all α ∈ {θ(P_k)}_{k∈K} do
  sorts the {P_k|θ(P_k) = α} such as, for a given distance,
                                  primitives in the stack are treated in priority
  for all P_k|θ(P_k) = α do
    if P_k is the first point of primitive j counterclockwise then
      insert j in the stack according the visibility of Pe in the orientation α
      if P_k is visible then
        if P_k in front of last point of VPoly(Pe) then
          replaces last point of VPoly(Pe) by P_k
        else
          add P_k to VPoly(Pe)
        end if
      end if
    else
      if P_k is visible then
        if ∃M behind P_k such as M will be visible after P_k is treated then
          add M to VPoly(Pe)
        end if
        add P_k to VPoly(Pe)
      end if
      removes j from the stack
    end if
  end for
  if the ray (Pe − α) intersects the primitive at the head of the stack then
    add the intersection point in VPoly(Pe)
  end if
end for
```

q_5 and Pe is added to the initial set of primitives, visibility must not be created from the tuple $(q_u, q_a, q_b) = (q_2, q_8, q_9)$ considering the notations of the figure 2.

3 Identifying perceptually closed curves

The Gestalt laws of perception (e.g. *proximity*, *smooth continuation*, *closure*) seem to depict the perceptual phenomena exploited by the human vision system. As the Saund's paper ([4]) reads, a visual system will take advantage of a detection of closed or nearly closed curves. Coherent objects, such as the symbols used in technical drawing domains, tend to be spatially compact and relatively uniform in surface appearence with respect to the surrounding background.

In this work, we adopt the famous framework that consists in starting from seed curve fragments and trying to extend it by tracing from one clique to another with which it is associated by at least one shared primitive. The key problem is to determine whether a new clique can be merged or not since primitives have not beforehand been clustered within a single partition.

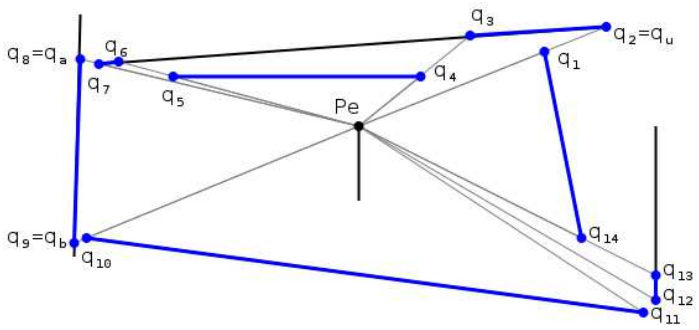


Fig. 2. Visibility polygon of the point Pe – from q_u, the primitive corresponding to q_a and q_b is visible due to the absence of any radial primitive such as [Pe q_5] (see text).

Similar approaches are reported in the litterature ([4,5]) from a junction graph whose nodes are the segments and edges specify the connexity (which may be *inexact* in some cases). The goodness measure to be optimized is based on a compactness measure, and possibly, on other criteria that figure the endpoint distance or the degree to which one end of the curve extends beyond the other. Regarding the compactness measure, this property is estimated from a ratio of area of the region bounded by the curve to the area of an approximated shape of the curve, the bounding box in [5], the convex hull in [4]. Nervertheless, the curves defining a symbol are not strictly compact and may contain concavities.

Here, we adopt a measure of goodness issued from the visibility relations extracted beforehand. Let s be a seed curve, c a clique to be merged and $\mathcal{L}$ the operator that enumerates the primitives collected in a set. If $\mathcal{L}(s) \cap \mathcal{L}(c) \neq \emptyset$ then the measure of goodness is defined as follows :

$$goodness(s \cup c) = 1 - \frac{\sum_{i \in Conf(s,c)} length(i)}{\sum_{i \in \mathcal{L}(s) \cap \mathcal{L}(c)} length(i)} \quad (1)$$

where $Conf(s, c)$ enumerates the primitive of $\mathcal{L}(s) \cap \mathcal{L}(c)$ that are conflicting, i.e. a primitive both viewing a primitive l at the left hand side in s [respectively in c] and a primitive r at the right hand side in c [respectively in s].

As an example, we report in the figure 3 the measures of goodness for 3 candidate fusions. The third one is clearly discarded since the primitive shared by the seed curve and the clique is conflictive. On the other hand, the impact of the conflictive primitives for the two other fusions is null.

Thus, the fusion's measure of goodness tends to 0 if primitives shared by the seed curve and the clique are in conflicting uses; tends to 1 otherwise.

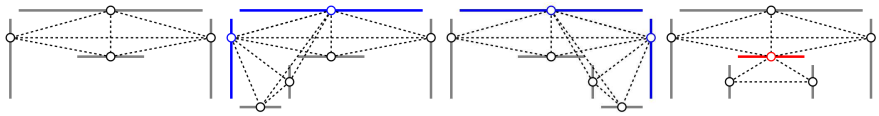


Fig. 3. (a): seed curve (fragmented) ; (b), (c) and (d): Possible fusions and their measure of goodness. A red segment denotes a conflictive primitive contrary to blue segment.

4 Symbol detection

4.1 Region of interest

Symbol spotting consists in localizing possible symbols in complex drawings without necessarily recognizing the symbols themselves, and especially, segmenting them. Such an approach aims to select regions of interest from a similarity measure of two descriptors, and to a certain extend, restrict the possible classes of symbols. Then, a more computational expensive process can be considered for the recognition. Without any *a priori* knowledge on the orientation and size of the objects to be found, dynamic framing windows are constructed in [1] from vector data to get a vectorial signature.

Here, we investigate an indexing procedure of the curves issued from the visibility graph's analysis. In [6], a compact shape representation is presented for retrieving line-patterns from large databases. A relational histograms is built from edges of a neighborhood graph. The resulting descriptor includes relative angle and length ratio attributes. Once a curve within the host VFG is found to be similar to ones of the VFG of at least one symbol's model, the recognition process of such symbol(s) is triggered.

4.2 Extending partial matching

Template matching techniques are rarely used for symbol recognition when dealing with unsegmented and distorted symbols. In [7,8], the efficiency of such a technique is reported regarding identification from flexible frameworks based on symbols' vector representation.

Given a template as an hypothesis and an estimated affine transformation, the template made of vectorial primitives is fitted to the data. Various measures, e.g. distance transform in [7], can then be considered to characterize the accuracy of the involved sumbol's class. In [8], the template can also be distorted during matching, adapting iterately the component's local parameters of the template trough the EM algorithm.

In [9], an error-tolerant subgraph isomorphism algorithm has been proposed for symbol recognition. The underlying representation during the matching process is an attributed cyclic string corresponding to the external polygonalized

boundaries of a region adjacency (sub)graph. A branch and bound approach is driven by boundaries substrings' attributes specified within the model graphs. This growing string approach is suitable for region adjacency graph but the latter is not enough expressive as (i.) it does not permit to model multi-component symbols and (ii.) it only relies on boundaries of loops.

Edit distance is very effective for string, tree and graph based representations provided that the edit operation costs can be determined [10]. Edit operations usually concern substitution, insertion and deletion but can be extended to merging or splitting to handle distortions.

Our basic idea to extend the growing string matching proposed by Lladós et al. is to let the algorithm searching for any *closed curve* (a region) or *nearly closed curve* which is *visible* to curves that belong to the partial matching. Nevertheless, cyclic string are initially employed by Lladós et al to solve the problem of the starting symbol of a closed boundary and to obtain a representation that is invariant under rotation. Managing both closed and nearly closed curves, the use of such an approach make us introduce some virtual segments to fill the gaps of fragmented curves. We always get closed boundaries which can be represented by cyclic strings. Moreover, the full visibility graph of curves falls into a region adjacency graph and the growing string is fully relevant.

The string matching has been also reformulated to take into account both line segments and circular arcs issued from the vectorization process. Indeed, despite the full visibility graph extraction process is described for a segments' set, we can use the arcs of circle initially found during the vectorisation step replacing the list of connected segments by the corresponding arcs. Figure 4 illustrates the alignement of two strings, namely a segment s and an arc a. The initial substitution cost $c(s \rightarrow a)$ is thus modified to introduce a repulsive part that is proportional to the inscribed angle.

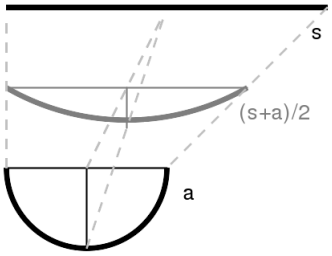


Fig. 4. Alignement of a line segment s and a circular arc a during attributed string edit distance (and the obtained mean primitive)

5 Conclusion

Two facts that motivate our approach should be noticed. First, we do not work on the junction graph since the seek for nearly closed curve cannot safely be achieved while working on it: junction can be missing, leading to a single fragment instead of a junction involving three fragments for example. Moreover, we do not use a distance based approach to define the relation between couple of primitives since it suggests to introduce *a priori* some thresholds which is a tedious task and even impractical for heterogeneous document images. Finally, the perceptually closed curves should be relevant enough to spot symbols within complex document. There, the indexing procedure allows to quickly find similar curves' couple and to trigger error-tolerant subgraph isomorphism detection.
We are not able to evaluate the performance of this system either for identification or localization since this paper describes a work in process. Such elements will be provided during the workshop.

References

1. Rusiñol, M., Lladós, J.: Symbol spotting in technical drawings using vectorial signatures. In: Sixth IAPR Workshop on Graphics Recognition, Hong Kong (2005) 35–45
2. O'Rourke, J.: Computational geometry column 18. SIGACT News **24** (1993) 20–25
3. Wismath, S.: Computing the full visibility graph of a set of line segments. Information Processing Letters **42**(5) (1992) 257–261
4. Saund, E.: Finding perceptually closed paths in sketches and drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(4) (2003) 475–491
5. Zuwala, D., Tabbone, S.: Une méthode de localisation et de reconnaissance de symboles sans connaissance a priori. In: Colloque International Francophone sur l'Écrit et le Document, Fribourg, Suisse (2006) 127–131
6. Huet, B., Hancock, E.R.: Line pattern retrieval using relational histograms. IEEE Transactions on Pattern Analysis on Machine Intelligence **21**(12) (1999) 1363–1370
7. Parker, J.R., Pivovarov, J., Royko, D.: Vector templates for symbol recognition. In: International Conference on Pattern Recognition. Volume 2. (2000) 2602–2605
8. Valveny, E., Martí, E.: Hand-drawn symbol recognition in graphic documents using deformable template matching and a bayesian framework. In: International Conference on Pattern Recognition. Volume 2. (2000) 2239–2242
9. Lladós, J., Martí, E., Villanueva, J.J.: Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(10) (2001) 1137–1143
10. Neuhaus, M., Bunke, H.: Self-organizing maps for learning the edit costs in graph matching. IEEE Transaction on System, Man, Cybernetic. B (2005)