Stable and Robust Vectorization: How to Make the Right Choices

Karl Tombre, Christian Ah-Soon, Philippe Dosch, Gérald Masini, Salvatore Tabbone

Stable and Robust Vectorization: How to Make the Right Choices*

Karl Tombre, Christian Ah-Soon†, Philippe Dosch,
Gérald Masini, Salvatore Tabbone
LORIA
B.P. 239, 54506 Vandœuvre-lès-Nancy CEDEX, France
Karl.Tombre@loria.fr

Abstract

In this paper, we discuss the elements to be taken into account when choosing one's vectorization method. The paper is extensively based on our own implementations and tests, and concentrates on methods designed to have few, if any, parameters.

1 Introduction

Vectorization, i.e. raster–to–vector conversion, has been at the center of graphics recognition problems since the beginning. Despite a lot of efforts, and many proposed solutions—including a lot of commercial software—, we have not yet reached methods which can be considered as sufficiently stable and robust to work as standalone "black boxes". The commercial software packages solve this problem by providing their vectorization method with a number of parameters, adapted to the various categories of drawings to be processed. The user is then in control of the whole process, although families of drawings can be associated with standard sets of parameters.

We believe in another way: one important factor for robustness it to minimize the number of parameters and thresholds needed in the vectorization process [20]. Thus, we have been working on an approach combining several methods, each of which having no or very few parameters, instead of implementing a single method with many parameters. In this paper, we discuss the elements to be taken into account when choosing one's vectorization method. We have no aim at being exhaustive in our coverage of the numerous existing methods, but we concentrate on the main paradigms used, and in many cases on methods we have implemented and tested ourselves; we believe they are representative of the field.

*This work is partly funded by France Telecom CNET.
†Now with Business Objects, Paris.

2 The quality of a vectorization method

For the sake of simplicity and conciseness, we will not discuss the phases which come prior to vectorization, such as binarization, even though there is a clear relationship between the quality of the latter's output and the quality of the vectors obtained. Nor will we consider the recognition and interpretation processes which may run after vectorization, although they often put requirements on the properties of the vectors yielded by the process.

As implied by the name, raster–to–vector conversion consists in analyzing a raster image to convert its pixel representation to a vector representation. The basic assumption is that such a vector representation is more suitable for further interpretation of the image; this typically holds for a scanned graphical document, but may be completely wrong for other kinds of documents.

Thus, the main quality requirement for a vectorization method is that the resulting vectors are well suited to the purpose for which they have been computed. For instance, if the main purpose is to store huge amounts of data in vector format, the total number of vectors often becomes an important quality factor. On the other hand, if high-level symbol recognition is to be performed on the vectors, we may prefer having more vectors, i.e. a lower data compression factor, but higher accuracy for some model-based recognition process to work well. If the application is to convert city maps with individual property limits, the precision with which vectors and junctions are positioned becomes crucial.

This variety of quality factors explains the difficulty in evaluating the performances of vectorization methods. The ideal would be to do some kind of goal-directed evaluation, as has been done for topics such as binarization [21] or thinning [15]. But in graphics recognition, there tends to be as many different goals as there are different projects! Therefore, the recent efforts on performance evaluation for vectorization have taken a different path: providing synthetic images rendered from selected CAD drawings, for which ground truth is available as a set of vectors. Even with this simple and uniform approach, different metrics can still be chosen for measuring the quality of a given vectorization, and as the contest at GREC'97 showed [8], it is not always easy to come out with a clear winner: for instance, is it more important to have well-positioned vectors—even if this means that some ground-truth vectors are split into several vectors in the vectorization result—or to have the same number of vectors as in the ground truth?

In the following discussion, we try to take into account several of these possible criteria, giving the pros and cons of various approaches.

3 Which steps are involved in vectorization?

Whereas a user would tend to look at vectorization as a whole, engineers and researchers involved in designing good raster–to–vector methods know that there are several steps in this process, each with specific quality requirements:

- The first step is to find the lines in the original raster image. Whereas the most common approach for this is to compute the skeleton of the image, a number of other methods have been proposed.

- The next step is to approximate the lines found into a set of vectors. This is performed by some polygonal approximation method, and there are many around, with different approximation criteria.

- After approximation, it is often necessary to perform some post-processing, to find better positions for the junction points, to merge some vectors and remove some others, etc.

- A last step sometimes performed is to find the circular arcs. We will not elaborate on this step in the present paper; a forthcoming paper will describe in details our choices for arc detection.

Of course, we are aware that it is too simplistic to present vectorization as being these four steps applied successively. Many vectorization systems actually perform them in a different order, or merge several of them into a single step—such as finding the lines and approximating them simultaneously, for instance. But we feel it is nevertheless important to discuss separately the criteria for choice of each of these steps.

4 Finding the lines

The first step in vectorization is to process a raster image, supposed to contain graphics[1], in order to extract a set of lines, i.e. of chains of pixels. The most intuitive definition for these lines is probably that they represent the set of significant medial axes of the original image, considered as a shape.

There are three main families of approaches for this step:

1. The first method which comes to mind is to compute the medial axis, i.e. the skeleton of the raster image. This is the most common approach, and skeletons are known to yield good precision with respect to the positioning of the line. But they also tend to give lots of barbs when the image is somewhat irregular, so they need some clever heuristics or post-processing steps, that weaken their generality and robustness. Another weakness of skeleton-based methods is that they displace the junction points, compared to the position wanted by the draftsman.

2. A second family of methods is based on matching the opposite sides of the line. These methods are better at positioning the junction points, but tend to rely too much on heuristics and thresholds when the drawings become complex.

3. A number of sparse-pixel approaches have also been proposed. The general idea is not to examine all the pixels in the image, but to use appropriate sub-sampling methods which give a broader view of the line. One limitation of these methods is that they are prone to “double detections” in some cases.

[1] We suppose in this paper that some text/graphics segmentation method has already been applied to the original image and that we work on the graphics part.

4.1 Skeleton-based methods

The main family of methods for finding the lines is that of computing the skeleton. There are two well-known paradigms for skeletonization methods:

- The first is that of "peeling an onion", i.e. iterative thinning of the original image until no pixel can be removed without altering the topological and morphological properties of the shape. These methods require only a small number of lines in an image buffer at any time, which can be an advantage when dealing with large images. But on the other hand, multiple passes are necessary before reaching the final result, so that computation times may become quite high.

- The second definition used for a skeleton is that of the ridge lines formed by the centers of all maximal disks included in the original shape, connected to preserve connectivity. This leads directly to the use of distance transforms [4], which can be computed in only two passes on the image. However, it is difficult to compute the distance transform without storing the whole image in memory.

In our group, we have been testing both approaches. For thinning, we applied the well-known algorithm illustrated by Fig. 1. This must be followed by a barb removal

0	0	0	1	x	0	1	1	1	0	x	1
x	1	x	1	1	0	x	1	x	0	1	1
1	1	1	1	x	0	0	0	0	0	x	1
x	0	0	x	1	x	x	1	x	0	0	x
1	1	0	1	1	0	0	1	1	0	1	1
x	1	x	x	0	0	0	0	x	x	1	x

Figure 1. Usual structuring elements used for thinning. In each case, the central pixel is set to 0 when the configuration is found (x means don't care). The algorithm consists of successively applying each of these structuring elements on the whole image, and to iterate as long as some pixels are deleted. The 8 elements can be applied in a single pass on the image through pipelining techniques.

procedure, which takes as parameter the maximum number of pixels to remove from free-end chains. The algorithm is straightforward and gives good results, but is very sensitive to noise.

We have also tested with success the use of skeletons computed from distance transforms. To guarantee the precision of the skeleton, we advocate the use of chamfer distances, which come closer to approximating the Euclidean distance. A good compromise between precision and simplicity seems to be the 3–4 chamfer distance transform (see Fig. 2), for which a good skeletonization algorithm has been proposed by Sanniti di Baja [9]. A single threshold on the significance of a branch enables correct removal of the smallest barbs.

After extracting the skeleton, the result of the process is a set of pixels considered as being the medial axes of the lines in the drawing. This set must be linked, to extract the chains making up the lines. Although many applications require some kind of chaining, there are surprisingly few papers or even textbooks giving explanations on the way it is

V_2	V_3	V_4
V_1	X	V_5
V_8	V_7	V_6

First pass : $Y = \min(V_1 + 3, V_2 + 4, V_3 + 3, V_4 + 4)$
Second pass : $Z = \min(Y, V_5 + 3, V_6 + 4, V_7 + 3, V_8 + 4)$

Figure 2. Computing the 3–4 distance transform in two passes over the image, the first from left to right and from top to bottom, the second from right to left and from bottom to top. Sanniti di Baja's algorithm performs some post-processing on this distance transform, such as changing labels 6 to 5 and 3 to 1; see reference [9] for details.

to be done. In algorithm 1, we give some details about a chaining algorithm we have designed, which takes advantage of the topological properties of the skeleton.

4.2 Matching opposite contours

Another family of vectorization methods is based on matching the opposite contours of the lines, working either directly on the contours of the binary image, or on a polygonal approximation of it. We experimented this approach ourselves some years ago, in the REDRAW system [2], and we still use it for thick lines. The basic principle of our algorithm is to work on the contours of the image, and not on the skeleton. The contours can be directly extracted during connected component labeling, and are oriented so that we know on which side the shape is. Polygonal approximation is performed on these contours, and opposite segments are matched; junction points are detected as being located at the intersection of matches involving adjacent segments (see Fig. 3). This method works

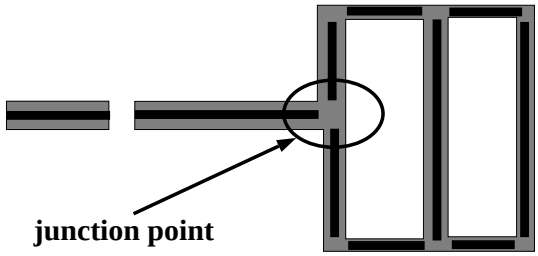

Figure 3. The basic principle of the REDRAW *vectorization method.*

nicely on straight lines; it gives good junction locations and is much less sensitive to noise than skeleton-based methods. However, as illustrated by Fig. 4, it often involves one–to–many or many–to–many matches, and this again leads to hard-to-master thresholds and heuristics in the case of complex drawings, with many curved lines, hatching, etc.

4.3 Sparse-pixel approaches

The paradigm of the third vectorization family is to take a "broader view" of the drawing to be analyzed, by avoiding systematic processing of all the pixels. The best representative of this family is the series of algorithms developed by Dov Dori's team [10, 24]. We designed ourselves a method in this family [22], based on previous work by Lin et al. for

Algorithm 1 Building chains by linking the pixels of a skeleton image.

```
Mark all skeleton pixels of degree ≠ 2, or of degree = 2 when 4 pixels make up a square
Delete isolated pixels
while there are marked points left do
  Choose a marked point p
  Create list l_p of all non-null neighbors of p
  while l_p ≠ ∅ do
    Choose a point q from l_p
    Set p temporarily to 0 to prevent premature looping
    Create a new chain c as [p, q]
    if q is marked then
      continueChain ← false
    else
      continueChain ← true
      Set q to 0
    end if
    while continueChain = true do
      q ← getNonMarked4Neighbor(q)
      if none is found then
        q ← getMarked4Neighbor(q)
        if none is found then
          q ← getNonMarked8Neighbor(q)
          if none is found then
            q ← getMarked8Neighbor(q)
            if none is found then
              q ← getAnyNeighbor(q)
            end if
            continueChain ← false
          end if
        else
          continueChain ← false
        end if
      end if
      Append q to c
      if continueChain = true then
        Set q to 0
      else if we are on a loop then
        Add p to c again to finish the loop
      end if
    end while
    Add c to the list of chains
    Restore mark on p
  end while
  Set p to 0 (it is completely processed)
end while
while there are pixels still unprocessed (they belong to perfect loops) do
  Start over again by marking a non-null pixel
end while
```

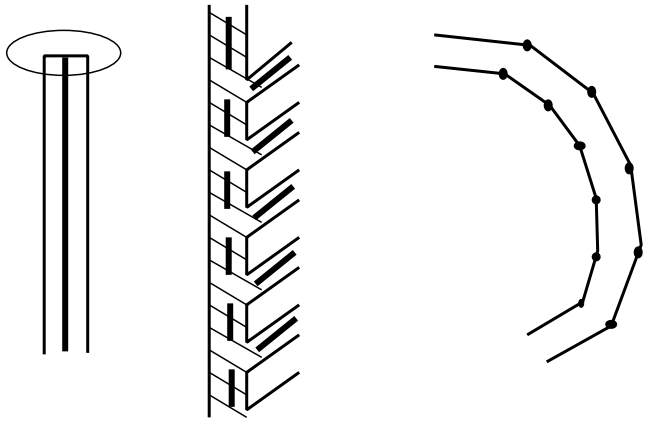

Figure 4. Some of the difficulties with the matching approach: extremity segments must not be matched (this involves a threshold); one–to–many matches lead to complex matching methods; many–to–many matching strategies become necessary for curved lines, which leads to the use of thresholds and/or heuristics.

a diagram understanding system [16]. The principle, illustrated by Fig. 5, is to split up the image into *meshes* and to extract lines by only analyzing the borders of the meshes. The results yielded by the method are good, but the choice of the sub-sampling rate (size of the mesh) is not always straightforward.

4.4 Elements of choice

In his ground-breaking work on edge detection, Canny [5] shows how three sometimes conflicting criteria must be taken into account to design a good edge detection filter. Let us paraphrase these criteria with simple words:

- detection: the filter must detect the true edges and not noise edges;
- localization: the filter must position the edges correctly, with minimal displacement;
- single response: the filter should not detect two edges (because of noise) where there is only one true edge.

Our impression is that, while lacking the same kind of fundamental work on the vectorization problem, our community has designed a number of methods which take these different criteria more or less into account:

- skeleton-based methods have a good localization rate, but tend to be very sensitive to noise, has we have seen;
- contour matching methods have a better detection rate, a relatively good localization factor, but rely on heuristics and complex matching schemes (when the drawing is not simple) which do not guarantee a single response in all cases;

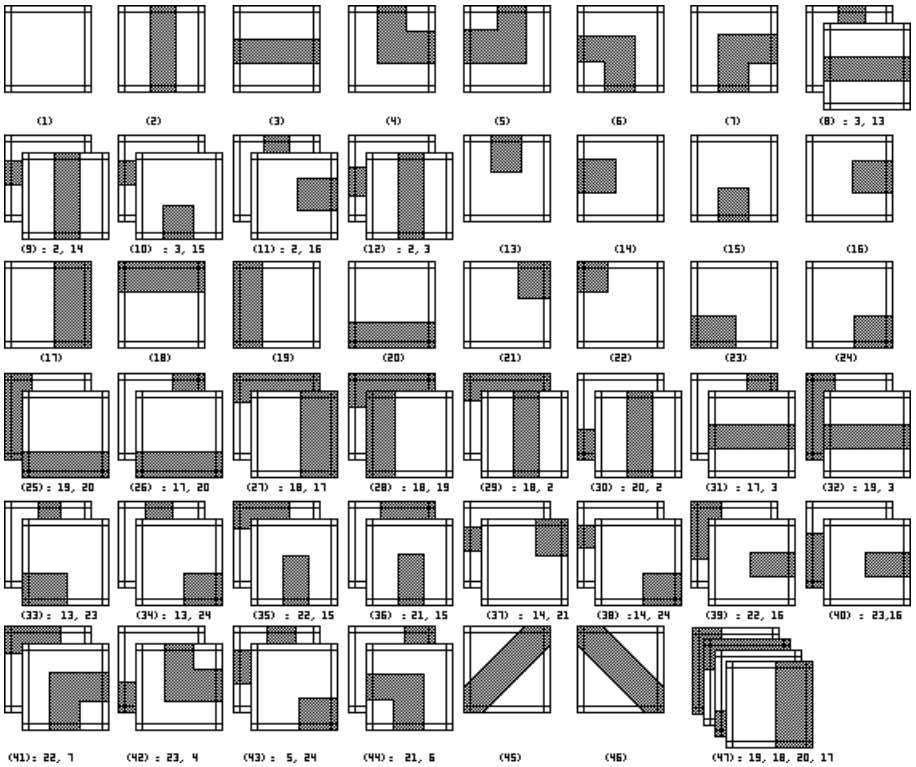

Figure 5. Principle of our sub-sampling method [22].

- sparse-pixel approaches work surprisingly well with respect to localization, but some of them (our sub-sampling method, for instance) may miss too small details, and our experiments show that they are also prone to double responses in some cases.

In addition, the correct positioning of junctions is often very important in our applications. All skeleton-based methods are weak with respect to the correct restitution of the junction at the location the draftsman wanted it to be. This is a direct consequence of

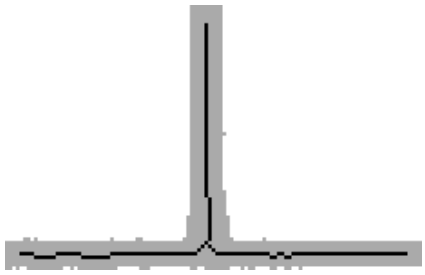

Figure 6. Position of the junction point with a skeleton-based method.

the fact that the skeleton follows the medial axis of the shape, whereas the position of the junction as envisioned by the draftsman is *not* on the medial axis of the shape (see Fig. 6). Contour matching methods are much better in positioning the junctions; this explains why some authors propose combined methods, such as using the skeleton for positioning of the

lines and contour matching to reduce noise and to have better junctions [12]. In sparse-pixel approaches, the junctions are also relatively well detected, but as all pixels are not explored, some methods tend to find junctions where there were none, as we experienced in our mesh-based method (see Fig. 7).

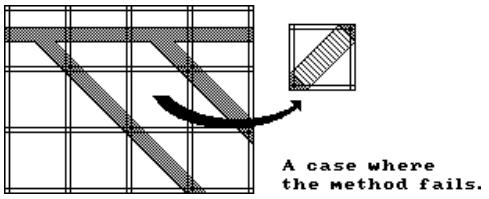

Figure 7. False detection of a junction in the mesh-based method [22] : as only the borders of the meshes are explored, the method falsely assumes that there is a junction between the two hatching lines.

Table 1 summarizes our elements of choice, comparing the different families of methods. We are aware that our ratings may be subjective, especially as we try to factorize the

	Skeleton	Contour matching	Sparse-pixel
Localization	++	+	+
Detection	–	+	+/-
Single response	++	+/-	–
Junctions	–	+	+/-

Table 1. Elements of choice.

behavior of large families of methods, which is not always possible without being unfair to some specific algorithm.

5 From lines to segments

The lines extracted from the previous step are to be represented by a set of segments. This is done by polygonal approximation. Here also, many methods exist. The first level for classifying these methods into different families is that of the criterion used to decide that a curve can be approximated by a line segment.

- The most usual criterion is that of the maximum distance between the curve and the segment. This leads to recursive splitting of the curve at the maximum deviation points, until all segments are valid approximations. As the position of the extrema points of the segments tends to be constrained by the initial pixel positions, it may make sense for the method to be followed by a fitting phase, where each segment is displaced to best fit the original curve. However, in the case of vectorization, this leads to the additional problem of keeping correct junctions while fitting each segment to the curve.

- A second possible criterion is that of the algebraic area of the surface between the curve and the segment. As this area can be computed iteratively, as the sum of successive triangles, very time-efficient iterative methods can be implemented for this approach. A problem with the approach is that it tends to displace the angular points, as the method only detects a change of general direction after having followed several points past the true angle.

- Other criteria include various angular measures [11] and curvature computations [3].

In our experiments, we have used two methods, representing the two first families mentioned above. For the recursive splitting approach, we have implemented Rosin and West's recursive split-and-merge method [19], which has the advantage that it does not require any user-given threshold or parameter. The principle is to recursively split the curve into smaller and smaller segments, until the maximum deviation is 0 or there are only 3 or less points left. Then, the "tree" of possible segments is traversed and the method keeps those segments maximizing a measure of significance, which is defined as a ratio between the maximum deviation and the length of the segment. Recently, Rosin has proposed other possible measures of significance [18], and we plan to explore this further in the coming months.

We have also used for many years an iterative method, that of Wall and Danielsson [23], enhanced in our team with a direction-change marking procedure to preserve the angular points. The method only needs a single threshold, on the ratio between the algebraic surface and the length of the segments. It is fast and efficient, but not as precise as the former. On the other hand, Rosin and West's method tends to split up the lines around a junction into too many small segments (see Fig. 8). This is a direct consequence

(a) Wall & Danielsson [23] with typical threshold = 20.

(b) Rosin & West [19].

Figure 8. Comparison of two polygonal approximation methods applied to Fig. 6.

of its preciseness and of the previously mentioned displacement of junction points by skeleton-based methods.

Based on these experiments, and depending on the ultimate goals for the graphics recognition process, we would recommend the following choice for the polygonal approximation method to be used:

- If the simplicity of the resulting set of vectors is important, the best choice is probably an iterative method like Wall & Danielsson's. It will give a number of segments

closest to the number in the original drawing. However, it is not optimal with respect to positioning of these segments.

A possible improvement of the method would be to perform several approximations with varying thresholds, to have some kind of "multi-scale" approximation, and to merge the results in order to optimize a $\frac{precision}{number}$ ratio.

- If the precision is the most important criterion, Rosin & West's method seems to us to be a good choice. An additional advantage of the method is that it does not require any explicit threshold or parameter. However, our experience is that symbol recognition processes tend to be disturbed by the symbol being split up into too many segments [1]. Also, as already said, the approximation will be precise *with respect to the original curve*, which does not necessarily correspond to the expected line, as we have seen with the displacement of junctions in a skeleton.

  Possible improvements include better significance criteria [18] and the use of post-processing steps (§ 6).

6 Post-processing

In the two steps described until now, we have never used any explicit knowledge about what vectors are supposed to be in a graphical document. The line finding methods working on the binary image only use simple topological and photometric rules to somehow find the medial axes in the image. As we have seen with skeletons, this can lead to results which are contrary to what the draftsman expects. The same can be said of the approximation algorithms we have described, which only use some deviation criteria to compute an approximation of the original curve, without making any assumption about what this curve is supposed to represent. Actually, the same approximation algorithms are also used in very different contexts, such as object recognition problems in computer vision, for instance.

Therefore, it is often necessary to add contextual knowledge at some stage of the vectorization process. For the sake of simplicity, we call this post-processing, although we are aware that some authors include contextual constraints throughout their vectorization process.

As the following brief survey shows, many different ideas have been proposed for adding this kind of domain knowledge to the vectorization process:

- When processing large number of very specific documents, it may make sense to develop a completely *ad hoc* vectorization system. For instance, Chhabra et al. have developed efficient methods for finding straight lines in telephone company drawings containing a lot of large tables [7]. In such a case, the direct recognition of the longest straight lines solves all the junction problems, as the junctions are simply the intersections of the straight lines found.

- It is also possible to add constraints, describing the "ideal" geometry of the result, to the vectorization process itself. This was proposed by Röösli & Monagan [17] at GREC'95.

- Several authors use general vectorization methods and propose a set of simple heuristics to correct the result, setting junctions straight, merging those which are close to each other, reconnecting lines split up by a missing pixel, etc. One of the best and most recent examples of such a system is that of Chen et al. [6]. These systems yield good results, but as they rely on heuristics, they tend to introduce a number of additional thresholds and parameters, which is contrary to the aims we have set ourselves.

- In our opinion, the most promising path for post-processing in a vectorization process, while remaining as generic and robust as possible, is that of introducing models of the "ideal" junctions (T junctions, L junctions, X junctions, Y junctions . . .) and correcting each junction by fitting one of these models to it. Janssen proposed a similar system based on morphological processing of the junction areas [13]. We are currently exploring various fitting methods, hoping to report very soon on promising results in that area.

7 Conclusion

Without aiming at presenting a complete state of the art of vectorization methods, we have tried in this paper to explore the main criteria for choosing a robust vectorization method, and the most common paradigms used in the different steps involved. As it seems difficult to provide a universal measure for assessing the performances of vectorization, we believe that the elements of choice given can be complementary to statistical performance evaluation processes.

References

[1] C. Ah-Soon and K. Tombre. Network-Based Recognition of Architectural Symbols. In A. Amin, D. Dori, P. Pudil, and H. Freeman, editors, *Advances in Pattern Recognition (Proceedings of Joint IAPR Workshops SSPR'98 and SPR'98, Sydney, Australia)*, volume 1451 of *Lecture Notes in Computer Science*, pages 252–261, August 1998.

[2] D. Antoine, S. Collin, and K. Tombre. Analysis of Technical Documents: The REDRAW System. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 385–402. Springer-Verlag, Berlin/Heidelberg, 1992.

[3] H. Asada and M. Brady. The Curvature Primal Sketch. *IEEE Transactions on PAMI*, 8(1):2–14, 1986.

[4] G. Borgefors. Distance Transforms in Digital Images. *Computer Vision, Graphics and Image Processing*, 34:344–371, 1986.

[5] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on PAMI*, 8(6):679–698, 1986.

[6] Y. Chen, N. A. Langrana, and A. K. Das. Perfecting Vectorized Mechanical Drawings. *Computer Vision and Image Understanding*, 63(2):273–286, March 1996.

[7] A. K. Chhabra, V. Misra, and J. Arias. Detection of Horizontal Lines in Noisy Run Length Encoded Images: The FAST Method. In Kasturi and Tombre [14], pages 35–48.

[8] A. K. Chhabra and I. T. Phillips. The Second International Graphics Recognition Contest—Raster to Vector Conversion: A Report. In K. Tombre and A. K. Chhabra, editors, *Graphics Recognition—Algorithms and Systems*, volume 1389 of *Lecture Notes in Computer Science*, pages 390–410. Springer-Verlag, April 1998.

[9] G. Sanniti di Baja. Well-Shaped, Stable, and Reversible Skeletons from the (3,4)-Distance Transform. *Journal of Visual Communication and Image Representation*, 5(1):107–115, 1994.

[10] D. Dori, Y. Liang, J. Dowell, and I. Chai. Sparse-Pixel Recognition of Primitives in Engineering Drawings. *Machine Vision and Applications*, 6:69–82, 1993.

[11] J. G. Dunham. Optimum Uniform Piecewise Linear Approximation of Planar Curves. *IEEE Transactions on PAMI*, 8(1):67–75, 1986.

[12] O. Hori and A. Okazaki. High Quality Vectorization Based on a Generic Object Model. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*, pages 325–339. Springer-Verlag, Heidelberg, 1992.

[13] R. D. T. Janssen and A. M. Vossepoel. Adaptive Vectorization of Line Drawing Images. *Computer Vision and Image Understanding*, 65(1):38–56, January 1997.

[14] R. Kasturi and K. Tombre, editors. *Graphics Recognition—Methods and Applications*, volume 1072 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1996.

[15] L. Lam and C. Y. Suen. An Evaluation of Parallel Thinning Algorithms for Character Recognition. *IEEE Transactions on PAMI*, 17(9):914–919, September 1995.

[16] X. Lin, S. Shimotsuji, M. Minoh, and T. Sakai. Efficient Diagram Understanding with Characteristic Pattern Detection. *Computer Vision, Graphics and Image Processing*, 30:84–106, 1985.

[17] M. R¨o¨osli and G. Monagan. Adding Geometric Constraints to the Vectorization of Line Drawings. In Kasturi and Tombre [14], pages 49–56.

[18] P. L. Rosin. Techniques for Assessing Polygonal Approximation of Curves. *IEEE Transactions on PAMI*, 19(6):659–666, June 1997.

[19] P. L. Rosin and G. A. West. Segmentation of Edges into Lines and Arcs. *Image and Vision Computing*, 7(2):109–114, May 1989.

[20] K. Tombre, C. Ah-Soon, Ph. Dosch, A. Habed, and G. Masini. Stable, Robust and Off-the-Shelf Methods for Graphics Recognition. In *Proceedings of the 14th International Conference on Pattern Recognition, Brisbane (Australia)*, pages 406–408, August 1998.

[21] Ø. Due Trier and A. K. Jain. Goal-Directed Evaluation of Binarization Methods. *IEEE Transactions on PAMI*, 17(12):1191–1201, December 1995.

[22] P. Vaxivière and K. Tombre. Subsampling: A Structural Approach to Technical Document Vectorization. In D. Dori and A. Bruckstein, editors, *Shape, Structure and Pattern Recognition (Post-proceedings of IAPR Workshop on Syntactic and Structural Pattern Recognition, Nahariya, Israel)*, pages 323–332. World Scientific, 1995.

[23] K. Wall and P. Danielsson. A Fast Sequential Method for Polygonal Approximation of Digitized Curves. *Computer Vision, Graphics and Image Processing*, 28:220–227, 1984.

[24] L. Wenyin and D. Dori. Sparse Pixel Tracking: A Fast Vectorization Algorithm Applied to Engineering Drawings. In *Proceedings of the 13th International Conference on Pattern Recognition, Vienna (Austria)*, volume 3, pages 808–812, August 1996.