Vectorization in Graphics Recognition: To Thin or not to Thin

Karl Tombre
LORIA–INPL*
tombre@loria.fr

Salvatore Tabbone
LORIA–Université Nancy 2*
tabbone@loria.fr

Abstract

Vectorization, i.e. raster–to–vector conversion, is a central part of graphics recognition problems. In this paper, we discuss the pros and the cons of basing one's vectorization process on skeletonization. While distance skeletons have proven to be robust and precise, they tend to distort the results at line extremities and junctions. In these cases, contour-matching approaches yield better results, but they have their own specific problems. A perspective is probably to combine the best of both methods.

1. Introduction

Graphics recognition is concerned with the analysis of graphics-intensive documents, such as technical drawings, maps or schemas. Vectorization, i.e. raster–to–vector conversion, is of course a central part of graphics recognition problems, as it deals with converting the scanned image to a vector form suitable for further analysis. Many vectorization methods have been designed throughout the years, and a number of software packages are available. Thus, in one sense, the basic raster–to–vector conversion problem might be considered to be solved. However, there is still a major problem of precision, robustness and stability of the vectorization processes. This has been and still is a matter of continuing research for our team [22, 23].

From the earliest works on vectorization, one of the main issues was whether or not to base the method on some kind of medial axis or skeleton computation. It is of course quite natural for image processing specialists to think of raster–to–vector conversion as a typical application for skeletonization. Indeed, a majority of systems use some kind of skeleton computation as a central part of their process. However, one must not forget that while a skeleton is meant to represent faithfully the analyzed binary shape, it does not always provide the best means to retrieve the actual thought of the draftsman. Typically, the way junctions and line extremities are processed by a skeletonization process is often very different from the way the user wants them to be processed. This leads to the introduction of a number of heuristics in the process, reducing the robustness of the method, or to costly manual editing.

Many other ideas have therefore been explored, with the aim of extracting the lines without explicitly computing the medial axis. In this paper, we will not review exhaustively all these ideas, but we propose a discussion about the pros and cons of using or not a skeleton as the central part of vectorization.

2. The problem

As implied by the name, raster–to–vector conversion consists in analyzing a raster image to convert its pixel representation into a vector representation. 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 [23].

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 of these around as well, using 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 discuss this in another paper at the conference [9].

In this paper, we will mainly concentrate on the choice of methods for the first step, that of finding the lines. But of course, as the choice of the polygonal approximation method also has an influence on the whole process, we will briefly discuss our choices for that step. Among the many available methods, we have chosen to use two methods. The first is Rosin and West's recursive split-and-merge method [20], which has the advantage that it does not re-

*Common address: LORIA, B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex, France

quire 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.

We have also used for many years an iterative method, that of Wall and Danielsson [24], 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 [23].

3. How to Find the Lines

To find the lines, we must 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. As explained in § 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 and extremity points, compared to the position wanted by the draftsman.

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.

A number of sparse-pixel approaches have also been proposed. The general idea is to avoid having to examine all the pixels in the image, by using 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.

The first family is the most common choice. In this paper, we discuss its advantages and disadvantages, comparing it with a method which retrieves directly the lines from the image, using contour matching.

[1] If necessary, some kind of text/graphics segmentation [11] must of course be applied to the original image before vectorization.

4. 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 [17]. 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 or similar measures [2, 7, 18], which can be computed in only two passes on the image.

In our group, we have been testing both approaches. The iterative thinning algorithm is straightforward and gives good results, but is very sensitive to noise. We therefore prefer to use 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. 1), for which a good skeletonization algorithm has been proposed by Sanniti di Baja [8]. A single threshold on the significance of a branch enables correct removal of the smallest barbs. The latter method is our preferred choice, it yields stable skeletons in an efficient way, and prunes reasonably well the small barbs.

The correct positioning of junctions is often very important in graphics recognition 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 the fact that the skeleton follows the centers of the maximal discs of the pattern, whereas the position of the junction as envisioned by the draftsman is *not* on these centers (see Fig. 2).

5. Matching Opposite Contours

Another family of vectorization methods, often proposed, relies on matching the opposite contours of the line [1, 4, 10, 12, 14]. They perform better with respect to positioning the junctions. Generally, these approaches are based on four main steps: extraction of the contour vectors,

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 1. Computing the 3–4 chamfer 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. See reference [8] for details.

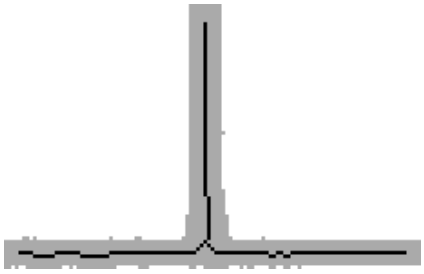


Figure 2. Position of the junction point with a skeleton-based method.

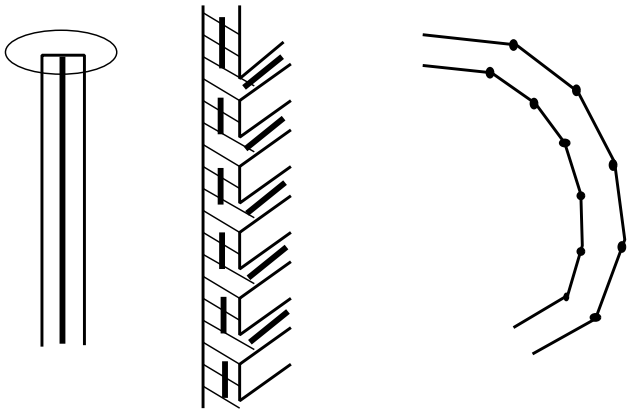


Figure 3. Some of the difficulties with the matching approach: extremity segments, one–to–many and many–to–many matchings.

matching of these vectors, generation of a medial line and junction processing (filling the gaps left by the matching process). The aim of the two first steps is to decompose the binary shape into slices ("blocks" in [10]), thus segmenting the graphical shape into regions, which themselves connect junctions. To extract and to bridge the medial line, the connection information obtained during the matching process is used. For example, an adjacency graph is built in [12]. In most cases, the medial lines yielded by matched contours can be extended until their intersections are found.

The main drawback of these approaches comes from the difficulty of correctly handling non-matches and mismatches between contours, and in the complexity of one-to-many and many-to-many matches (Fig. 3). This leads to hard-to-master thresholds and heuristics in the case of complex drawings. Some authors propose combined methods, such as using the skeleton for positioning of the lines and contour matching to reduce noise [13].

Furthermore the matching criteria are based on orientation similarities between contours vectors. That is, the method fails when the binary pattern is not symmetric.

To improve the matching procedure, we have recently suggested [21] to use the gradient orientation defined by Canny's operator [3]:

$$G_\sigma(x,y) = -\frac{1}{2\pi\sigma^2} exp(-\frac{x^2+y^2}{2\sigma^2}),$$

where G is the 2D Gaussian filter and σ the standard deviation. The direction θ at each point is:

$$\theta_{(x,y)} = \tan(\frac{\partial_y(I(x,y) * G)}{\partial_x(I(x,y) * G)})^{-1}$$

where $*$ denotes convolution. It is well known that the gradient vector is perpendicular to the contour. Thus, each contour point is matched using the orientation θ. In this case the match is dense and robust, as it relies on contour points and the contours are smoothed with the Gaussian filter. Consequently, mismatches don't affect the global qualities of the results. Figure 4.c shows the application of the method.

6. Contextual Information

Whereas the line finding methods discussed until now are independent of any a priori information about the nature of the drawing, it is often necessary to add contextual knowledge at some stage of the vectorization process. For some applications, this kind of knowledge favors a given method. For instance, the vector-matching approach described by Chang [4] gives good results on Chinese characters, as the vectorial characteristics of Chinese characters are known and integrated in the process.

Many different ideas have been proposed for adding this kind of domain knowledge to the vectorization process. When processing a 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 [6]. 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.

Several authors prefer to 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. [5]. These systems yield good results, but as they rely on heuristics, they tend to introduce a number of additional thresholds and parameters.

When we look for something specific, such as walls in architectural drawings, it may be very interesting to use contour-matching methods. Here, the contextual knowledge is the fact that the walls are basically horizontal or vertical rectangles. The underlying idea of the method is to reconstruct the 2D geometry of the graphics. To do this, a broad skeleton is extracted from the graphics using the point-to-point matching approach described in § 5. Each skeleton fragment represents the signature of a 2D primitive (in our case a rectangle). We do not use this "skeleton" as the basis for polygonal approximation, but solely for reconstruction of the corresponding rectangles, which correspond to the walls. Figure 4 illustrates this approach on a simple architectural drawing. Figures 4.b and 4.c show the extracted contours and the signature of the broad skeleton. We can observe that the contours are clean and well located. The circumscribed rectangles in the horizontal and vertical directions are presented in Figure 4.d, and the extracted medial lines of the rectangles in Figure 4.e. For the sake of comparison, Fig 4.f shows the result of a skeleton-based vectorization, using the 3–4 distance transform (§ 4 and a polygonal approximation.

7. Discussion

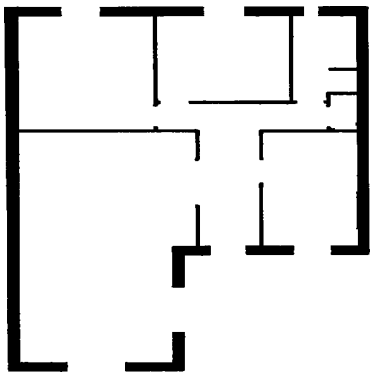
As the wall detection example described in the previous section demonstrates, contour matching associated with contextual knowledge can yield much better results than the usual skeletonization approaches. However, in the general case, it is not possible to associate a priori knowledge with all types of vectors to be extracted. Given the complexity and lack of stability of contour matching methods in the general case, we still believe that despite its weaknesses, the skeletonization approach is more general and robust. Whatever the graphics, the qualities of the results are always good, especially with a good skeletonization algorithm like the 3–4 distance transform method explained in § 4.

However, these methods, working on the binary image, only use simple topological, geometric and photometric rules to somehow find the skeleton in the image. As we have seen with skeletons, this can lead to a lack of precision around the junctions. In this case, we need to post-process the image in order to improve the detection and the localization of the junctions. 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 [15]. 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 [19]. We are currently exploring various fitting methods, hoping to report soon on promising results in that area.

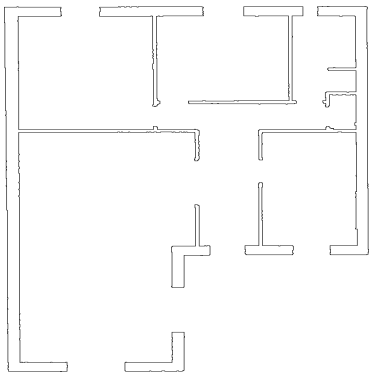
As illustrated by the example of Fig. 4, there are also certainly ideas from the contour-matching approach which can be reused in a more general framework. For instance, when we take a broader view of the respective merits of the two approaches, we can say that skeleton-based methods emphasize the *distance* criterion (finding the points which are centers of maximum enclosed circles) whereas the point-to-point matching approach described in § 5 emphasizes the *direction* criterion. If a first "crude" vectorization could segment a drawing into pieces which have the same direction, we could use directional distance transforms on each of these pieces, and this would eliminate the problems at the line extremities and simplify junction processing. Such a two-step vectorization is also currently under study in our group.

8. Conclusion

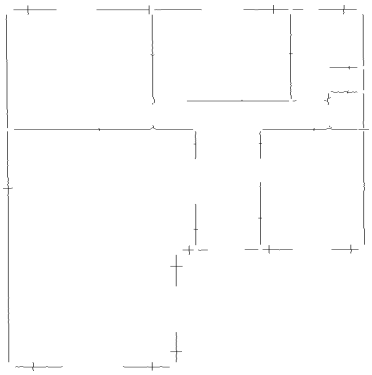
In this paper, we have discussed the respective qualities of skeletons and contour matching methods for building a robust vectorization method. Of course, we also need to follow the evolution in neighboring fields, where new methods may emerge. Even if they must still be considered to be too experimental or too computationally expensive to be applicable for real-size graphics recognition problems, they may mature in the coming years to propose alternatives to the current techniques used for computing a "skeleton" or for approximating a line. This includes ideas like the use of energy minimization techniques to compute the medial axis of shapes; for instance, a recent paper proposes a statistical framework based on Markov random fields, which yields surprisingly good results on simple shapes [25].



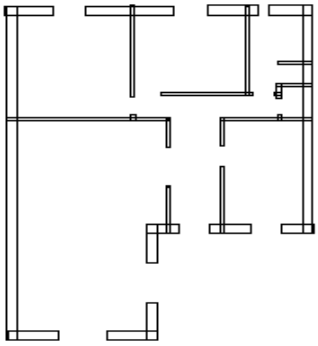
(a) Original image.



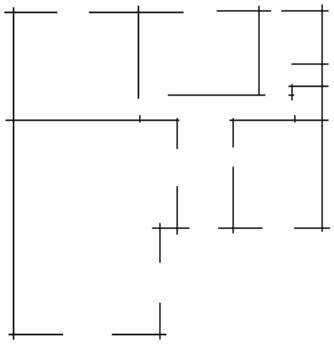
(b) Contour detection.



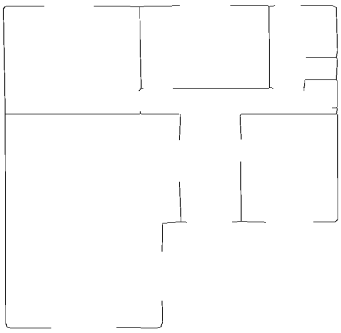
(c) Broad skeleton extraction.



(d) Reconstruction into 2D primitives.



(e) Medial lines of the 2D primitives.



(f) Classical skeletonization followed by polygonal approximation.

Figure 4. Contour matching method compared with usual skeletonization.

References

[1] 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.

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

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

[4] F. Chang, Y.-C. Lu, and T. Pavlidis. Feature Analysis Using Line Weep Thinning Algorithm. *IEEE Transactions on PAMI*, 21(2):145–158, Feb. 1999.

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

[6] 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 [16], pages 35–48.

[7] J. Y. Chiang, S. C. Tue, and Y. C. Leu. A New Algorithm for Line Image Vectorization. *Pattern Recognition*, 31(10):1541–1549, Oct. 1998.

[8] G. S. 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.

[9] P. Dosch, G. Masini, and K. Tombre. Improving Arc Detection in Graphics Recognition. In *Proceedings of the 15th International Conference on Pattern Recognition, Barcelona (Spain)*, Sept. 2000. To appear.

[10] K.-C. Fan, D.-F. Chen, and M.-G. Wen. Skeletonization of Binary Images with Nonuniform Width via Block Decomposition and Contour Vector Matching. *Pattern Recognition*, 31(7):823–838, July 1998.

[11] L. A. Fletcher and R. Kasturi. A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images. *IEEE Transactions on PAMI*, 10(6):910–918, 1988.

[12] C.-C. Han and K.-C. Fahn. Skeleton Generation of Engineering Drawings via Contour Matching. *Pattern Recognition*, 27(2):261–275, 1994.

[13] 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.

[14] O. Hori and S. Tanigawa. Raster-to-Vector Conversion by Line Fitting Based on Contours and Skeletons. In *Proceedings of 2nd International Conference on Document Analysis and Recognition, Tsukuba (Japan)*, pages 353–358, 1993.

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

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

[17] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning Methodologies — A Comprehensive Survey. *IEEE Transactions on PAMI*, 14(9):869–885, Sept. 1992.

[18] C. W. Niblack, P. B. Gibbons, and D. W. Capson. Generating Skeletons and Centerlines from the Distance Transform. *CVGIP: Graphical Models and Image Processing*, 54(5):420–437, Sept. 1992.

[19] M. Röösli and G. Monagan. Adding Geometric Constraints to the Vectorization of Line Drawings. In Kasturi and Tombre [16], pages 49–56.

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

[21] S. Tabbone and L. Wendling. Décomposition graphique sous forme de primitives 2D. In *Actes du 2ème Colloque International Francophone sur l'Écrit et le Document, Lyon (France)*, July 2000. To appear.

[22] K. Tombre, C. Ah-Soon, P. 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, Aug. 1998.

[23] K. Tombre, C. Ah-Soon, P. Dosch, G. Masini, and S. Tabbone. Stable and Robust Vectorization: How to Make the Right Choices. In *Proceedings of 3rd International Workshop on Graphics Recognition, Jaipur (India)*, pages 3–16, Sept. 1999. Revised version to appear in a forthcoming LNCS volume.

[24] 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.

[25] S.-C. Zhu. Stochastic Jump-Diffusion Process for Computing Medial Axes in Markov Random Fields. *IEEE Transactions on PAMI*, 21(11):1158–1169, Nov. 1999.