Improving the Accuracy of Skeleton-Based Vectorization

Xavier HILAIRE[1,2] and Karl TOMBRE[1]
[1] LORIA, B.P. 239, 54506 Vandœuvre-ls-Nancy, France
[2] FS2i, 8 impasse de Toulouse, 78000 Versailles, France

June 21, 2001

Abstract

In this paper, we present a method for correcting a skeleton-based vectorization. The method robustly segments the skeleton of an image into basic features, and uses these features to reconstruct analytically all the junctions. It corrects some of the topological errors usually brought by polygonal approximation methods, and improves the precision of the junction points detection.

We first give some reminders on vectorization and explain what a good vectorization is supposed to be. We also explain the advantages and drawbacks of using skeletons. We then explain in detail our correction method, and show results on cases known to be problematic.

1 Introduction

Vectorization is the process which automatically converts a raster image to a set of graphical primitives such as vectors and arcs. These primitives are assumed to be those which made up the drawing when it was drafted; therefore, vectorization can be considered as some kind of "reverse engineering" process.

An "ideal" vectorization system should therefore be able to yield vectorial data as close to the "original" as possible, in number as well as in position. Unfortunately, this is not always the case, for three main reasons:

1. the images are often disturbed (printing defects, folds, smears, etc.) and noisy (scanning as well as binarization noise), and have sometimes visible distortions due to the mechanical scanning process, or to skew;

2. there is not necessarily a unique vectorial solution for the input data: two different sets of vectors may generate the same set of pixels;

3. all known vectorization processes have some imperfections of their own, which means that errors are introduced by the process.

At GREC'99, we discussed some qualitative elements which should be taken into account when choosing the different steps of one's vectorization method [6]. We emphasized the problem of being able to position both the line segments and the junctions between them with sufficient precision. We pointed out that despite its weaknesses, we still consider *skeletonization* of a binary image to be the best compromise; indeed, it is a widely used technique for vectorization. However, it may introduce topological as well as numerical distortions. For the topology, the number of detected junction points may be too low (Fig. 1,b1) or too high (Fig. 1,b2).

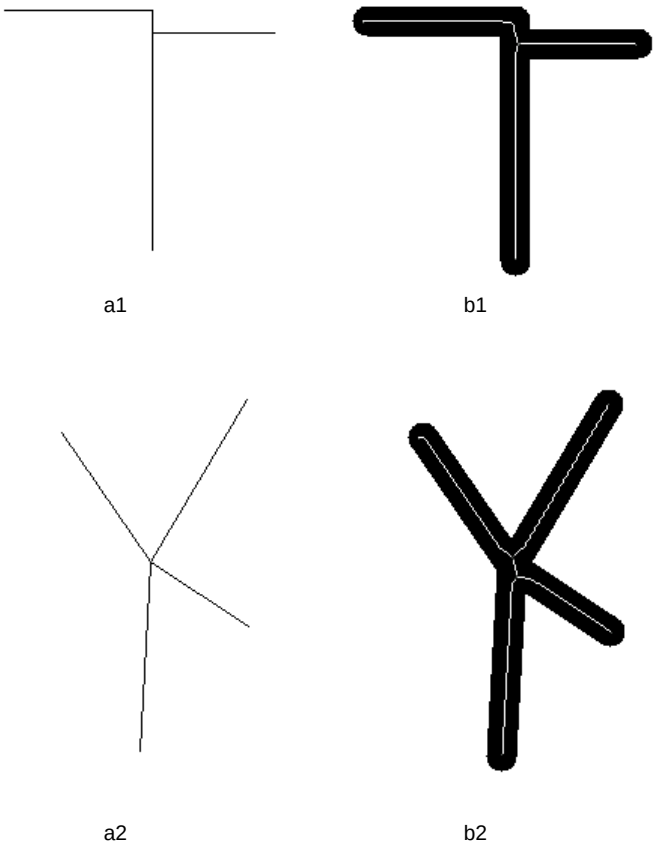

Figure 1: Topological errors of the skeletonization. a1, a2: original vectors (without their thickness); b1, b2: skeletons of bitmaps obtained from these vectors (with their thickness).

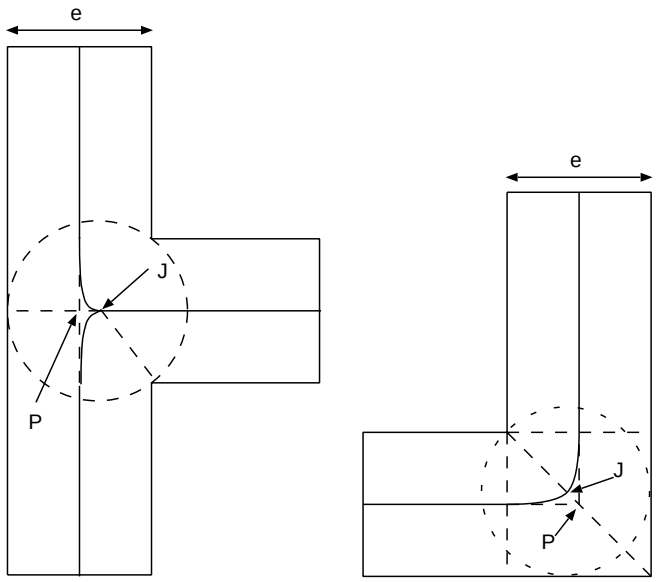

Figure 2: Distortions introduced by skeletonization. Expected points: P. Points yielded by skeletonization: J.

There may also be numerical distortions in the areas where the geometrical features cover each other; one consequence is that the detected junction points are not located where they are expected to be. In the two most common cases, those of L and T junctions, the distortion is parabolic, and it is easy to demonstrate (Fig. 2) that the distance between the actual position and the expected position increases with the thickness e of the line: $d(P,J) = \frac{1}{8}e$ for a T junction, and $d(P,J) = \left(\frac{3\sqrt{2}}{2} - 2\right) e$ for a L junction.

2 Our method

The points raised in the previous section have led us to propose a vectorization method which is still based on the skeleton, but which is able to correct the errors introduced by polygonal approximation and to improve the precision of the junction points.

The basic point is that the skeleton is a good descriptor of the median axis for elongated and isolated shapes, but is a poor descriptor for their intersections. The idea is therefore not to take into consideration those parts of the skeleton which are included in the overlapping between two or more shapes. Once these parts are eliminated, we perform an *analytical* reconstruction of each junction from the remaining parts of the skeleton. In summary, our method has the following steps:

1. *Layer separation*—Separate the graphical drawing into homogeneous thickness layers.

2. *Skeletonization*—Compute the skeleton of the binary image.

3. *Segmentation*—Segment the skeleton into straight segments and circular arcs, and eliminate the geometric primitives which are shorter than the thickness e of the processed layer. Build a graph $\mathcal{G}$ whose edges are the primitives and whose nodes e_i are the connection points between them.

4. *Instantiation*—Define the support equation for each primitive, through linear regression, with the associated uncertainty domain.

5. *Dual correction*—For all nodes e_i of $\mathcal{G}$ do

   (a) Build the set E_0 of primitives stemming from e_i

   (b) If there is another node than e_i in $\mathcal{G}$
   - Let e_j be the node closest to e_i (Euclidean distance)
   - Add to E_0 all the primitives stemming from e_j

   (c) Build the successive subsets $E_1, E_2, ..., E_{n-1}$, using the following algorithm, starting with $u = 0$:

   i. Construct F as the biggest subset of E_u for which all primitives do intersect. Let e'_u be the intersection node.

   ii. If $F \neq E_u$ then
   - Set E_{u+1} to $E_u \setminus F$
   - Set E_u to F
   - Increment u and loop at 5(c)i

   (d) For $u = 0, \ldots, n-1$, replace the extremity of each primitive of E_u stemming from e_i or e_j by e'_u

   (e) Connect all the e'_u

6. Repeat step 5 as long as the global topology is modified by the process.

We will now describe each of these steps in detail.

2.1 Layer separation

This is an important step for the method we propose, as we need to work on image layers with lines of homogeneous thickness. Simple mathematical morphology methods can be used to get these different layers, as we have explained in previous papers [5].

2.2 Skeletonization

As we explained in detail at the last GREC, we use Sanniti di Baja's skeletonization algorithm, based on the 3–4 distance transform, to compute the skeleton [1]. The extracted skeleton is chained using the algorithm described in [6].

2.3 Segmentation

The first difference with the method we have used until now in our group [6] is that we do not perform a polygonal approximation immediately after the skeletonization. The various polygonal approximation methods can of course convert a chain of pixels to a curve in a very efficient way, but they disturb the analytical junction reconstruction algorithm we propose. The method proposed by Rosin and West [4] yields an excellent approximation of the original curve, but is very sensitive to noise and to the choice of starting points, as illustrated by Fig. 3: if the starting points are A and D, the splitting phase introduces a cutting point R which leads to splitting $[BC]$ into two segments. If A is displaced towards A', the cutting point R becomes close to a point R' which is neighbor of C, and the decomposition of $[BC]$ is thus much better, but a new problem may be introduced on $[CD]$.

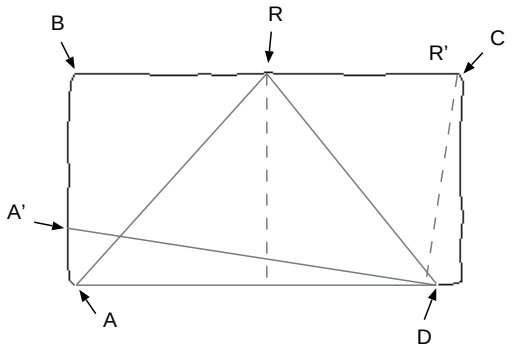

Figure 3: Sensitivity to noise of the cutting point in the Rosin and West approximation.

The method proposed by Wall and Danielsson [8] does not have this drawback, but it is not very scalable, which makes circular arcs detection, among others, difficult: for a circle, the angle α from the center of the first cutting point, with respect to the starting point, is dependent on the radius R of the circle, as the polygonal approximation uses a fixed threshold K, and we have the relation $R(1 - \frac{\sin\alpha}{2\alpha}) = K$. Thus, a small circle will be approximated by fewer segments than a bigger circle, and vector-based circle detection methods, such as the one developed in our group [2], will be less successful.

We therefore segment the skeleton using another method. First, we note that the anchor points[1] are good cutting points for separating primitives, as they correspond to their meeting points. These anchor points are located in the zones where the skeleton is "distorted", and not in the "useful" parts of the skeleton, which define the primitives. We can therefore make an initial segmentation of the skeleton at these anchor points, and thereafter only process one branch at a time.

Each branch is then segmented using a robust sampling method close to RANSAC [3]. The segmentation algorithm has the following steps:

1. Select the best candidate segment $\mathcal{S}^\star$
2. Select the best candidate circular arc $\mathcal{C}^\star$
3. Extract the best of these two primitives $\mathcal{S}^\star$ and $\mathcal{C}^\star$ from the chain of points
4. If the remaining length of the chain is smaller than e, end the process, else go back to step 1

Let us now give some details for these steps.

2.3.1 Selection of the best candidate segment

We first choose two random points I_0 and J_0 from the chain to be processed (Fig. 4), and we instantiate a line model $\mathcal{S}$ passing through these points. We then enlarge the domain $[I_0, J_0]$ by "decrementing" I_0 and "incrementing" J_0 as long as these points verify $d(I_0, \mathcal{S}) \leq \varepsilon$ and $d(J_0, \mathcal{S}) \leq \varepsilon$. This gives a new segment $[I, J]$, which is the longest segment contained in the initial $[I_0, J_0]$ segment without having more outliers.

This step is repeated k times, and we keep the model $\mathcal{S}^\star$ for which the number of points contained in $[I, J]$ is maximum. The number of trials k can be estimated so that

[1] Meeting points for at least three branches of the skeleton.

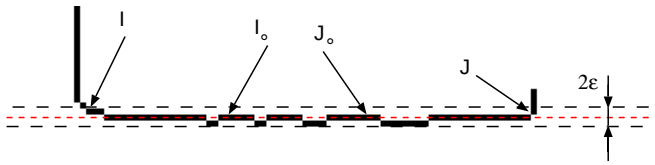

Figure 4: Segmentation through sampling.

the probability p that at least one of the couples of initial points (I_0, J_0) is good, is greater than a given value τ. A couple of initial points (I_0, J_0) is considered as good when I_0 and J_0 are simultaneously inlier points and belong to the same primitive. If L is the length (in number of points) of the chain to be segmented, e is the length (in number of points) of the smallest primitive which can be extracted from the chain, and α is the ratio total number of inlier points / L, the worst case (when all primitives are of length e), is:

$$p = 1 - \left(1 - \alpha^2 \frac{e}{L}\right)^k$$

We can therefore choose k such that $p \geq \tau$, which gives

$$k \geq \frac{\log(1-\tau)}{\log\left(1 - \alpha^2 \frac{e}{L}\right)} \tag{1}$$

2.3.2 Selection of the best candidate circular arc

The method is similar to the previous one, but we must now choose three points instead of two, and the expressions for p and k change:

$$p = 1 - \left(1 - \alpha^3 \left(\frac{e}{L}\right)^2\right)^k$$

$$k \geq \frac{\log(1-\tau)}{\log\left(1 - \alpha^3 \left(\frac{e}{L}\right)^2\right)} \tag{2}$$

2.3.3 Segmentation graph

At the end of this segmentation step, we build a topological representation of the segmentation, with a structure close to a planar map. We build a graph $\mathcal{G}$ whose nodes are the connection points of the primitives, and whose edges are the primitives themselves. Each edge contains two pointers to the nodes forming its extremities, and a pointer to the set of points associated with the primitive. Each node contains pointers to the list of primitives to which they are connected, and the label of the connected component in which it is to be found.

All this information is used in the junction reconstruction phase, as explained in the next section. A first transformation of graph $\mathcal{G}$ is performed to eliminate all chain cycles whose total length is smaller than e, as suggested in [1]. When an edge of $\mathcal{G}$ disappears, the graph is updated accordingly.

This correction may still not be sufficient to represent the drawing, as the initial topology is computed from the skeleton, which is not necessarily correct, as we have seen

in Fig. 1. Therefore, additional corrections are needed. We must also compute the actual position of each node. This is the objective of the following steps.

2.4 Instantiation of the primitives

This step aims at defining the support equations of the primitives, and their associated uncertainty domains, and from there, to compute the initial positions to give to each node. The instantiation consists in finding the support equation $F(x, y) = 0$ for each primitive. In our case, this may be line equations $F_d(x, y) = x \cos\theta - y \sin\theta + C = 0$ or circle equations $F_c(x, y) = (x - \alpha)^2 + (y - \beta)^2 - r^2 = 0$. These equations can be obtained:

- From the I_0, J_0 pair of points of the best model found during the sampling step,
- or by another robust technique, or through linear regression (e.g. algebraic least squares).

Even if it has the potential to yield a good final precision, linear regression must be handled with care: unknown shapes, such as quadrics, will usually be vectorized as chains of small segments, and linear regression may introduce large distortions in the intersections of all these segments. When a recognition step for such shapes is foreseen in a later phase, linear regression must therefore be avoided.

To each primitive p_i with its equation $F_i(x, y) = 0$, we also associate an uncertainty domain $\Delta(p_i)$, which is the space delineated by the two curves distant of ε from $F_i(x, y) = 0$, on both sides, where ε is a constant dependent on the global noise in the image—we usually set $\varepsilon \approx 1$. This domain is used to solve possible topological ambiguities which may appear during the last correction step, which we will now describe.

2.5 Dual correction

This last step consists in changing the topology and the estimated positions of the junction points. These two changes are performed simultaneously. We will describe the method by distinguishing the general case, where $N \geq 3$ primitives intersect, and the specific case of the intersection of $N = 2$ primitives.

2.5.1 General case: $N \geq 3$

We illustrate this with Fig. 5, which corresponds to $N \geq 3$ primitives. Here, the skeletonization yields two distinct junction points J_1 and J_2, which are represented in the graph by nodes e_1 et e_2. This is not correct, as the 4 primitives have been constructed to intersect at a unique point. The correction we propose relies on two remarks. First, if the position of junction points J_1 and J_2 are obtained by skeletonization with a poor precision, the *analytical* computation of their position, through the segment pairs to which they are connected, will yield two points J_1' and J_2' with a better precision. Then, if J_1' and J_2' are close enough, we can suppose that the primitives they connect do also intersect, whereas they must be considered as non intersecting if $d(J_1', J_2') > e$. The reader should note that this is only a hypothesis–we have no formal proof that this is always true, but practice has shown it is reasonable to assume so.

These two observations lead us to the following correction method. We first look for the two closest nodes e_i and e_j of $\mathcal{G}$, belonging to the same connected component, and we build the set E_0 of primitives connected to these nodes, except the primitive (e_i, e_j)[2].

[2]This can be extended without loss of generality to the case when $\mathcal{G}$ contains only one node e_i. The primitives stemming from e_i are grouped in the same way and we just write $e_j = e_i$.

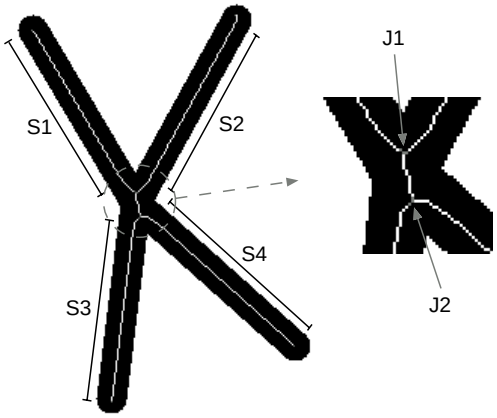

Figure 5: Intersection of $N \geq 3$ primitives.

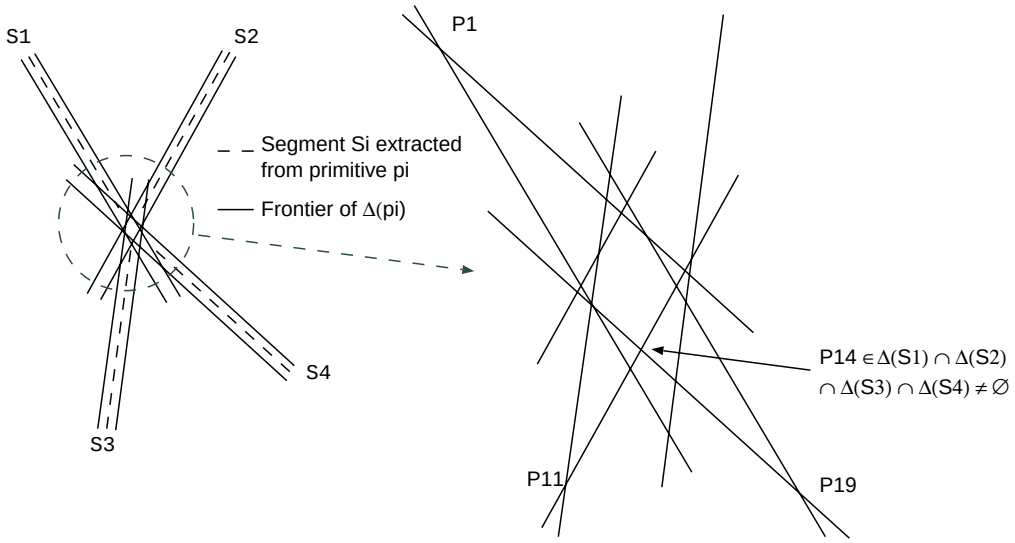

Figure 6: Intersection of uncertainty domains.

We include the condition that they must belong to the same connected component of the image, so that we avoid connecting primitives that were disconnected in the original drawing. The set E_0 simply represents a list of primitives which *may* intersect at one or more points. We must then find these points.

We therefore construct, from E_0, the set P_0 of intersection points for the borders of the uncertainty domains $\Delta(p)$ associated with each primitive p of E_0, as illustrated by Fig. 6.

Then, we check for each intersection point $X \in P_0$ that it is inside the uncertainty domain $\Delta(p)$ of each primitive $p \in E_0$. This check can be easily performed by evaluating the sign of $F_{sup}(X).F_{inf}(X)$, where F_{sup} and F_{inf} are the equations of the borders of $\Delta(p)$. When X belongs to a given domain $\Delta(p)$, we consider X as being also a point of p. Then, to each candidate point $X \in P_0$, we associate the set $\mathcal{U}$ of primitives to which p does belong, and we keep the point M which has the largest associated set $\mathcal{U}$. The fact that M exists implies that the primitives of $\mathcal{U}$ do intersect, so we can define a new set $E_1 = p \in E_O : \Delta(p) \notin \mathcal{U}$ and change the value of E_0: $E_0 = E_0 \setminus E_1$. The procedure is then repeated with E_1 and so on, until there are no more primitives to group.

At the end of this step, we have n groups of primitives $E_i, i = 0, ..., n-1$, each group

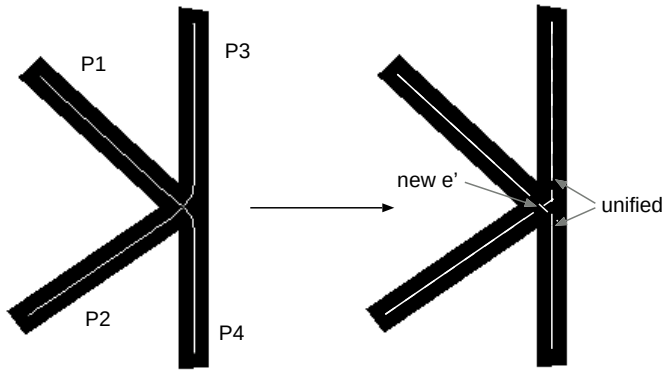

Figure 7: Isolated primitives after computing the sets E_i.

containing a variable number of intersecting primitives. We then consider separately the cases $|E_i| = 1$ and $|E_i| \geq 2$ for the following steps.

Case a: $|E_i| = 1$. This is a special case, where a single isolated primitive does not intersect with any other, despite the estimation which had been implied by the skeleton information. Although it is exceptional, this may happen, as illustrated by Fig. 7, where the primitive pairs (P_1, P_2) and (P_3, P_4) cannot intersect *by construction*. As the pair (P_3, P_4) has no intersection either, these two primitives are isolated. From a topological point of view, these isolated primitives are left unchanged. This means that their common node (e_i or e_j in the present case) is kept as it was. In the example given by Fig. 7, the two extremities of P_3 and P_4 are merged and represent the same node. The resulting junction is formed by $n = 2$ primitives, which constitutes a particular case described in § 2.5.2.

Case b: $|E_i| \geq 2$. This is the general case, where the intersection of the uncertainty domains is non empty, so that there is a junction point M inside this intersection. We then propose to compute the position of M through the weighted orthogonal least squares method. For each set E_i of primitives $p_1, ...,$ we compute the point M which minimizes the sum

$$S = \sum_{j=1}^{|E_i|} \omega_j d^2(M, p_j)$$

where $d(M, p_j)$ is the distance from M to primitive p_j, and where each ω_j is set to be the length—in number of pixels—of primitive p_j. This weighting gives a better confidence to the final estimation, as the least squares error induced by a displacement of M, even a small one, is proportional to the length of the reference primitive.

This minimization problem is linear as long as the p_j are lines; thus, its solution can be computed analytically. The linearity is lost as soon as there is a circular arc, but the solution remains analytical as long as E_i contains only two primitives. With three or more primitives, of which one at least is a circular arc, we must use a numerical method, as there is no general analytical solution anymore. We do not yet take into account the case where $|E_i| \geq 3$, as a numerical method may end up finding a local minimum, and there are probably many such minima inside the intersection domain. Also, we have until now never met this case in the architectural drawings we have processed, as circular arcs usually appear in the thin-line layer, where they are associated with simple segments to represent doors, windows, etc. But in a more general framework, we must probably take this problem into account...

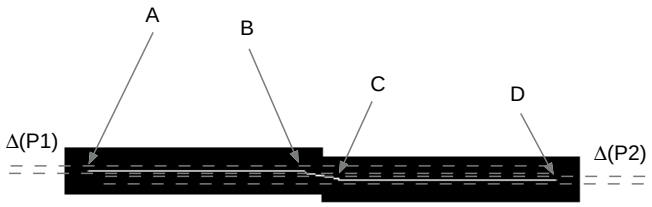

Figure 8: Primitives without apparent intersection.

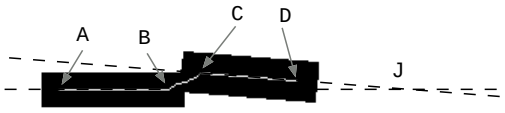

Figure 9: Combined effects of shift and small angle.

Once the intersection point M has been determined, the node to which it corresponds is given M as its new geometrical position. The supports of the primitives stemming from this node are then re-instantiated using the following rules:

1. the position of the other extremity remains unchanged,
2. the supports of the segments are re-instantiated by the line passing through the two extremities,
3. the supports of the circular arcs are re-instantiated by the circle passing through the two extremities, with the same radius, and with the center located in the same half-plane as in the previous model[3].

2.5.2 Special case $N = 2$

This case may generate difficult ambiguities, if the supports of the primitives are parallel, depending on the kind of primitives. We process separately three types of possible junctions: segment–segment, segment–arc, arc–arc.

Case a: segment–segment. The difficult case appears when the two supports are strictly parallel, while being topologically connected. This is illustrated by Fig. 8, where the borders of $\Delta(P_1)$ and $\Delta(P_2)$ have no point in common, whereas the primitives they define are topologically connected (the BC chain is removed as its length is smaller than the line thickness). The problem is to find a way to vectorize this part of the graphics.

There may also be a numerical problem, when the value of the angle formed by two segments is sufficiently close from π to put the intersection points of the borders outside of the acceptable zone (Fig. 9).

We propose the following solution to this problem: using arbitrary-precision computation, we determine if $[AB]$ and $[CD]$ define a computable intersection point J. If this is the case, the validity of J is given by the sign of $(\vec{AB}.\vec{BJ}).(\vec{DC}.\vec{CJ})$, which must be equal or greater than zero. If J is not computable, we disconnect the segments, and leave them as they are.

[3] We are aware that this way or re-instantiating circular arcs supports is somewhat arbitrary. In fact, we assume that the displacement induced by the new M remains small. It would certainly be more general and robust to determine the position of M by trying to minimize the error of a model with N branches stemming from M.

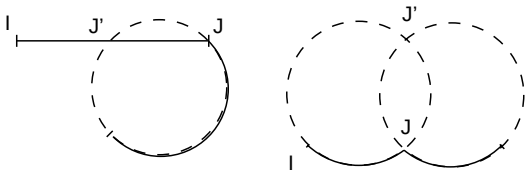

Figure 10: False junction points.

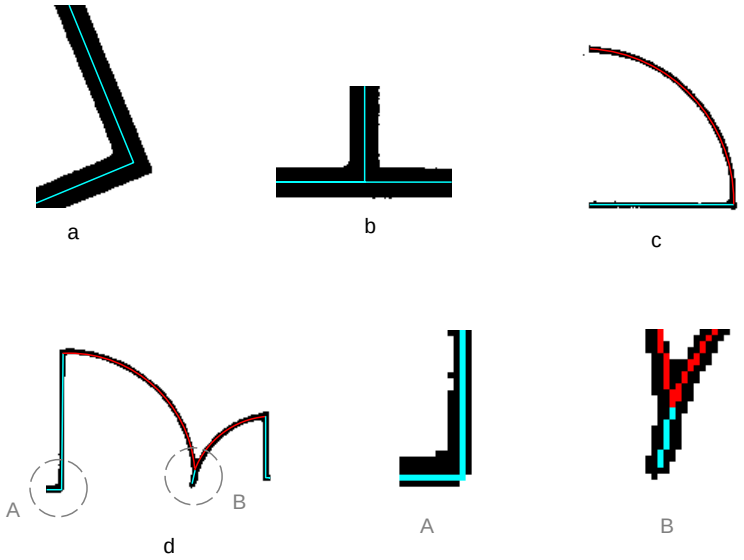

Figure 11: Corrections made in the most frequent cases.

Case b: segment–arc and arc–arc. There are no special robustness problems with these two cases, except the fact that the intersection of the primitive supports may create a false junction point J', in addition to J, as illustrated by Fig. 10. A simple comparison of the lengths of the arcs IJ and IJ' is actually sufficient to eliminate this false point.

2.6 Connecting the e'_u nodes

This is probably the most critical part of the process, as we only have two pieces of information on the nodes e'_u which have been created. First, we know that they are located in the intersection zone of a set $\mathcal{E}$ of primitives which are topologically connected, according to the skeleton. Secondly, we know that they correspond to intersections of subsets of $\mathcal{E}$, according to the uncertainty computation. We have no other *a priori* information which may help us in deciding whether these points should be connected or not.

As a first try, we propose to connect these points pairwise, starting with the closest pair, and ending with the pair of points whose distance is the largest.

3 Results

3.1 Usual cases (real data)

In Fig. 11, we report typical results obtained on the most frequent cases. The method easily corrects the usual distortions on L and T junctions. The extraction of the arc and the computation of its intersection with the segment is also correct in case c). Case d),

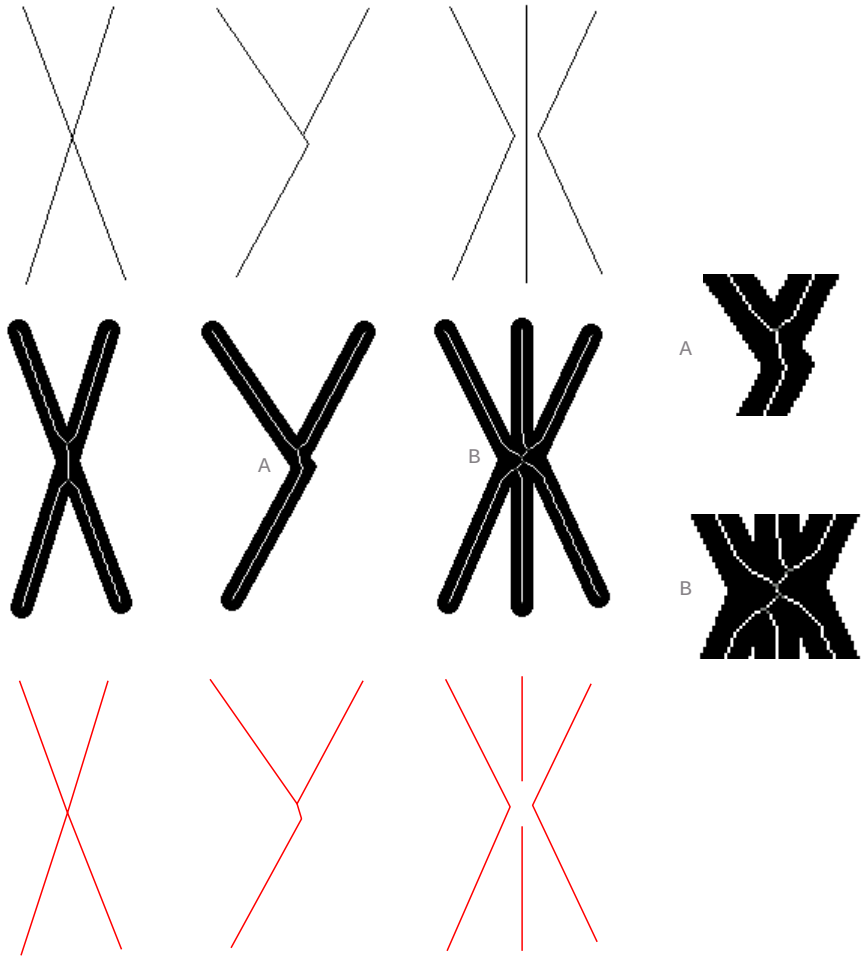

Figure 12: Corrections made in more difficult cases.

on the contrary, is more difficult. First, the result of vectorization depends on the value given to constant ε during processing. With $\varepsilon = 1.2$, detail A shows that 2 segments are extracted. When ε comes closer to 1, a circular arc is inserted between the two segments; actually, the binary image may be interpreted in both ways. Secondly, there is still a problem at the intersection between the two arcs. Here, the left arc is intercepted by the right arc, and is thus truncated. The skeleton contains a local anchor point connecting the two chains which correspond to the arcs, and a third chain which prolongates the left arc. The latter chain is interpreted as being a segment, and cannot be removed, as its length is larger than the line thickness. This defect is not corrected by our method.

3.2 Difficult cases (synthetic data)

Fig. 12 shows other results, obtained on synthetic data. The first group shows the initial construction of junctions. The second group shows the lines thickened to a width of 15 pixels, and skeletonized. The last group shows the extracted vectors. The X junction is processed in two steps. First, the upper and lower pairs are grouped into two junction points J_1 and J_2, as suggested by the skeleton. It is only when step 5 is iterated that J_1 and J_2 are merged.

The Y junction is processed differently, and has a defect. As for the X junction, the

upper segments are grouped to yield an intersection point J. But the lower branch remains disconnected. Initially, the chain connecting its upper extremity J' to the anchor point is removed (length < 15), so there is no valid intersection with one or the other of the two upper branches. When topology is checked in the last step of the algorithm, J and J' are linked, but the final result is not exactly the expected one. This emphasizes a default of the method: if the two left segments had been grouped first, the final resulting topology would have been different.

There is a similar problem with the last case: the left and right pairs of branches are correctly grouped, but the intermediate segments are not; their lengths are shortened by the cutting points computed during segmentation.

4 Conclusion and perspectives

The method we have presented yields good results on "classical" drawings, and has the advantage to be simple and easily extendible. As an example, it is possible to introduce more complicated curves, such as conics and splines, typically found in CAD drawings. However, the method still suffers from some drawbacks, and we think that it is necessary to make several improvements in future work.

First of all, it is obvious that the random sampling method used to segment the skeleton is not time efficient at all. From equations 1 and 2, it is easy to see that the number of trials to be performed reaches huge values when the value of e decreases. This problem actually happens because the method always assumes that the worst case occurs. As noted in [7], an adaptive version of the random method may be considered. Such a method should be able to reduce the number of trials, as well as to focus on a promising region by taking into account the results computed at each step.

The method is also not able to extract interrupted patterns. This means, for example, that the vectorization of two primitives, one intercepting the other, will always consist of at best four primitives, not two. Actually, this problem requires interpretation in many cases: are we talking about two segments locally forming an isolated X junction, for example, or about a very long line intercepting a small arc that is a part of a door? But both situations actually weaken the precision of the estimated position of the junction point–if, of course, the existence of this point makes sense.

A still more complicated problem is probably the way we connect the computed junction points e'_u in the last stage of the method. In fact, there are many possible connections of those points, but the rules used to connect them should certainly not come exclusively from the vectorization.

References

[1] 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.

[2] Ph. Dosch, G. Masini, and K. Tombre. Improving Arc Detection in Graphics Recognition. In *Proceedings of 15th International Conference on Pattern Recognition, Barcelona (Spain)*, volume 2, pages 243–246, September 2000.

[3] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus : A Paradigm Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

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

[5] 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 14th International Conference on Pattern Recognition, Brisbane, Australia*, pages 406–408, August 1998.

[6] K. Tombre, Ch. Ah-Soon, Ph. Dosch, G. Masini, and S. Tabbone. Stable and Robust Vectorization: How to Make the Right Choices. In A.K. Chhabra and D. Dori, editors, *Graphics Recognition – Recent Advances*, volume 1941 of *Lecture Notes in Computer Science*, pages 3–18. Springer Verlag, 2000.

[7] Aimo Törn and Antanas Žilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1989.

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