A Survey of Non-thinning Based Vectorization Methods

Liu Wenyin Dov Dori

Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology, Haifa 32000, Israel
{liuwy; dori}@ie.technion.ac.il

We survey the methods developed up to date for crude vectorization of document images. We classify them into six categories: thinning based, Hough Transform based, contour-based, run-graph based, mesh-pattern based, and sparse pixel based. The crude vectorization is a relatively mature subject in the Document Analysis and Recognition field, though there are rooms to improve. The purpose of the survey is to provide researchers with a comprehensive overview of this technique for them to choose a suitable method when developing their vectorization algorithms and systems.
Keywords: Vectorization, Document Analysis and Recognition, Polygonalization

1 Introduction

Vectorization, also known as raster to vector conversion, is a process that finds the vector lines from the raster images. It is widely used in the document analysis and recognition (DAR) field as a preprocessing of high-level object recognition, such as optical character recognition (OCR) and graphics recognition. Basic vectorization concerns grouping the pixels in the raster image into raw wires that described by several attributes, such as characteristic points and line width. Advanced vectorization includes line fitting and extending, which yields fine wires. We refer to crude vectorization as the basic vectorization process that takes raster (binary) image as input and yields coarse wire fragments, which may be bars (non-zero width straight line segments) and polylines (chains of bars linked end to end).

Many crude vectorization methods have been developed and implemented since the image processing techniques were introduced more than thirty years ago. These methods are roughly divided into six classes: Hough Transform (HT) based methods, thinning based methods, contour based methods, run-graph based methods, mesh pattern based methods, and sparse pixel based methods. However, except for the HT based methods, a typical vectorization process includes (1) medial axis points sampling, or medial axis representation acquisition, which is the kernel processing for information reduction so that only the important points that represent the medial axis are determined; (2) line tracking, which follows (tracks) the medial axis points found in the first stage into chains of points; (3) line segment approximation or polygonalization, which removes non-critical points from the point chains found in the second stage and uses bars and polylines to represent the remaining critical points. The main difference among the above mentioned classes of vectorization methods lies in the first two subprocesses and there are several polygonalization algorithms that can be employed in the third subprocess.

Thinning based methods are so conventional that most of the earlier vectorization systems, e.g., Kasturi et al. (1990), took advantages of them. Generally, the objective of thinning is to reduce the data volume such that only the image's

topological shape, which is size and orientation invariant. The result is usually for further processing. Most thinning algorithms are capable of maintaining connectivity. However, the major disadvantages are high time complexities (e.g., cubic for iterative thinning), loss of shape information (e.g., line width), distortions at junctions (e.g., at "T" and "X" intersections), and false and spurious branches (e.g., at line ends and corners). Although they may be used in vectorization of line drawings, their common application is OCR, in which the image size is usually small and the line width is not critical. Comprehensive surveys of thinning algorithms are given by Tamura (1978), Smith (1987), and Lam et al. (1992). Performance evaluation of thinning algorithms have been done by Lee et al. (1991), Lam and Suen (1993), Jaisimha et al. (1993), etc.

In this paper, we survey the non-thinning methods and provide researchers with a comprehensive overview of this technique for them to choose a suitable one when developing their vectorization algorithms/systems.

2 Hough Transform Based Methods

Dori (1997) discusses the application of Hough (1962) Transform (HT) in the vectorization of straight line images by transforming spatially extended patterns in binary image data into spatially compact features in a parameter space. This means that a difficult global detection problem in the image space is transformed into a more easily solved local peak detection problem in the parameter space. One way the HT can be used to detect straight lines is to parameterize it according to its slope and intercept. Straight lines are defined in Equation (1).

$$y = mx + c \tag{1}$$

Thus, every line in the (x,y) plane corresponds to a point in the (m,c) plane. Every point on the (x,y) plane can have an infinite number of possible lines that pass through it. The gradients and intercepts of these lines form on the (m,c) plane a line described by Equation (2).

$$c = -xm + y \tag{2}$$

The (m,c) plane is divided into rectangular "bins" which accumulate for each black pixel in the (x,y) plane; all the pixels lying along the line in Equation (2). When the line of Eq (2) is drawn for each black pixel, the cells through which it passes are incremented.

After accounting for all the pixels in the image space, lines are detected as peaks in the transform space. Taking into account noise, each peak that is greater than a predefined threshold is used to form a line defined in Eq (1). In practice, this assumed line may be a combination of several collinear line segments (bars). Hence, the pixels on the original image along the assumed line is followed so that the end points of these segments are found. The line width is also determined during the line tracking by examining the width at each pixel.

For straight line detection, the HT visits each pixel of the image once. Therefore, its time complexity is linear to the total pixel number, which is the product of the image width and height. Since each side of the image is linear to the image resolution, we use the image resolution as the unit in analyzing the time complexity

of vectorization algorithms. Hence, the time complexity of the HT based vectorization method is quadric to the image resolution.

Since peaks are expected to be formed in the (m,c) plane for points whose m and c belong to broken or noisy lines in the original image, HT can be used to detect lines in noisy images. However, since the gradients and intercepts are sampled sparsely, they may not be so precise as the original lines. Hence, the quality of detected lines is far less precise for slanted lines. This can be seen from Figure 6(b), which is produced by an implementation of the HT based vectorization method by Dori (1997). Moreover, the HT based methods can yield bars only and cannot generate polylines.

3 Contour Based Methods

Aiming at lowering down the computational burden of thinning, another group of vectorization algorithms try to reduce the data volume before sampling the medial axis points. The main idea is finding the edges (contour) of the line object (shape) first and calculating then the middle points of the pair of points on two opposite parallel edges. This group of vectorization algorithms sample and track (follow) the medial axis points simultaneously. This is different from thinning based algorithms, which do line tracking after all medial axis (skeleton) points are sampled. The main computational operation in these methods is edge detection and polygonalization. The time complexity of edge detection is linear to the pixel volume, i.e., quadric to the image resolution. Polygonalization is only linear to the pixels on the contour, as discussed in Section 7. Hence, this group of vectorization algorithms have quadric complexity and are much faster than thinning based algorithms. Moreover, the line width is also much easier to obtain, which is very important for higher level drawing interpretation.

Assuming the edges are found and polygonalized, the medial axis points of two approximately parallel edges are defined by Jimenez and Navalon (1982) to be the midpoints of the perpendiculars projected from one side to the other. Starting with a "well behaved" edge (which is a vector), a linear search finds the nearest edge on the other side of the object and several perpendiculars are projected from one to the other. Subsequent edges are easier to find simply by following the edge until a junction point is found.

The problem with all algorithms of this type is how to deal with junctions. There are two main problems associated with junctions. The first one is the merging junction formed by intersection at a small angle, which is most likely be missed during tracking. The second case is the cross intersection, where how to join up the lines are problematic. It is important that the algorithm be robust and able to deal with all images without misjudging a junction and producing an incorrect skeleton. Hence, it is hard to use in curve image vectorization.

4 Run Graph Based Methods

Extending the ideas of Di Zenzo and Morelli (1989) and Boatto et al. (1992), Monagan and Roosli (1993) define more formally the run graph as a semi-vector

representation of a raster image before line detection and other segmentation. It is sufficient for structural representation of the line-like images and efficient for line extraction. It is information preservative and easier to operate.

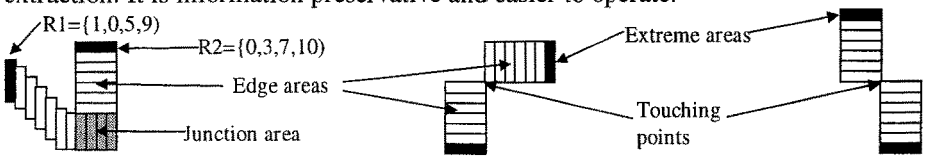

Figure 1. Illustrations of run graph representation of raster image.

As shown in Figure 1, several cases of run graph representation for raster images are illustrated. A run has a direction, which can be either horizontal or vertical. It is a maximal sequence of black pixels in its direction. Hence, a *run* can be defined using the following quadruplet (quaternion)

$R = \{d, c_{d'}, b_d, e_d\}, b_d \leq e_d$

where d is referred to as the *direction* of the run, which can be either 0 for horizontal or 1 for vertical, $c_{d'}$ is referred to as the *orthogonal coordinate*, which is the coordinate of the run in the direction orthogonal to the run direction, that is, $c_{d'}$ is the row number if d is horizontal, or the column number if d is vertical, b_d is referred to as the *begin (directional) coordinate*, which is the coordinate of the first pixel of the run in the run direction, and e_d is referred to as the *end (directional) coordinate*, which is the coordinate of the last pixel of the run in the run direction. A vertical run R1 and a horizontal run R2 are illustrated in Figure 1. A run can also be expressed by two endpoints, in which case, the direction is inferred from the two points' coordinates. If their x coordinates are equal, the direction is vertical, if their y coordinates are equal, the direction is horizontal. However, the above expression is more efficient to operate in the line extraction procedure.

More definitions is required in order to give the formal definition of run graph. A *subrun* is a part of a run. Runs A and B are *adjacent* if they have the same direction, the difference of their orthogonal coordinates is 1, and the difference of their maximal begin coordinates and their minimal end coordinates is less than or equal to 1. If A and B are adjacent and A's orthogonal coordinate is smaller than that of B, A is called *predecessor* of B, and B is called *successor* of A. A run is *regular* is it has only one predecessor and only one successor, otherwise, it is *singular*. Two runs are *conjugate* if they are orthogonal and overlap one and only one pixel, i.e., they cross. A vertical run is a *short* run if it is regular and not longer than all its conjugates. A horizontal run is a *short* run if it is regular and shorter than all its conjugates.

An *edge area* consists of a maximal sequence of adjacent short runs in the same direction. An *extreme area* consists of only one run that has no adjacent run on one side. A *junction area* consists of a maximal sequence of adjacent vertical runs/subruns of pixels belonging neither to vertical nor to horizontal short runs. Monagan and Roosli (1993) introduce the concept of *touching point*, which does not actually cover any pixel point in the image. A *touching point* is formed between two adjacent runs if they do not overlap in their direction, or between two orthogonal runs if they do not overlap but touch each other end to end, as shown in Figure 1. However, this definition is contradictory to the definition of edge area, since two

touching and adjacent runs are parts of an edge area. Removing this definition, a run graph RG can be formally defined as follows.

$RG = <V, E>$,

where V is a set of nodes (vertices), which are either junction areas or extreme areas, and E is a set of edges, which are edge areas that linking between nodes.

According to Boatto et al. (1992), the procedure of constructing a run graph of an image is as follows. The first step is to build both horizontal and vertical simple run graphs, which consists of only horizontal runs and vertical runs, receptively. Second, edges are built as sets of adjacent regular short runs. The remaining pieces of the image, encoded as lists of vertical runs and subruns, are the node areas.

The line extraction procedure takes as input such a run graph. The shape of each node is then refined by a heuristic procedure (runs splitting) that attempts to minimize the area of the node and to maximize the lengths of the connected edges. The midpoints of the short runs in the edge areas are taken as the skeleton points, which further undergo a polygonalization procedure to produce final polylines (or bars). As can be seen, the line extraction is efficient due to working on the semi-vector run graph representation.

However, the preprocessing, the run graph construction also takes time, which visits each black pixel on the image at least once. This means, the complexity of the run graph based vectorization is quadric to the image resolution. Disadvantages of the method also include inaccurate intersection points due to coarse locations of junction areas, false (undesirable) junction areas yielded when the run direction changes, or at noise points on uneven edges. So it is not appropriate for curve (arc) representation, i.e., vectorization of arc images.

5 Mesh Pattern Based Methods

Mesh patterns are first introduced by Lin et al. (1985) to detect characteristic patterns in diagrams, e.g., logic connection diagrams. A connection diagram understanding system is then developed based on this method. The basic idea is to divide the entire image using certain meshes and the characteristic patterns are detected by only checking the distribution of the black pixels on the border of each unit mesh. A control map is then made for the image using these patterns. Finally, the extraction of long straight line segments are performed by analyzing the control map. Figure 2 shows the principle of the mesh pattern based line detection method. In Figure 2(a), the image is divided into square meshes, which are defined by an equal proper mesh size, n. Each unit mesh is analyzed according to the pixels on the one pixel wide border only. It is then labeled according to its characteristic pattern identified to a known one in the pattern database. The image is then represented by a control map, in which each unit mesh in the original image is replaced with its characteristic pattern label. In Figure 2(b), the central part of the image in Figure 2(a) is represented by a control map consisting of two meshes, whose labels are "K" and "T", respectively. The lines are recovered and tracked from mesh to mesh in the control map by analyzing the characteristic patterns of the meshes, as shown in Figure 2(c).

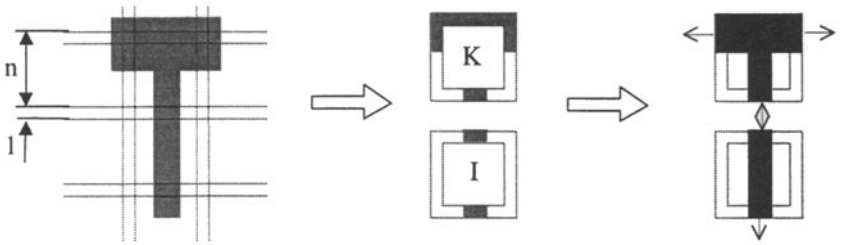

Figure 2. Illustration of mesh pattern based line extraction method. (a) image and meshes. (b) mesh pattern labels and the control map of the two central meshes of the image. (c) Lines extracted by analyzing the control map.

In their characteristic pattern database, Lin et al. (1985) have defined 51 known characteristic patterns and give each of them an identification label. The other unknown complicated patterns are labeled with question marks (?). Such areas are further processed by a more complex detailed processing procedure during the control map analysis. This procedure scans every pixel in these areas and every black pixel is labeled as being a line pixel or a feature point accordingly.

Vaxivere and Tombre (1992, 1995) extend the mesh pattern based method for mechanical engineering drawings analysis and recognition. They use dynamic meshes. Complicated (question labeled) meshes are further split into several smaller but known mesh patterns, whose shape may not be squares. Their vectorization result is a data structure describing the image as segments of different types (e.g., thick line, thin line, and contour of black blob) and junctions between these segments.

The mesh pattern based line detection methods are fast due to sparse pixel access. Since only the pixels on the mesh border are visited, its time complexity is linear to one side of the image, i.e., linear to the image resolution. The access ratio, which is the ratio of the number of accessed pixels to the total number pixels in the image, is about $2/n$, if all meshes are labeled with known labels. However, if no extra memory is used for borders of neighbor meshes, each accessed pixel may be accessed twice since it is on the border of two neighbor meshes. Anyway, as claimed by Lin et al. (1985), the access ratio, is about 2/15 to 3/15, when n is 16 pixels. The access time increases as the number of question labeled meshes increases. Lin et al. (1985) also claimed that an optimum mesh size is just a little larger than the maximum line width in the image, in which case, the processing time is about 55% of the processing time of the complete application of the detailed processing procedure to the entire image.

However, the mesh size is hard to control. Big mesh size introduces more question labeled meshes, which requires much more processing time. Small mesh size increases the access ratio and may also make the line extraction difficult. Moreover, it is not suitable for detection of more complex line patterns, e.g., arcs and discontinuous lines (e.g., dashed and dash-dotted lines). As claimed by Lin et al. (1985), the proper mesh size, n, should be larger than the maximum width of the line segments but smaller than the minimum interspace between two line segments on the image. In this case, in the areas where only long straight line segments exist, each side of a unit mesh border intersects mostly one line segment of the image.

Moreover, n should be smaller than the smallest (shortest) line segment of the image, so that the background area judgment can be simply performed on the control map by detecting the characteristic patterns labeled with blank labels. This excludes the dot segments, whose lengths can be as small as their widths, or even shorter). These dot segments may be missed during line tracking. It may obtain good performance to apply the method to sparse, straight, and long lines.

The nature of investigating only the borders of meshes may link lines with gaps that are shorter than the mesh size. This may be a big advantage in some cases but a big disadvantage in others, since its better to fill the gaps of a broken line due to noise but gaps of dashed lines should be left unattended.

6 Sparse Pixel Based Methods

Orthogonal Zig-Zag (OZZ) is a novel vectorization algorithm developed by Dori et al. (Chai and Dori 1992, Dori et al. 1993). Like the mesh sampling (Lin et al. 1985), OZZ samples the image sparsely. The idea of OZZ is as follows. A horizontal scan line, which moves down 10 pixels each time, goes from left to right of the image. When it encounters a black pixel, it enters in the area and an OZZ procedure begins. It goes ahead until a white pixel is encountered (i.e., the light hits the edge of the area) or the traveling length within the black area (which is referred to as a run) exceeds a predefined threshold. If it hits the edge of the area, the midpoint of the run is recorded, it turns orthogonally within the black area, and the OZZ procedure is continued, as shown in the slant case of Figure 3. If the run exceeds a predefined threshold, e.g., 30 pixels (Chai and Dori 1992), it stops, two new lights are emitted from the stop orthogonally, one to left and the other to right. When they hit the edges of the black area, a run is formed by joining the two lights' traces together. The midpoint of the joint run is recorded and a new light that is orthogonal to the joint run is emitted from the midpoint. The OZZ procedure is continued from this newly emitted light, as shown in the horizontal case of Figure 3. After the OZZ procedure hits the end of the area, another OZZ procedure is performed by emitting a light orthogonal to the scan line from the point where the scan line hits the area outside for the first time, if a slant area is visited, as shown in the slant case of Figure 3. These midpoints are followed during the light tracking and used as the medial axis points of the bar image. They are further processed by the polygonalization, so that only bars are yielded by breaking the point chain into bars where necessary.

After the horizontal pass is finished, a vertical pass is performed similarly to the horizontal pass. In the vertical pass, the vertical scan line goes from top down every 10 pixels. When the scan line hits the edge of a black area, the OZZ procedure begins. After the vertical pass is over, the combination of bars found in the two passes are performed, so that overlapped bars are omitted.

OZZ is time-efficient due to the sparse sampling of the image. As shown in Figure 3, the number of pixels visited by OZZ is linear to sum of the image width and height. Hence, it is linear to the image resolution. However, it is also noise-sensitive. Moreover, it is designed to yield bars only. Hence, curve images are vectorized to a set of bars overlapping each other at their ends. This is shown in Figure 6(c). Therefore, it does not perform well in vectorizing arc images.

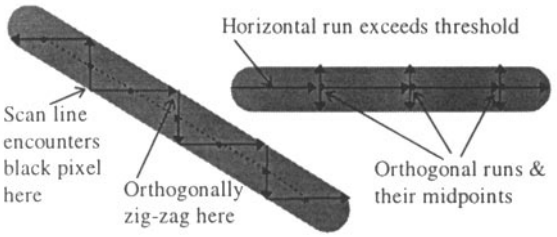

Figure 3. The principle of Orthogonal Zig-Zag (OZZ) vectorization algorithm.

Inspired by the OZZ idea, Liu and Dori (1996) developed the sparse pixel vectorization (SPV) algorithm. SPV improves the OZZ method in the following three aspects. (1) The general tracking procedure starts from a reliable starting medial axis point found by a special procedure for each black area. (2) A general tracking procedure is used to handle all three cases of OZZ, i.e., horizontal, vertical, and slant. Therefore, only one pass of the scanning is needed and combination of the two passes is avoided. It is faster than OZZ. (3) Junction recovery procedure is introduced wherever a junction is encountered during line tracking.

To start the SPV algorithm, a reliable starting medial axis point is found first, e.g., P_3 in Figure 4, which is approximately middle points of both the horizontal and vertical runs passing it. The lengths of both the horizontal and vertical runs are known at the first medial axis point P_3. If the horizontal run is longer than the vertical run, the bar's inclination is more horizontal than it is vertical (i.e., its slant is less than 45°), in which case the length direction is set as horizontal, otherwise the length direction is defined to be vertical. The width direction is defined to be orthogonal to the length direction. If the length direction is horizontal, the tracking is done first to the right, then to the left, and if the length direction is vertical, the tracking is done first downwards, then upwards.

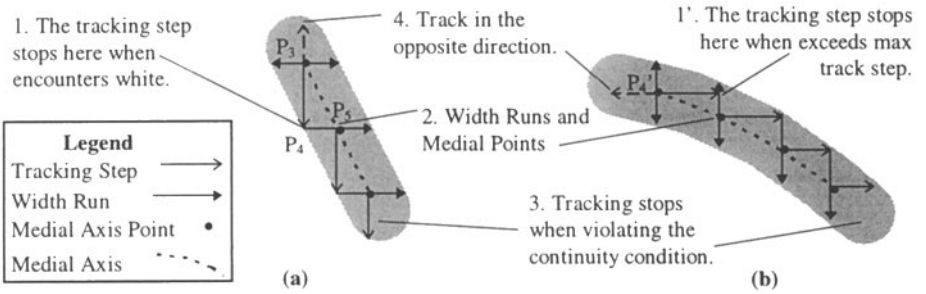

Figure 4. Illustrations of the general procedure of Sparse Pixel Tracking. (a) A vertical tracking case. (b) A horizontal tracking case.

As Figure 4(a) shows, going from the first medial axis point P_3 in the tracking direction, which is vertical, we begin the main Sparse Pixel Tracking procedure for this area. To start the tracking cycle, we take a tracking step from the last medial axis point and reach point P_4. From P_4 we make two opposite directed width runs, from which we get the (undirected) width run and its middle run point P_5 that serves as the new medial axis point. This is the end of tracking this cycle. We repeat these tracking cycles while recording and monitoring the medial axis points and the width run lengths, as long as all of the following four continuation conditions are satisfied.

(1) The difference among the width runs is below some threshold.
(2) The medial axis point is not in an area occupied by another detected vector.
(3) The pixel visiting direction is the same as that of the previous tracking cycle.
(4) The length of the tracking step is greater than zero.

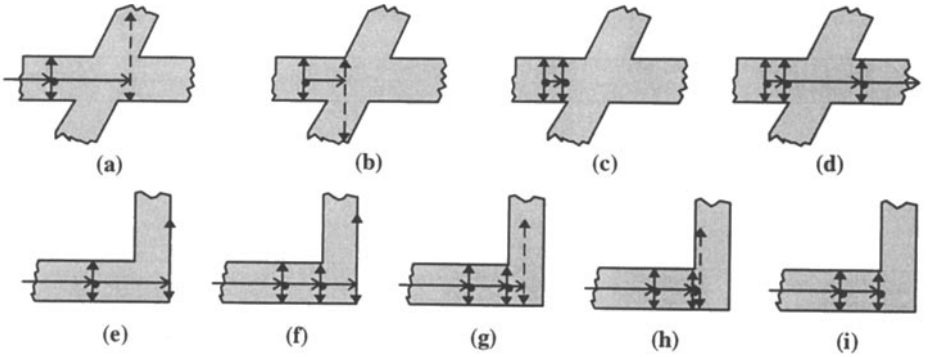

Figure 5. Demonstration of the Junction Recovery Procedure. (a)-(d) Cross case. (e)-(i) Corner case.

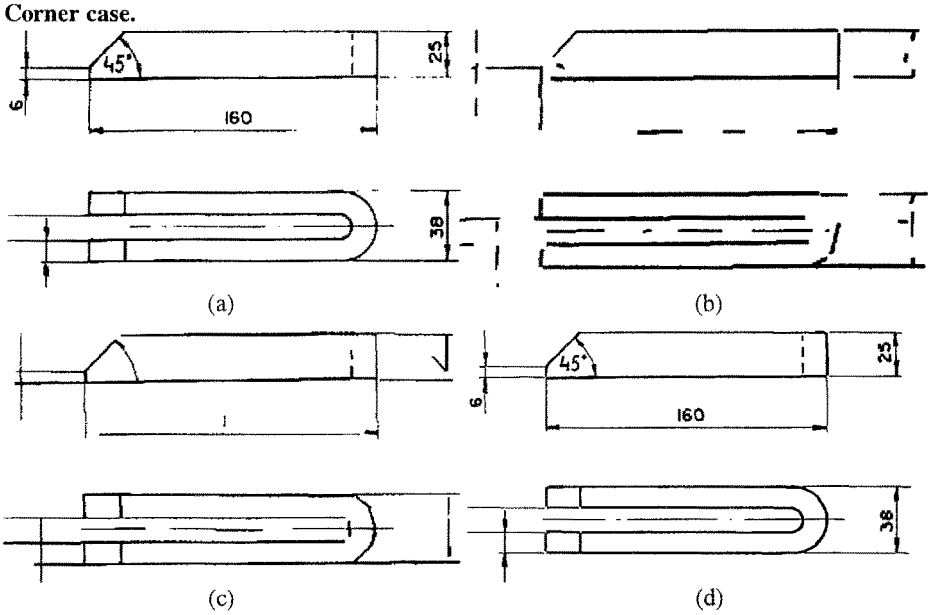

Figure 6. Comparison of HT, OZZ, and SPV results. (a) Original image. (b) HT based vectorization. (c) OZZ vectorization. (d) SPV.

When one or more of the first three continuation conditions is violated, the Sparse Pixel Tracking procedure pauses and a Junction Recovery Process starts using the Junction Recovery Procedure, as exemplified by a cross and a corner in Figure 5. Junction recovery is an iterative procedure consisting of three steps: (1) retreating to the last medial axis point; (2) adjusting the tracking step length by halfsizing the length of the current tracking step; and (3) testing the conditions at the new position. If the test fails, i.e., the new medial axis point also violates one of the above conditions, a new iteration is reapplied with the tracking step halfsized again. The iterations halt when the tracking step becomes zero, as in the fourth condition. If at

the new medial axis point all the conditions are met again, then we continue the general tracking procedure, using the normal tracking step from the new medial axis point. By so doing, the procedure may overcome some uneven area, where the width runs are significantly different, as demonstrated in Figure 5(a-d). If the tracking step length becomes zero, the tracking stops at the last medial axis point, as in the case of a corner, shown in Figure 5(e-i).

Figure 6(d) has shown the SPV improvement to the OZZ algorithm and the HT base method. SPV has been successfully used for line detection by Yoo et al. (1998) and evaluated by Liu and Dori (1997) as time efficient and shape preservative.

7 Polygonalization and Line Width Determination

The result of the line tracking procedure is a chain of points (or polyline) that are on the approximate medial axis of the black area. Although it can also be regarded as a vector representation of this black area, some intermediate points are redundant because they are (approximately) on the straight line segments formed by their neighbors. These points should be removed from the chain list in order to give the most concise vector representation of this black area, that is, to use the polyline with the fewest edges (or vertices) to approximate the original one while maintaining the original shape well. This is done by a procedure called polygonal approximation. Most vectorization methods apply the polygonalization procedure to the tracked coarse polyline. To preserve the original shape well, a value denoted by ε is predefined as a parameter to decide which points should be removed and therefore to constraint the precision of the resulting polyline. The smaller the value, the more vertices but the more precise the edges. Usually ε is set to be about 1 pixel to keep the original shape to the most extent in the coarse vectorization. But in the situation of polygonal approximation to engineering drawing lines, ε can be set up to half of the line width.

It is usually done by finding as few as possible critical points from the point chain, such that the distances of the non-critical points between every two nearest critical points to the line joined by these two critical points are less than ε. So the criticality of a point is determined by its distance to the line formed from its two neighbor critical points on both sides. But critical points can not be available in advance. The determination of critical points is the main procedure of polygonalization.

Various polygonalization algorithms are also developed, which can be classified into two groups. The first class of methods, e.g., those of Montanari (1970) and Hung and Kasvand (1983), use a global way to examine the criticality of points. The second class of methods, e.g., those of Sklansky and Gonzalez (1980) and Jimenez and Navalon (1982), use local optimization, that is, the polygonalization is done by going ahead from a new critical point (one endpoint of the chain is a critical point for initialization) each time and finding as many as possible intermediate non-critical points before the next critical point is found. Among the above four polygonalization algorithms, the one of Sklansky and Gonzalez (1980) is suitable for all applications. Although it cannot guarantee the minimum number of critical

points due to one-pass local optimization, it is very time efficient and the criticality of the remaining points is guaranteed. Moreover, the algorithm is theoretically sound very much and quite simple to implement. In general, the time complexity of this method is linear to the number of the original vertices. This is confirmed by experiments of Dunham (1986), who also shows that the number of critical points left are also minimum among the nine algorithms tested.

After the fine polyline is determined, the line width can be calculated from the average of the width at each intermediate points of the polyline. The procedure can be applied to all groups of vectorization methods. If the width is not recorded, the original image should be referred so that the width at these points can be obtained.

8 Summary

Comparing the vectorization methods discussed in this paper, we list some characteristics of them in Table 1. Good methods should preserve the shape information, which includes line width, line geometry, and intersection junction, as much as possible so that the post processing may have a good basis to start. Moreover, it should be fast enough for practical use in real life systems.

Table 1. Characteristic comparison of the vectorization methods.

Method	Subprocess sequence*	Com-plexity	Geometry quality	Width remain	Junction recovery	Image constraints	Applications examples
HT	HT, t	quadric	Poor	No	Yes	sparse, straight	Dori [5]
Iterative thinning	s, t	cubic	High	No	No	clean, thin	Kasturi et al. [11]
Contour	edge detection, t	quadric	Poor	Yes	No	straight	[10]
Run-graph	run-graph construction, t	quadric	Poor	Yes	No	straight	Boatto et al. [1]
Mesh	s while t	linear	Poor	Yes	Yes	sparse, long	[15][24][25]
OZZ	s while t	linear	Poor	Yes	Yes	straight	Dori et al. [2][4]
SPV	s while t	linear	Good	Yes	Yes		[17][26]

*s means medial axis points sampling and t means line tracking.

Reference

[1] Boatto L et al. (1992) An Interpretation System for Land Register Maps. *IEEE Computer* 25(7):25-32

[2] Chai I, Dori D (1992) Orthogonal Zig-Zag: An Efficient Method for Extracting Lines from Engineering Drawings. In: *Visual Form*, eds. Arcelli C, Cordella LP, Sanniti di Baja G, Plenum Press, New York London, pp 127-136

[3] Di Zenzo S and Morelli A (1989) A useful image representation. In: *Proc of 5th Int. Conf. on Image Analysis and Processing*, Singapore, pp 170-178.

[4] Dori D, Liang Y, Dowell J, I. Chai (1993) Spare Pixel Recognition of Primitives in Engineering Drawings. *Machine Vision and Applications* 6:79-82

[5] Dori D (1997) Orthogonal Zig-Zag: an Algorithm for Vectorizing Engineering Drawings Compared with Hough Transform. *Advances in Engineering Software* 28(1):11-24

[6] Dunham JG (1986) Optimum uniform piecewise linear approximation of planar curves. *IEEE PAMI* 8(1):67-75

[7] Hough PVC (1962) A method and means for recognizing complex patterns, USA Patent 3,096,654, 1962.

[8] Hung SHY and Kasvand T (1983) Critical points on a perfectly 8- or perfectly 6-connected thin binary line. *Pattern Recognition* 16:297-284.

[9] Jaisimha MY et al. (1993) A Methodology for the Characterization of the Performance of Thinning Algorithms. In: *Proc. of 2nd ICDAR*, pp 282-286

[10] Jimenez J and Navalon JL (1982) Some Experiments in Image Vectorization. *IBM J. Res. Develop* 26:724-734

[11] Kasturi R et al. (1990) A System for Interpretation of Line Drawings. *IEEE PAMI* 12(10):978-992

[12] Lam L, Lee SW, and Suen CY (1992) Thinning methodologies - A comprehensive survey. *IEEE PAMI*:14(9):869-887.

[13] Lam L, Suen CY (1993) Evaluation of Thinning Algorithms from an OCR Viewpoint. In: *Proc. of 2nd ICDAR*, Tsukuba, Japan, pp 287-290

[14] Lee S et al. (1991) Performance Evaluation of Skeletonization Algorithms for Document Image Processing. In: *Proc. of 1st ICDAR*, France, pp 260-271

[15] Lin X et al. (1985) Efficient Diagram Understanding with Characterisuc Pattern Detection. *Computer Vision, Graphics and Image Processing* 30:84-106

[16] Liu W et al. (1995) Object Recognition in Engineering Drawings Using Planar Indexing. In: *Proc. of GREC'95*, Penn. State Univ., USA, pp 53-61

[17] Liu W, Dori D (1996) Sparse Pixel Tracking: A Fast Vectorization Algorithm Applied to Engineering Drawings. In: *Proc. 13th ICPR*, Vienna, Austria, Volume III (Robotics and Applications, pp 808-811

[18] Liu W, Dori D (1997) A Protocol for Performance Evaluation of Line Detection Algorithms. *Machine Vision Applications* 9(5/6):240-250

[19] Monagan G and Roosli M (1993) Appropriate Base Representation Using a Run Graph. In: *Proc. of 2nd ICDAR*, Tsukuba, Japan, 1993, pp 623-626

[20] Montanari U (1970) A note on the minimal length polygonal approximation to a digitized contour. *CACM* 13(1):41-47.

[21] Sklansky J and Gonzalez V (1980) Fast Polygonal Approximation of Digitized Curves. *Pattern Recognition* 12:327-331

[22] Smith RW (1987) Computer Processing of Line Images: A Survey. *Pattern Recognition* 20(1):7-15

[23] Tamura H (1978) A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint. In: *Proc. of 4th ICPR*, Kyoto, Japan, pp 715-719

[24] Vaxiviere P and Tombre K (1992) Cellestin: CAD Conversion of Mechanical Drawings. *IEEE Computer* 25(5): 46-54

[25] Vaxiviere P and Tombre K (1995) Subsampling: A Structural Approach to Technical Document Vectorization. In: *Shape, Structure and Pattern Recognition*, eds. Dori D and Bruckstein A, World Scientific, 1995, pp 323-332

[26] Yoo J-Y et al. (1998) Information Extraction from a Skewed Form Document in the Presence of Crossing Characters. In: *Graphics Recognition--Algorithms and Systems*, eds. K. Tombre and A. Chhabra, *Lecture Notes in Computer Science*, Vol. 1389, pp139-148, Springer, April, 1998