Line Net Global Vectorization: an Algorithm and Its Performance Evaluation

Jiqiang Song[1], Feng Su[1], Jibing Chen[1], Chiewlan Tai[2], and Shijie Cai[1]

[1] Department of Computer Science of Nanjing University, Nanjing, China
[2] Department of Computer Science of HongKong University of Science and Technology, Hong Kong

Abstract

In this paper, an efficient global algorithm for vectorizing line drawings is presented. It first extracts a seed segment of a graphic entity from a raster image to obtain its direction and width, then tracks the pixels under the guidance of the direction so that the tracking can track through junctions and is not affected by noise and degradation of image quality. Thus, an entity will be vectorized in one step without postprocessing. The relations among lines are also used to realize the continuous vectorization of a line net. The speed and quality of vectorization are greatly improved with this algorithm. The performance evaluation is carried out both by theoretical analysis and by experiments. Comparisons with other vectorization algorithms are also made.

1. Introduction

Vectorization, i.e. raster-to-vector conversion, is widely used in the area of document analysis and recognition as the fundamental step for high-level interpretation of document image. Most vectorization methods divide the conversion process into two steps: crude vectorization and postprocessing. They extract as many as possible line segments without junctions from raster image during the crude vectorization step, then extend and combine the line segments into exact graphic entities, such as straight lines, arc, and curve, during the postprocessing step.

The main reason for this division of two steps is that these vectorization methods cannot extract a graphic entity in one step if there are some complicated junctions on it. Typical examples of two-step vectorization are skeletonization based methods, such as thinning based algorithm [1], contour based algorithm [2] and RLE based algorithm [3]. Since the skeletonization is always based on morphology analysis, noise and junctions cause the distortion of segments, which leads to the failure or lower accuracy in the postprocessing step (Fig.1). Another weakness is that when the number of segments is large, analyzing the junctions and searching for collinear segments of an original entity is time-consuming. These methods are inefficient for vectorizing real-life scanned raster image of engineering drawings with mixed text/graphics contents.

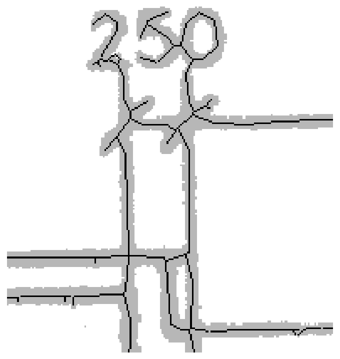


Figure 1. **Result of skeletonization**

The Sparse Pixel Vectorization (SPV) algorithm [4] - a pixel tracking algorithm – is another example of two-step vectorization. SPV uses a proper scan line interval and a tracking step to enhance the performance of vectorization. However, when the size of the junction is larger than the tracking step or when the quality of the scanned image is poor, an original entity will be broken up into several pieces (Fig.2). So SPV still need postprocessing.

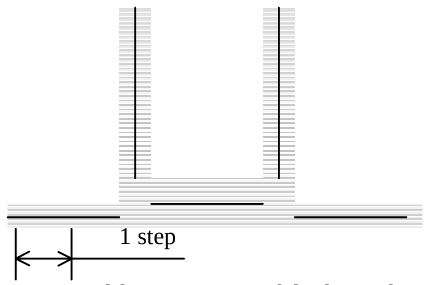


Figure 2. **Tracking stops at big junctions by SPV**

In this paper, we present a *Line Net Global* (LNG) vectorization algorithm for vectorizing line drawings. LNG assures that an original entity is vectorized in one step so that postprocessing is no longer required. To understand the advantages of LNG algorithm clearly, we make some comparisons with other algorithms during the description of LNG.

2. The line net global vectorization algorithm

We define a connected component of line entities in a raster image as a *Line Net* (LN), which usually expresses a component with an engineering meaning in a line drawing. Figure 3 shows a simple example of LN. The LNG algorithm performs an LN-oriented vectorization based on the global information in the raster image.

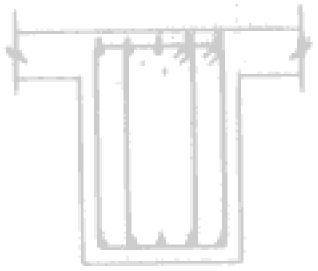
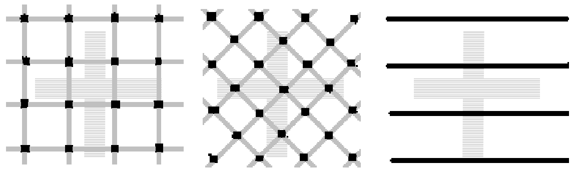


Figure 3. **A line net**

2.1. Scanning image by interlaced mesh

To reduce redundant accesses to the raster image so as to speed up vectorization, many algorithms use a mesh for sampling. However, if no mesh dot encounters the area of an entity, the entity will be missed. Figure 4 shows three different types of mesh. LNG scans the image using the interlaced mesh, while SPV using the equidistant scan lines. Let T be the interval between two adjacent scan lines in SPV and the standard mesh size. It is easy to calculate the sampling rate (SR) of each mesh type.


a.Standard mesh b.Interlaced mesh c.Equidistant scan lines
Figure 4. **Three types of mesh**

$SR_{std} = 1 / (T+1)^2$
$SR_{LNG} = 1 / ((T/2+1)*(T+1))$
$SR_{SPV} = 1 / (T+1)$

It can be seen from Fig.4 that, although the standard mesh has the lowest SR, it also has the largest missing chance. For all $T \in [1,H]$ (where H is the height of the image), $SR_{LNG} > SR_{SPV}$ and the interlaced mesh clearly encounters more entities than the equidistant scan lines. If a foreground pixel is encountered, the LN vectorization is started.

2.2. Tracking an entity

SPV starts tracking from any reliable medial point by a predefined tracking step, and only tracks in two directions (horizontal and vertical). For a slant line, it performs the zigzag tracking. There are two cases that SPV will fail to vectorize an entity in one step. Firstly, if the size of the junction is larger than the tracking step, tracking stops (Fig.2). Secondly, because the tracking only preserves direction of horizontal or vertical, tracking will diverge at some junctions (Fig.5). The radical reason for this failure is that the tracking is not guided by the entity direction. This is the same reason why other vectorization methods cannot track through junctions.

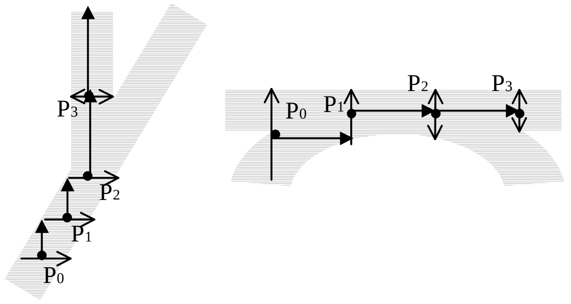


Figure 5. **Tracking diverges by SPV**

LNG overcomes this difficulty by first finding a *Seed Segment* (SS) and then making a *Direction Guided Tracking*. An SS is a regular segment of an entity that features the direction and width of the entity. The SS of a straight line must meet all the following three conditions:

(1) It is a rectangle area with an acceptable density of foreground pixel.
(2) The density must be symmetric with respect to both the long axis and short axis of the rectangle.
(3) The ratio of length and width must be acceptable.

The SS of an arc could be found from three adjacent SS' of the straight line whose perpendicular bisectors are intersecting at one point.

Extracting an SS: The first foreground pixel of an LN encountered by the scanning process is usually located at the edge of the LN, which is always a part that does not contain any feature of the entity. The following description of the algorithm is used to extract an SS from a starting point.

```
found = F;
direction[4] = {right, bottom, left, top};
while (!found && (search scope ≤
                    maximum search scope))
{
    for (i=0; i<4; i++)
    {
        Searching SS in the direction[i] of starting
            point, and striding over junctions;
            if (get a correct SS) found = T;
        }
        if (!found) enlarge the search scope;
    }
```

Direction guided tracking: An SS gives the preliminary direction and width of an entity. It is used to guide the tracking of the entity, hence the process is called *Direction Guided Tracking.* It uses the Bresenham scan conversion algorithm for generating the points on the tracking path. The tracking path starts from the long axis of the SS and extends towards two opposite directions as long as there are foreground pixels in the image at the path points and the perpendicular width runs are similar to or longer than the width of SS (Fig.6). Since the direction of the extension is determined, junctions on the path will not stop the extension. During the extension, minor direction adjustment should be done to correct possible offset or rotation of SS. This assures that SS could be sufficiently extended to cover the original entity.

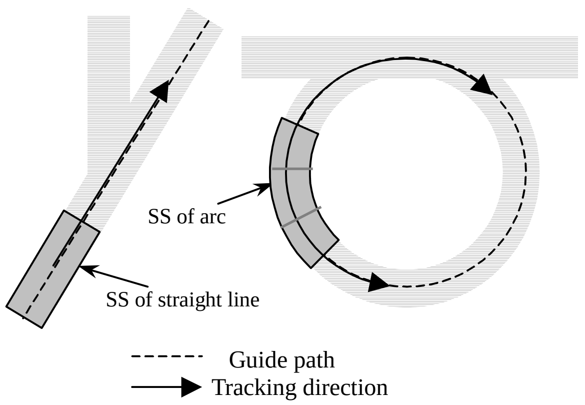


Figure 6. **Direction guided tracking by LNG**

2.3. Finding intersected entities from junctions

The recover of junction is always a difficult part of vectorization. Most current methods first extract segments without junctions, then analyze the branches of every junction to recover the connecting relation around it. However, the variety of junction and the degradation of image quality often make it too hard to analyze. In contrast, LNG uses the junction information of a vectorized entity to vectorize the entities that intersect with it.

LNG analyzes the perpendicular runs along the path of a vectorized entity to detect junctions on it. The change of the length of the perpendicular runs varies for different types of junctions (Fig.7). LNG classifies the junctions into three types: Perpendicular Junction (PJ), Oblique Junction (OJ) and Undetermined Junction (UJ). PJ and OJ indicate a perpendicular intersecting entity and an oblique intersecting entity respectively. UJ indicates an undetermined entity, such as a character or a symbol. Each type of junction is stored in a respective stack together with the information detected around it.

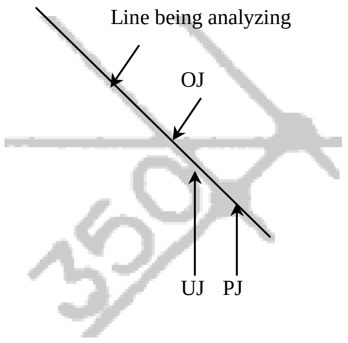
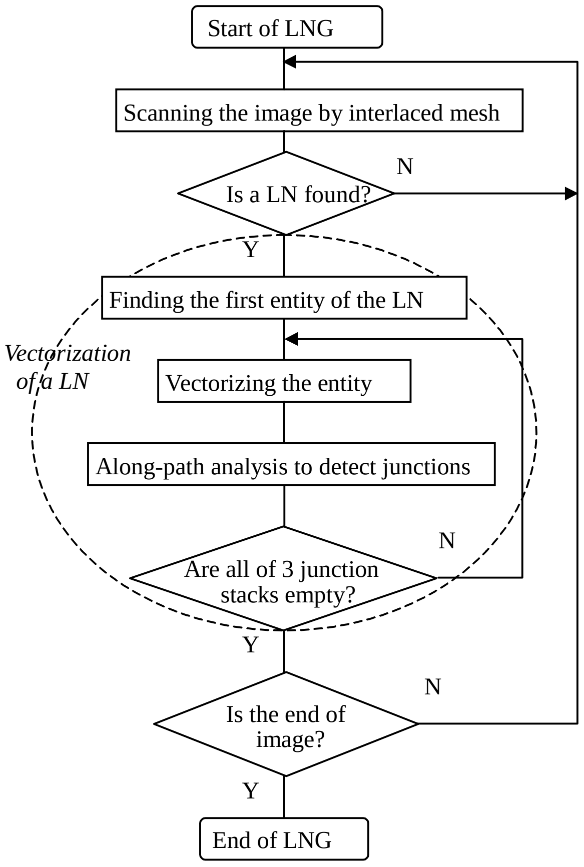


Figure 7. **Three junction types**

After an entity is vectorized, the bitmap corresponding solely to it will be erased to avoid repetition and to simplify the related junctions. LNG checks the stacks of all junction types in the order PJ→OJ→UJ. If any of the stacks is not empty, then pop a junction and track the intersecting entity; otherwise, complete the vectorization of this LN and resume scanning the raster image for a new SS. The priority of each junction type is assigned based on its efficiency to obtain the direction and width of an entity. PJ has the highest priority because the direction and width is already available. OJ has the second priority because the direction must be detected. UJ has the lowest


Start of LNG
Scanning the image by interlaced mesh
Is a LN found?
N
Y
Finding the first entity of the LN
Vectorization of a LN
Vectorizing the entity
Along-path analysis to detect junctions
Are all of 3 junction stacks empty?
N
Y
Is the end of image?
N
Y
End of LNG
Figure 8. **Flow chat of LNG**

priority because SS detection will have to be performed again. Knowing that an entity may be related to more than one junctions in an LN, this order of priority ensures that an LN will be vectorized in the fastest way.

2.4. Flow of LNG

The vectorization of an entire drawing by LNG is carried out one LN after another as shown in Figure 8.

To vectorize an entity, LNG first detects an SS, then extend it using *Direction Guided Tracking* to get the entire entity.

3. Performance evaluation

The performance of a vectorization system depends on two indices: the speed of the automatic vectorization and the cost of the interactive correction. A vectorization system is useful only if the total time of the vectorization and the correction for a drawing is much less than the time for redrawing this drawing manually with a CAD system.

3.1. Speed of Vectorization

We analyze theoretically the process of vectorizing an entire drawing by LNG, SPV and Skeletonization Based Vectorization methods (SBV) (Table.1) to evaluate the speed of each algorithm.

From the analysis, we conclude that the computing complexity of LNG is the lowest among these algorithms. Since every original entity is vectorized only once, LNG does not require a postprocessing step. Furthermore, an LN is vectorized continuously by its inherent intersecting relation without re-scanning the image. Thus, the speed of the vectorization is expected to improve. LNG has been tested with about 100 scanned images. The results of our experiments confirm the theoretical analysis. Table 2 shows the speed of LNG in three typical experiments. Image ds34 is a contest image [5], while image Stair and Details are scanned images of real engineering drawings.

Table 2. Speed of LNG (CPU: PII233, Memory: 128M)

Image	Size (Pixel^2)	Vectorized entity count	Run time (seconds)
Stairs	3358*3478	377	9.1
ds34	7200*4692	564	14.3
Details	16215*11856	1670	53.8

In comparison, for the contest image ds34, SPV spends about 25 seconds to obtain the acceptable detection rate [6], which is only the basic vectorization. If postprocessing is done to yield the exact graphic entities, the time will be much longer. Thus, LNG is faster than SPV. For the A0 size image Details, the time of LNG is much less than that of the preliminary skeletonization by skeletonization based vectorization methods.

3.2. Locating junctions

For most types of line drawings, especially for engineering drawings, locating junctions is very important to later interpretation. LNG is able to locate junctions better and faster than other algorithms.

Accuracy of the location: Since junctions are the most complicated part in a scanned image, it is hard to recover the locations of the junctions depending only on local analysis. To avoid the complex analysis of junctions,

Table 1. Contrast among LNG, SPV and SBV

	LNG	**SPV**	**SBV**
Preliminary step	No	No	Skeletonization
Vectorization of an entity	1 step: Tracking under the guidance of the direction and width of SS	2 steps: (1) Sparse pixel tracking $\rightarrow$ polyline (2) Postprocessing $\rightarrow$ graphic entity	2 steps: (1) Line fitting $\rightarrow$ polyline (2) Postprocessing $\rightarrow$ graphic entity
Polygonization to remove redundant points	No	Yes	Yes
Analysis of junctions	Simply analyzing the width run of vectorized entities	Analysis to recover the connected relationship of junctions	Complex Analysis to adjust distortion and recover junctions
Average count of accessing a foreground pixel	1 < count < 2, for accessing the pixel at junction more than 1 times	count < 1, for tracking the pixel sparsely	Depending on the approach of skeletonization, but obviously count > 2

LNG locates a junction by calculating the intersection of the related entities. Therefore, the accuracy of the locations depends on the accuracy of the entity attributes, which is more reliable than local analysis.

Speed of the location process: In other methods, it is impossible to obtain the relationship between entities while vectorizing them. After vectorization, every entity is calculated for intersections with all other entities to locate the end points. The time complexity of the intersection calculation is $O(N^2)$, where N is the number of entities. When N is large in a drawing, it is very time-consuming. LNG makes it more efficient by using an LN oriented vectorization. During the vectorization of an LN, starting from the second entity, every entity is introduced by its intersecting entity. Some end points (about 50 percent) can be located during this processing and are marked. After the vectorization of the LN, the entity with unmarked end points is tested for intersections only with other entities in the same LN. Therefore the time complexity is $O(M^2)$, where M is the number of entities in the LN. M is usually much smaller than N so that the time for locating the end points is clearly reduced.

3.3. Accuracy of entity attributes

The accuracy of entity attributes is another important aspect of the performance of vectorization algorithms. If the acceptance threshold increases, the inaccurate attributes may lead to false alarms.

All the other vectorization algorithms discussed above depend on local information. They demand that most parts of an entity be in good quality to rebuild it. Since a real scanned raster image cannot be in good quality, local algorithms often miss or break an original entity. The noise and degradation on each segment of an original entity will cause inconsistent attributes. If they can be combined successfully, the attribute of the combined entity often deviate slightly from its original bitmap in the raster image.

LNG can vectorize an entity in the presence of noise or degradation of junction because it only requires some good part of the entity, such as the SS, as a reference for direction adjustment. *Direction Guided Tracking* can track through junctions and is insensitive to noise and degradation to get the entire entity in one step. Therefore, the attributes of the entity are detected considering the entire entity so that the shape preservation of its original bitmap is guaranteed.

Even if there is no good part on an entity in an LN, chances still exist that it will be found through its intersecting entity. LNG is a global algorithm that uses not only information of one entity but also of its environment.

3.4. Cost of interactive correction

The quality of the vectorization is more important than the speed of the algorithm. The cost of interactive correction increases dramatically if there are too many correction operations such as redrawing the missed entities, deleting false entities and modifying attributes of entities. The costs of correction for the different types of errors are as following:

Missing: 1 (Adding)

Mistaken attributes: 1 (Modifying)

Breaking into *X* parts: *X*-1 (Deleting) + 1 (Modifying)

In view of the cost of interactive correction, breaking an original entity into several parts is worse than missing it or mistaking its attributes. Therefore, if the problem of breaking an original entity is solved, the cost of correction will decrease significantly.

Table 3 shows the cost of interactive correction for the forenamed experiments when the acceptance threshold is set as 0.85. To give a quantitative evaluation of the cost of the correction, we assume that adding an entity, deleting an entity and modifying an entity have the same cost.

Table 3. Cost of interactive correction by LNG

Image	Misses	False Alarms	Correction cost	Vectorized rate
Stairs	9	17	26	0.928
ds34	18	36	54	0.903
Details	29	93	122	0.925

We can conclude from the table that, LNG has a stable performance for images of various sizes. Since it does not break up an original entity, the cost of interactive correction is low.

4. Conclusion

The main features of the LNG algorithm introduced in this paper are the following: (1) it vectorizes an entity in one step under the guidance of an SS, and the SS is dynamically adjusted during tracking, so that the quality of the vectorization can be guaranteed; (2) it uses the junction relationship of entities to improve the speed of the vectorization of an LN.

With this algorithm, two-step vectorization is no longer required. The theoretical analysis and some experimental results on the performance of LNG are presented and compared with other vectorization algorithms. Our experiments confirm that LNG is an efficient vectorization algorithm for real-life scanned images. Figure 9 is an example of LNG.

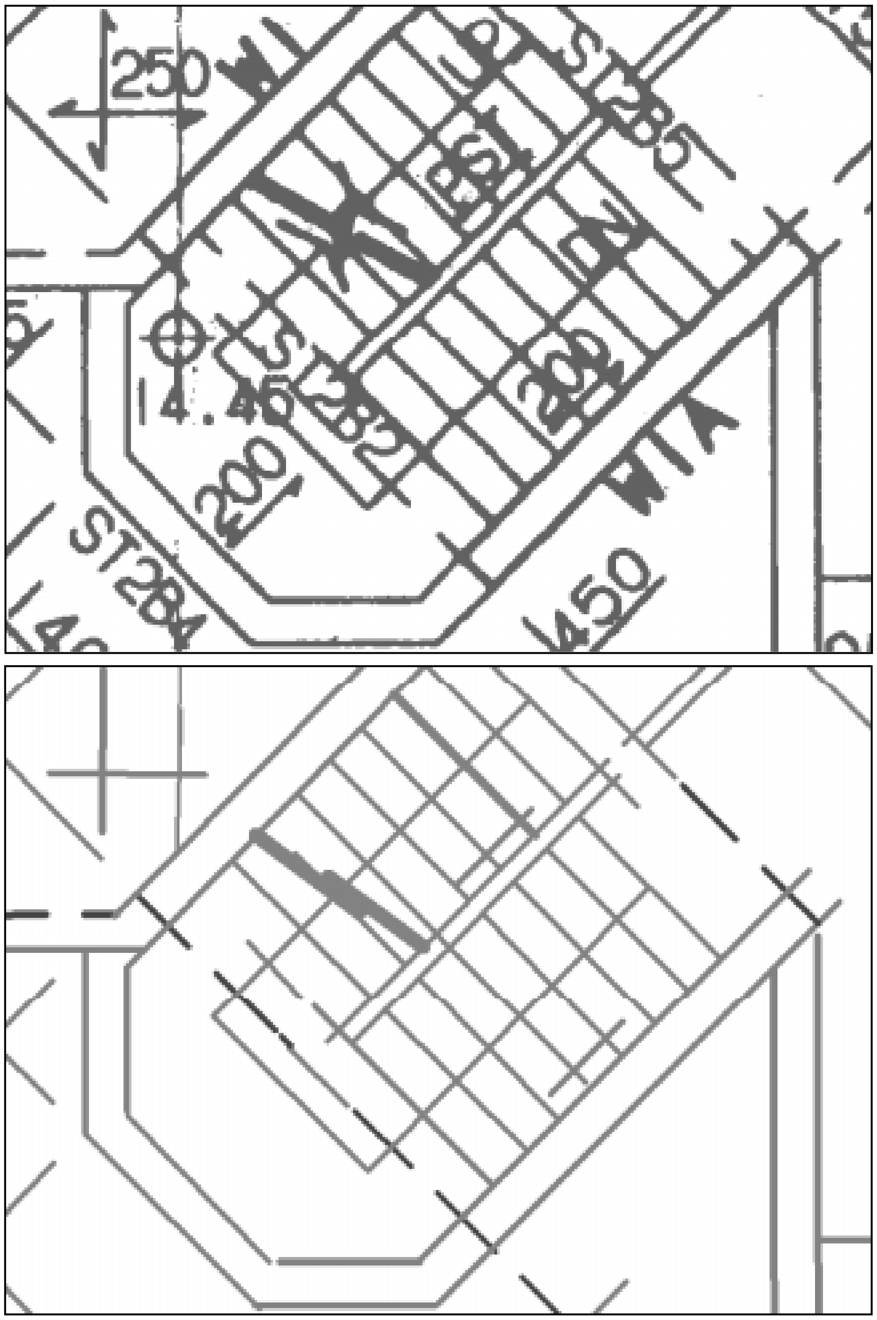


Figure 9. **Fraction of a test image (top) and line vectorization result with LNG (bottom)**

References

[1] A.Datta, S.K.Parui. “A robust parallel thinning algorithm for binary images”. *Pattern Recognition*, 1994, 27(9): 1181-1192.

[2] O.Hori and S.Tanigawa. “Raster-to-vector conversion by line fitting based on contours and skeletons”. In *Proceeding of the International Conference on Document Analysis and Recognition*, 1993: 353-358.

[3] S.D. Zenzo, L. Cinque, and S, Levialdi. “Run-Based Algorithms for Binary Image Analysis and Processing”. *IEEE Transactions on PAMI*, 1996, 18(1): 83-89.

[4] Dori D, and Liu W. “Sparse Pixel Vectorization: An Algorithm and Its Performance Evaluation”. *IEEE Trans. on PAMI*. 1999, 21(3): 202-215.

[5] Chhabra A, and Philips I. Web pages for the Second International Graphics Recognition Contest. http://graphics.basit.com/iapr-tc10/contest.html

[6] Liu W, Wang X, Tang L, and Dori D. “Impact of Sparse Pixel Vectorization Algorithm Parameters on Line Segmentation Performance”. In *Proceeding of the third IAPR international Workshop on Graphics Recognition*, 1999: 323-330.