
PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 38 (2005) 539–552

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

A Hough transform based line recognition method utilizing both parameter space and image space

Jiqiang Song*, Michael R. Lyu

Department of Computer Science & Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, P.R. China

Received 19 September 2004; accepted 20 September 2004

Abstract

Hough Transform (HT) is recognized as a powerful tool for graphic element extraction from images due to its global vision and robustness in noisy or degraded environment. However, the application of HT has been limited to small-size images for a long time. Besides the well-known heavy computation in the accumulation, the peak detection and the line verification become much more time-consuming for large-size images. Another limitation is that most existing HT-based line recognition methods are not able to detect line thickness, which is essential to large-size images, usually engineering drawings. We believe these limitations arise from that these methods only work on the HT parameter space. This paper therefore proposes a new HT-based line recognition method, which utilizes both the HT parameter space and the image space. The proposed method devises an image-based gradient prediction to accelerate the accumulation, introduces a boundary recorder to eliminate redundant analyses in the line verification, and develops an image-based line verification algorithm to detect line thickness and reduce false detections as well. It also proposes to use pixel removal to avoid overlapping lines instead of rigidly suppressing the $N \times N$ neighborhood. We perform experiments on real images with different sizes in terms of speed and detection accuracy. The experimental results demonstrate the significant performance improvement, especially for large-size images.

Keywords: Line recognition; Hough transform; Large-size image; Parameter space; Image space; Line verification

1. Introduction

Graphical element recognition is fundamental to various image-understanding applications. Hough Transform (HT) is a powerful tool for finding parameterized shapes in digital images [1]. We can easily find a large number of HT-based graphics recognition methods in literature [2–16], which are capable of detecting straight lines, circles, ellipses and other curves in both binary and grayscale images. HT-based methods are robust over different image qualities since HT converts a difficult global detection problem in image space into a more easily solvable peak detection problem in the HT parameter space; therefore, they can deal with noise, degradation, and partial disconnection, even in complicated background. However, HT-based methods are seldom applied to the domain of engineering drawings because of two common limitations:

(1) Their reported applications are limited to small-size images. The attempts of applying HT to large-size images are usually discouraged since when the image size expands, the processing time and the memory cost grow

* Corresponding author. Hong Kong Applied Science & Technology Research Institute, 1st Floor, Photonics Center, 2 Science Park East Avenue, Hong Kong Science Park, Shatin, N.T., Hong Kong SAR, P.R. China. Tel.: +852 3406 2714; fax: +852 3406 2801.
E-mail address: songjq@ieee.org (J. Song).

0031-3203/$30.00
doi:10.1016/j.patcog.2004.09.003

rapidly, and the random aligned noises become noticeable. Unfortunately, engineering drawings usually come with quite large sizes.

(2) They do not detect the line thickness, which is critical to some applications, say, understanding engineering drawings. Some methods assume the line thickness is constant in an image and then take it as an input parameter to enhance the performance [13,16]. However, this assumption is not true for most engineering drawings.

One may think that it is not necessary to apply HT-based methods to engineering drawings since there are already many graphics recognition methods in this domain. In fact, according to our study, the popular graphics recognition methods for engineering drawings, including skeletonization-based methods [17,18], contour-based methods [19] and pixel-tracking methods [20,21], mainly depend on the pixel-level connectivity of graphic elements. As a result, they handle good quality images well, but their performance drops significantly when the image quality is too poor to keep the connectivity. HT is no doubt the best choice under this circumstance. Therefore, this paper proposes a new HT-based method which handles large-size images more efficiently and can detect line thickness.

Besides the inherently high computation complexity of HT, we believe the limitations of existing HT-based methods partly come from their only working in the HT parameter space. With the image size expanded, the number of feature points contributing to an HT parameter increases dramatically, leading to much larger memory cost and much longer processing time for line verification. Moreover, since the feature points are only a small portion of all pixels in an image, it is impossible to detect the thickness of a line. Therefore, this paper proposes to utilize the image space to overcome these limitations. The experiments demonstrate that the proposed HT-based line recognition method is very efficient and accurate, especially for large-size images, and it can detect line thickness as well.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the proposed method in detail. The experimental results are reported in Section 4. Finally, Section 5 draws our conclusions.

2. Related work

The standard HT for straight line is depicted by Eq. (1), where (x, y) denotes a point in the Cartesian coordinates, i.e., the image space, and (r, θ) represents its parameter in the polar coordinates, also called the HT parameter space. All points on the same line in the image space will intersect at one point in the HT parameter space.

$$r = x \times \cos\theta + y \times \sin\theta. \quad (1)$$

Generally, using HT to recognize lines in images consists of three steps: first, perform HT on feature points in the image space and accumulate hits for each parameter in the HT space; second, detect peaks in the HT parameter space; and third, verify the line indicated by the peak parameter.

The first step has been well investigated so far. The conventional HT treats all angles equally, which results in heavy computation, huge parameter space, and less-salient peaks. Consequently, current research interests focus on how to select angles to accelerate the transform safely and efficiently. An abundant number of improved HT methods, e.g., gradient-based HT [3,4], randomized HT [11], probabilistic HT [9,12] and sampling HT [10], have been proposed to accelerate the accumulation and to highlight the peaks.

The peak detection methods can be divided into two classes: local peak detection methods and global peak detection methods. The former class, e.g., [5], is the common way, which finds the local maxima within an $N \times N$ neighborhood over the HT parameter space, where N is very critical. Using a larger N will suppress some real lines, while using a smaller N will yield overlapping lines, i.e., repetitive detections. To avoid the dilemma of local peak detection, Princen et al. [8] proposed an iterative global peak detection method, which first selects the globally highest peak in the HT parameter space, then removes those feature points contributing to this peak, and finally repeats the HT accumulation and the global peak detection. This method is more robust than clearing a rigid-size neighborhood, but it is very time-consuming for a large-size image due to iterative accumulations. In fact, it is only applied to the sub-image level in [8].

The line verification step has two purposes: (1) to determine the exact parameters of line segments, including starting point, end point and thickness, along the line indicated by the detected peak, and (2) to eliminate the random aligned feature points. The common method for line verification [15] is sequentially checking the connectivity of feature points within the narrow strip area determined by the peak parameter (r, θ), the quantization interval Δr, and the sampling interval $\Delta\theta$. Since the line equation is calculated frequently and the contributing feature points are searched iteratively, this step may be more time-consuming than the previous two steps for large-size images containing numerous lines. Moreover, since existing methods only depend on feature points, they cannot detect the line thickness. However, the improvement on this step is seldom addressed because small images are always considered.

3. Line recognition method

The idea of this method is to utilize the handy image space to help accelerate the whole process and detect the line thickness. As engineering drawings are usually scanned, stored and processed in a binary format, we assume that the image discussed in this paper is monochromatic (i.e., black for foreground and white for background). We predefine two thresholds, T_{min} and T_{max}, to indicate the thinnest

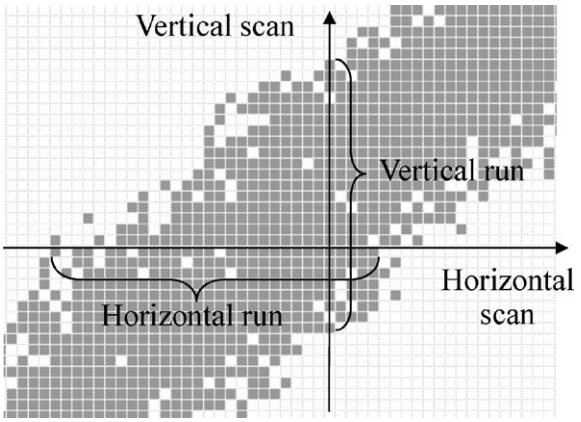

Fig. 1. Orthogonal run length scanning.

and the thickest acceptable line thickness, respectively. The proposed method consists of three major steps: HT accumulation (voting), peak selection and sorting, and line verification.

3.1. Feature point selection

Considering the efficiency of a voting process, we select only a small portion of pixels in the image, called feature points, to participate the voting. We define the feature points as the medial points of line-like areas. The feature points are obtained by orthogonal run-length scanning, described as follows. During the scan, we assume one-pixel-long white gaps will not break a black run to tolerate the degraded image quality.

(1) Horizontally scan the image stepping S_h rows, where S_h is the image height divided by 3000. For a horizontal black run, if its length is between T_{min} and T_{max}, the midpoint of this run is selected as a horizontal feature point, as shown in Fig. 1.
(2) After the horizontal scanning finishes, scan the image vertically stepping S_v columns, where S_v is the image width divided by 3000. For a vertical black run, if its length is between T_{min} and T_{max}, we further calculate the length of the horizontal run passing the midpoint of this vertical run. If the length of horizontal run is also between T_{min} and T_{max}, this midpoint will be skipped because it has already been used in the horizontal scanning; otherwise, this midpoint is selected as a vertical feature point, as shown in Fig. 1.

By the above feature point selection algorithm, the points of non-line-like areas and the non-medial points are excluded from voting, which enhances the localization accuracy of the resulting lines and accelerates the voting process.

3.2. Gradient prediction

In the conventional HT, a feature point votes for all angles. This causes unnecessary computations and memory costs. Moreover, it also produces false peaks in the parameter space due to random aligned feature points. Following the idea of gradient HT, we perform an image-based gradient prediction for each feature point before voting.

According to Section 3.1, we have classified a feature point into either a horizontal feature point or a vertical feature point. For a horizontal feature point, we calculate the lengths of vertical, left diagonal, and right diagonal runs passing the feature point, as shown in Fig. 2a. For a vertical feature point, we calculate the lengths of horizontal, left diagonal, and right diagonal runs, as shown in Fig. 2b. Then, the direction with the longest run length, denoted by *L_Dir*, is the possible line direction in the image space, which is perpendicular to the predicted gradient direction in the parameter space, i.e., θ. The voting angle range for this feature point is finally determined by the predicted gradient direction, as shown in Eq. (2). Fig. 2c illustrates that the voting angle ranges of neighboring gradient directions are partly overlapped. This is to tolerate the prediction error. The extent of overlapping can be reduced to speed up the voting process. According to Eq. (2), the voting angle range spans 60°, which is one third of the range of conventional HT. Of course, one can check more directions to achieve more accurate gradient prediction; however, checking any other directions in the image space will be much slower than these four directions.

$$\textit{Voting angle range (in degree)} = \begin{cases} 0 \leqslant \theta \leqslant 30 \text{ Y} \\ \quad 150 \leqslant \theta \leqslant 180, & L\_Dir = \textit{Vertical} \\ 60 \leqslant \theta \leqslant 120, & L\_Dir = \textit{Horizontal} \\ 15 \leqslant \theta \leqslant 75, & L\_Dir = \textit{Right diagonal} \\ 105 \leqslant \theta \leqslant 165, & L\_Dir = \textit{Left diagonal}. \end{cases} \quad (2)$$

3.3. Thickness-weighted voting

Since only using feature points to vote does not conform to the human perception that thick lines are more salient than thin lines, and moreover, this will lower the signal-to-noise ratio. Therefore, we employ the thickness-weighted voting. In the previous processing, we have obtained the lengths of runs in four directions passing a feature point. Denote the lengths of horizontal, vertical, left diagonal, and right diagonal runs by *H_Len*, *V_Len*, *LD_Len*, and *RD_Len*, respectively. The voting weight for a feature point is determined by Eq. (3), which uses the length of the run perpendicular to the line direction to approximate the line thickness:

$$\textit{Voting weight} = \begin{cases} H\_Len, & L\_Dir = \textit{Vertical} \\ V\_Len, & L\_Dir = \textit{Horizontal} \\ LD\_Len, & L\_Dir = \textit{Right diagonal} \\ RD\_Len, & L\_Dir = \textit{Left diagonal}. \end{cases} \quad (3)$$

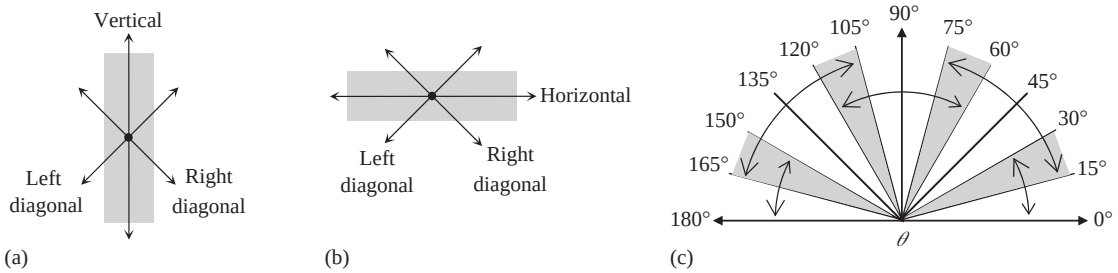

Fig. 2. Gradient prediction for feature points.

3.4. Boundary recorder

For the efficiency of processing large-size images, the proposed method does not store the feature points contributing to each parameter. Some existing methods also do the same way; however, one important reason for their time-inefficiency is that they do not know which part of the strip area contains feature points. Thus they have to either recalculate all feature points with the known θ to pick out those with the same (or similar) r, or check every position within the strip area in the image. Obviously, neither way is fast for a large-size image. Usually, only a small part of the strip area contains the feature points. According to this fact, we add a boundary recorder to each parameter cell to record the *minimum scope* that contains the feature points contributing to this parameter.

The boundary of each parameter consists of two feature points, called "upper boundary" and "lower boundary", which enclose all other feature points contributing to this parameter. Since the dimension of parameter space grows rapidly when the image size becomes large, one should be careful in adding any byte to the parameter cell. According to Eq. (1), given r and θ, one dimension of the image coordinates can be calculated from another dimension. So we only need to record one dimension of the coordinates in the parameter cell. The choice of X- or Y-dimension coordinates depends on θ (Fig. 3). When $45° < \theta < 135°$, the line in the image space is nearly horizontal, so X-dimension is chosen to record the boundary; otherwise, Y-dimension is chosen. The initialization and the recording process of the boundary recorders are described by the following C pseudo-codes, where $\min(a, b)$ returns the smaller one between a and b, and $\max(a, b)$ returns the bigger one.

```
Initialization: for all parameter cells
    Param[r][θ].accumulator = 0;
    Param[r][θ].lower_boundary = max(image_height, image_width);
    Param[r][θ].upper_boundary = min(0, 0);

Recording: when a feature point (x, y) contributes to a parameter (r, θ)
    Param[r][θ].accumulator + = voting_weight;
    IF (45° < θ < 135°) {
        Param[r][θ].lower_boundary = min(Param[r][θ].lower_boundary, point.x);
        Param[r][θ].upper_boundary = max(Param[r][θ].upper_boundary, point.x);
    }ELSE {
        Param[r][θ].lower_boundary = min(Param[r][θ].lower_boundary, point.y);
        Param[r][θ].upper_boundary = max(Param[r][θ].upper_boundary, point.y);
    }
```

Consider an image of size $W \times H$ (width by height), whose range breadth of r is $[-W, \sqrt{H^2 + W^2}]$ for $0° \leqslant \theta \leqslant 180°$. The memory requirement for the 2D HT parameter space can be calculated as follows:

$$\textit{Memory Requirement} = \frac{(\sqrt{H^2 + W^2} + W)}{\Delta r} \times \frac{180}{\Delta\theta} \times \textit{size of (param cell)} \quad \text{bytes.}$$

We choose $\Delta r = 2$ and $\Delta\theta = 1°$ to keep both the accuracy of direction and the clustering effect. The parameter cell contains one accumulator, usually an integer (4 bytes), and two boundary recorders that are short integers (2 bytes for each), totally 8 bytes. For a large image of A0-size engineering drawing scanned with 300 dpi, which is fine enough to digitize a line as thin as 0.003 in, W is about 14,000 pixels

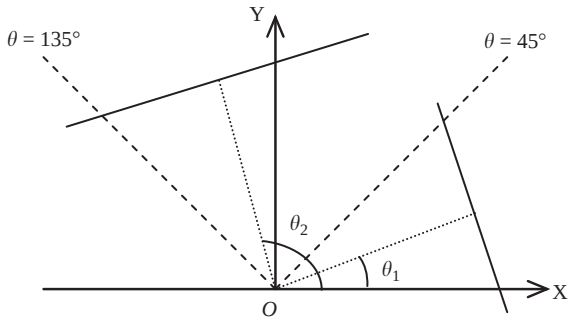

Fig. 3. Choice of the recording dimension.

and H about 9900 pixels. Then, the memory requirement for the parameter space is approximately 23.8 MB, which is obviously affordable for modern computers.

3.5. Peak detection

In this step, we introduce a threshold L_{min} standing for the minimum acceptable line length. The threshold for peak height in the parameter space are derived by Eq. (4), which represents a line with both the minimum length and the minimum thickness, considering the scanning steps:

$$T_{peak} = \frac{T_{min} \times L_{min}}{\max(S_h, S_v)}. \quad (4)$$

After all feature points have been transformed, we detect all peaks in the HT parameter space. A parameter can be identified as a peak if and only if its accumulated value is both larger than T_{peak} and the maximal in the 5×5 neighborhood of the parameter. Meanwhile, all detected peaks are sorted into a peak list in descending order of accumulated value. The line verification will begin from the head of the peak list; thus, it can also take advantage of the global peak. The pixels corresponding to a verified line segment are erased from the image immediately, i.e., turning them from black to white. Since the line verification is based on the image space but not on feature points, the overlapping lines are successfully avoided by erasing the already-verified line segments. Thus, it does not need the re-accumulation after processing each peak and it is much faster than the iterative global peak detection method [8].

3.6. Line verification

Owing to the recorded boundary information, we only need to analyze the valuable part in the image space determined by the peak parameter to find the evidence of line segments and detect the line thickness of each segment.

According to the boundary recording process defined in Section 3.4, the image coordinates of two boundary points can be calculated easily, denoted by P_{lb} and P_{ub}. The line verification analyzes the pixels along the straight-line direction from P_{lb} to P_{ub} sequentially. To avoid the heavy computation in solving the line equation frequently, we adopt the off-the-shelf rasterization method—Bresenham algorithm for straight line [22], which generates a straight-line path point with at most three additions—to generate the eight-connected path points from P_{lb} to P_{ub}, denoted by P_i ($i = 1 \ldots n$), where $P_1 = P_{lb}$ and $P_n = P_{ub}$. The detailed image-based line verification algorithm is as follows, where G_{max} stands for the maximum acceptable gap length:

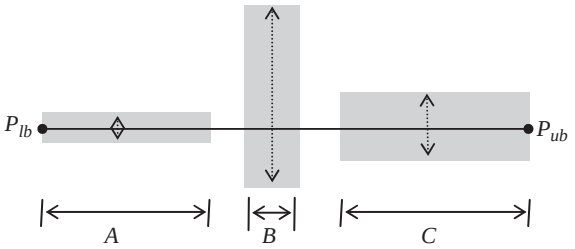

Fig. 4. Line verification.

```
gap_count = 0; start_pos = 1;
FOR (i = 1 TO n) {
  IF (Pi is black) {
    gap_count = 0;
    IF (start_pos = = 0) start_pos = i;
    IF (i = =n) {
      IF (i-start_pos ⩾ Lmin) {
        IF (VerifySegment(start_pos, i) = = True) {
          Accept this segment;
          Erase the pixels of this segment;
        }
      }
    }
  }ELSE {
    gap_count + = 1;
    IF (gap_count = = Gmax or i = = n) {
      IF (i-gap_count-start_pos ⩾ Lmin) {
        IF (VerifySegment(start_pos,
          i-gap_count) = = True) {
          Accept this segment;
          Erase the pixels of this segment;
        }
      }
      start_pos = 0;
    }
  }
}
```

This algorithm checks all black segments from P_{lb} to P_{ub} that are longer than L_{min} and do not contain gaps longer than G_{max}. For example, there are three valid black segments in Fig. 4, denoted by A, B and C, respectively. For each segment, it calls VerifySegment(*start*, *end*) to verify

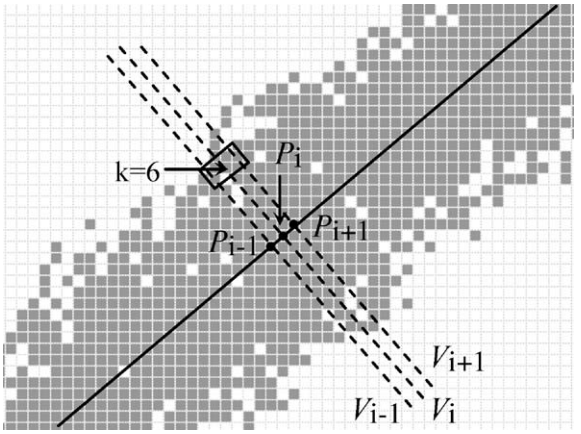

Fig. 5. Local line thickness detection.

the validity of the segment by checking the line thickness, displayed as arrowheaded dashed lines in Fig. 4. The line thickness of a segment is voted by all local line thickness detected at each black P_i ($i = start \ldots end$). If the line thickness is larger than T_{max}, this segment may be the cross section of a line in other direction, e.g., the segment *B*, so that it should be rejected. Since this algorithm does not depend on a single threshold for the decision purpose, it can distinguish between true lines and a random alignment of feature points correctly. Moreover, it can detect more than one line segment with different line thickness in the same straight-line direction.

Considering the degraded quality of the image, we develop a missing-pixel-tolerant approach to detect the local line thickness. For each P_i, we use P_{i-1} and P_{i+1} (if available) to help the decision (Fig. 5). Let V_i denote the straight-line path passing P_i and perpendicular to the line $P_{lb}P_{ub}$. V_i is also generated by the Bresenham algorithm. $V_i(k)$ ($k = -T_{max} \ldots + T_{max}$) is the point on the V_i path with k steps away from P_i, particularly, $V_i(0) = P_i$. The detection starts from $k = 0$ and increases k by 1 iteratively until the number of black pixels among $Vj(t)$ ($j = i - 1 \ldots i + 1$, $t = k \ldots k + 1$) is less than 4, and then records the stopped k as k_{max}. Next, it decreases k from 0 by 1 with the same criteria to get k_{min}. Finally, the local line thickness is calculated as $k_{max} - k_{min} + 1$. This approach can detect correct line thickness from degraded quality images as well as high quality images. The verified line segments will be stored with three parameters: starting point, end point, and line thickness.

3.7. Line erasing

After all line segments contributing to a peak have been verified, the pixels corresponding to these line segments should be erased from the image immediately to avoid overlapping lines. It is easy to erase the pixels of a line segment by erasing a rectangular area determined by the parameters (the long axis is from the starting point to the end point, and the length of short axis equals the line thickness). This is correct for an isolated line segment. However, if there are other under-detected line segments intersecting this line segment (Fig. 6a), their intersection parts will also be erased so that the under-detected line segments will be separated (Fig. 6b). This problem also exists in those line verification methods based on removing feature points.

Instead, we employ an intersection-preserving approach based on detecting the trends of branches at the intersection [21]. It simply erases those parts whose local line thickness are less than or similar to the thickness of this line segment as rectangular areas. For other parts, i.e., intersection parts, it detects the trend of branches toward the line segment to approximate the borders for erasing and for preserving purposes (Fig. 6c). This approach only erases the pixels belonging to the verified line segments. Thus, the overlapping lines are avoided, and the entirety of under-detected line segments is kept as well.

3.8. The overall paradigm

Based on the above key techniques, Fig. 7 shows the overall paradigm of the proposed HT-based line recognition method. The whole processing is divided into three steps by two vertical dashed lines. The first step performs feature point selection and gradient prediction on the image space,

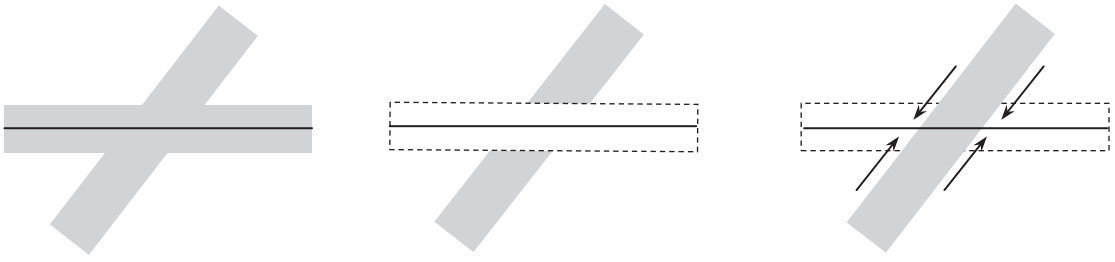

Fig. 6. Erasing the pixels corresponding to the horizontal line segment.

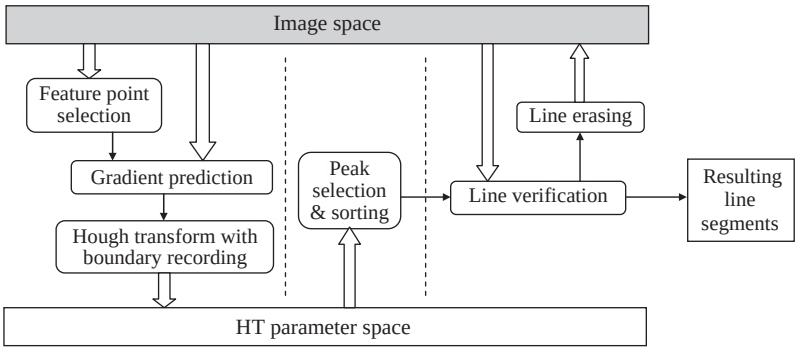

Fig. 7. Overall paradigm of the proposed HT-based line recognition method.

and then generates the HT parameter space. The second step selects peaks from the parameter space and sorts them in descending order of accumulated values. The last step verifies the line segments according to the image space, erases the verified line segments, and finally produces the collection of resulting line segments.

4. Experiments

In our experiments, the testing images consist of five images [23] scanned from a real engineering drawing, which contains straight lines of various thickness and texts of various font sizes. Fig. 8 illustrates the thumbnails of these images. The image size ranges from A4 to A0, as shown in Table 1. All experiments were performed on a PC with P4 2.4G CPU and 1G RAM.

4.1. Conventional HT-based method

To demonstrate the necessity of using the image space, we implement the conventional HT-based line recognition algorithm to test its performance on large-size images. The conventional HT-based method uses only feature points to verify lines. It can be implemented in two ways: by recording feature points or by re-voting feature points. The algorithms implemented by these two ways are denoted by CHT-recording and CHT-revoting, respectively. They are optimized for the maximized speed.

The CHT-recording algorithm selects feature points by the orthogonal run-length scanning described in Section 3.1. Each feature point votes for every direction. The feature points contributing to an HT parameter are recorded with this parameter in the parameter space. Local maximal points in the parameter space are detected as peaks. For each peak, the line verification checks the connectivity of contributing feature points to recognize line segments. The experimental result of this implementation is shown in Table 2, which reveals two important facts:

(1) It cannot handle large-size images since recording the contributing feature points is very space-consuming. In this experiment, it cannot even complete the processing of images A2, A1, and A0 due to the huge memory costs. For the image A4, the memory cost is 870 MB, which is affordable by the operating system. Therefore, the processing time is reasonable. However, this algorithm consumes 1720 MB for the image A3. Since the memory cost is far beyond the physical memory of the operating system, the processing time is excessively long due to the frequent swapping between the memory and the hard disk. This can be evidenced by the observation that the average CPU usage during the swapping is lower than 10%.
(2) For large-size images, the peak detection and the line verification are much more time-consuming than the HT accumulation. Taking the experiment on image A4 as an example, the time for HT accumulation is less than one eighth of the total processing time.

Instead of recording the feature points contributing to every parameter, the CHT-revoting algorithm re-votes all feature points and collects the feature points contributing to a peak during the line verification. Since this implementation keeps only a global list of feature points, the memory cost is significantly reduced. Therefore, it can handle all these five testing images, although the speed is quite slow (Table 3). The reason for the time-inefficiency is that the re-voting mechanism makes the line verification much slower comparing to the HT accumulation. For the largest image A0, the total processing time is approximately 30 times of the time for HT accumulation.

On the other hand, the detection accuracy of both implementations of CHT is not satisfactory because of many overlapping lines and unavailable line thickness information.

4.2. The proposed method

The proposed method is implemented with the following threshold setting: $T_{min} = 0.005 \times R$, $T_{max} = 0.1 \times R$, $L_{min} = 0.15 \times R$, and $G_{max} = 0.03 \times R$, where R is the scan resolution of testing images. In this experiment, R is

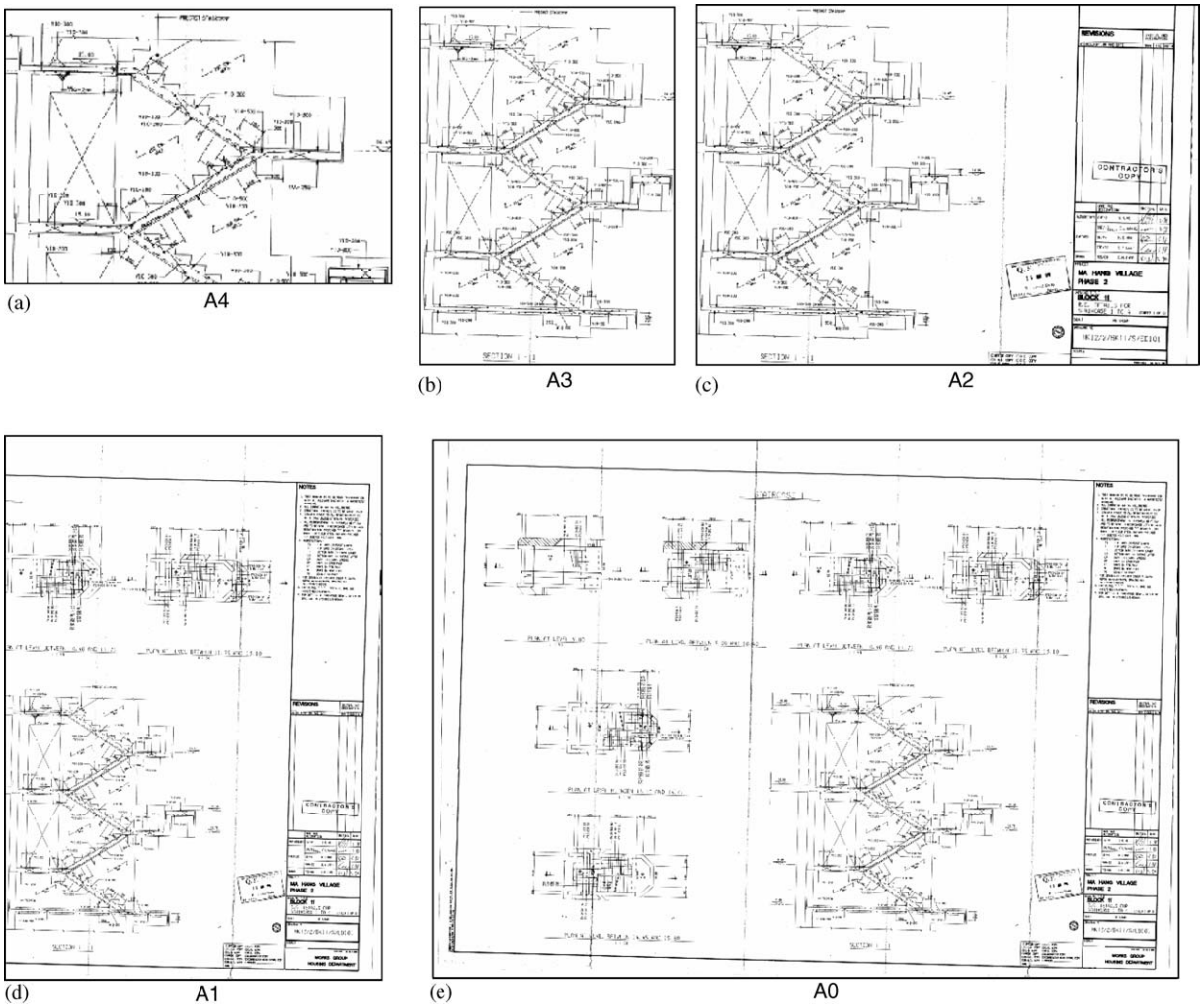

Fig. 8. Five testing images.

Table 1
Information of testing images

Image	A4	A3	A2	A1	A0
Size (Pixel^2)	3533 × 2527	4990 × 3537	6959 × 4990	10078 × 6959	13783 × 10078
Storage (MB)	1.05	2.11	4.18	8.41	16.78

Table 2
Performance of the CHT-recording algorithm

Image	Memory cost (MB)	Total processing time	Time for HT accumulation
A4	870	133 s	16 s
A3	1720	8 h 41 min 40 s	81 s
A2, A1, A0	N/A		

300 dpi; therefore, these four thresholds are 2, 30, 45 and 9 pixels, respectively. In contrast with the conventional HT-based method, the proposed method handles all testing images efficiently and reduces both the memory cost and the processing time significantly.

We evaluate the memory cost, processing time, and detection accuracy of our proposed method. The total processing time for line recognition, denoted by $\boldsymbol{Time_{all}}$, is further divided into three parts: $\boldsymbol{Time_{vote}}$, $\boldsymbol{Time_{sort}}$, and $\boldsymbol{Time_{verify}}$, which stand for the time spent on HT accumulation, on

Table 3
Performance of the CHT-revoting algorithm

Image	Memory cost (MB)	Total processing time (s)	Time for HT accumulation (s)	*DR* (%)	*FR* (%)	Detection accuracy (%)
A4	8.9	46	2	67.4	22.5	72.5
A3	14.3	122	6	65.2	23.6	70.8
A2	20.3	241	9	68.7	31.1	68.8
A1	20.5	258	10	59.8	38.6	60.6
A0	29.5	459	15	60.3	44.9	57.7

peak selection and sorting, and on line verification, respectively. The detection accuracy is measured in two aspects: **Detection Rate** (*DR*) and **False Rate** (*FR*). *DR* indicates how many percent of ground-truth lines is correctly detected, while *FR* shows how many percent of detected lines is wrong. The evaluation is based on both the localization accuracy and the thickness precision. The ground-truth lines in each testing image are manually identified. A ground-truth line (G_i) is described by two endpoints, length (L_g^i) and thickness (T_g^i). Accordingly, the length and thickness of a detected line (D_i) are denoted by L_d^i and T_d^i. Considering the visual importance of longer lines, *DR* and *FR* are defined as length-weighted forms.

$$DR = \frac{\sum_i DR_i}{\sum_i L_g^i} \times 100\%, \text{ where}$$

$$DR_i = \begin{cases} 0, & \textit{match not found} \\ L_d^i \cdot e^{-\frac{Di}{2 \cdot T_g^i}} & \\ \quad \cdot \left(1 - \frac{|T_d^i - T_g^i|}{2 \cdot T_g^i} \cdot w\right), & \textit{otherwise} \end{cases}$$

$$FR = \frac{\sum_i FR_i}{\sum_i L_d^i} \times 100\%, \text{ where}$$

$$FR_i = \begin{cases} 1, & \textit{match not found} \\ L_g^i \cdot \left[1 - e^{-\frac{Di}{2 \cdot T_g^i}}\right. & \\ \quad \left. \cdot \left(1 - \frac{|T_d^i - T_g^i|}{2 \cdot T_g^i} \cdot w\right)\right], & otherwise. \end{cases}$$

The calculation of *DR* searches for the best matched detected line for each ground-truth line within a predefined searching scope. If no matched detected line is found, a zero is accumulated; otherwise, a product of three terms regarding length, endpoint localization and thickness is accumulated. In the second term for endpoint localization, D_i is the sum of distance of two detected endpoints to the ground-truth line. In the third term for thickness, w is the importance factor of thickness, ranging from 0 to 1. In our application, the line thickness is critical for engineering drawing interpretation; thus, w is set to be 1. Similarly, the calculation of *FR* searches for the best matched ground-truth line for each detected line. Note that each ground-truth line can only be matched once; therefore, repetitive detection will cause false alarms. A good line detection algorithm should achieve a high *DR* and a low *FR*. Consequently, we can define a unified measurement of the detection accuracy as follows:

$$\textbf{\textit{Detection Accuracy}} = 0.5 \times DR + 0.5 \times (1 - FR).$$

The performance of our method is shown in Table 4. We observe that the memory cost is lower than that of the CHT-revoting algorithm since our method does not record feature points. Notably, the proposed method is much faster than the conventional HT method. For the proposed method, $\textbf{\textit{Time}}_{\textbf{\textit{vote}}}$ is proportional to the number of black pixels in the image, $\textbf{\textit{Time}}_{\textbf{\textit{sort}}}$ to the number of line segments, and $\textbf{\textit{Time}}_{\textbf{\textit{verify}}}$ to the total length of line segments. The time efficiency of the image-based line verification is confirmed by analyzing the distribution of processing time. For the medium-size images A4, A3 and A2, $\textbf{\textit{Time}}_{\textbf{\textit{verify}}}$ is similar to $\textbf{\textit{Time}}_{\textbf{\textit{vote}}}$. Even for the largest image A0, $\textbf{\textit{Time}}_{\textbf{\textit{verify}}}$ is only three times of $\textbf{\textit{Time}}_{\textbf{\textit{vote}}}$. Compared with the above two CHT algorithms, the ratio of $\textbf{\textit{Time}}_{\textbf{\textit{verify}}}$ to $\textbf{\textit{Time}}_{\textbf{\textit{vote}}}$ is greatly reduced.

The proposed method also achieves stable detection accuracy above 82%. Compared with the detection accuracy of the CHT-revoting algorithm shown in Table 3, the proposed method has higher detection rates due to better endpoint localization and thickness precision. The proposed method also enjoys lower false rates due to less repetitive detections. Fig. 9 shows the recognition result of the largest image A0. As indicated in the figure, the false detections mostly come from two sources: (1) aligned texts in the drawing, and (2) long folds on the paper. These two types of false detections are hard to eliminate in the graphics level, while they should be removed by semantic analysis, such as layout analysis. Fig. 10 shows a fraction of the test image where the line quality is degraded (Fig. 10a) but the recognition result is satisfactory (Fig. 10b). The recognized lines are displayed with their detected line thickness. We can see that both the localization and the line thickness are detected correctly. The vertical broken lines are also recovered, while the horizontal dashed lines are retained.

Fig. 11 compares the performance curves over various image sizes between our method and the CHT-revoting algorithm, demonstrating the significant performance gain by utilizing the image space. Compared with the CHT-revoting

Table 4
Performance of our proposed method

		A4	A3	A2	A1	A0
Memory cost (MB)		6.8	8.3	13.2	17.1	26.9
Time (s)	$Time_{all}$	1.96	6.53	12.17	25.43	59.28
	$Time_{vote}$	0.86	3.47	4.28	6.39	9.63
	$Time_{sort}$	0.11	0.27	1.58	5.28	18.63
	$Time_{verify}$	0.80	2.36	5.38	12.88	30.03
Detection rate (%)		78.9	77.6	78.1	77.5	79.1
False rate (%)		2.8	3.0	12.5	11.8	9.9
Detection accuracy (%)		88.1	87.3	82.8	82.9	84.6

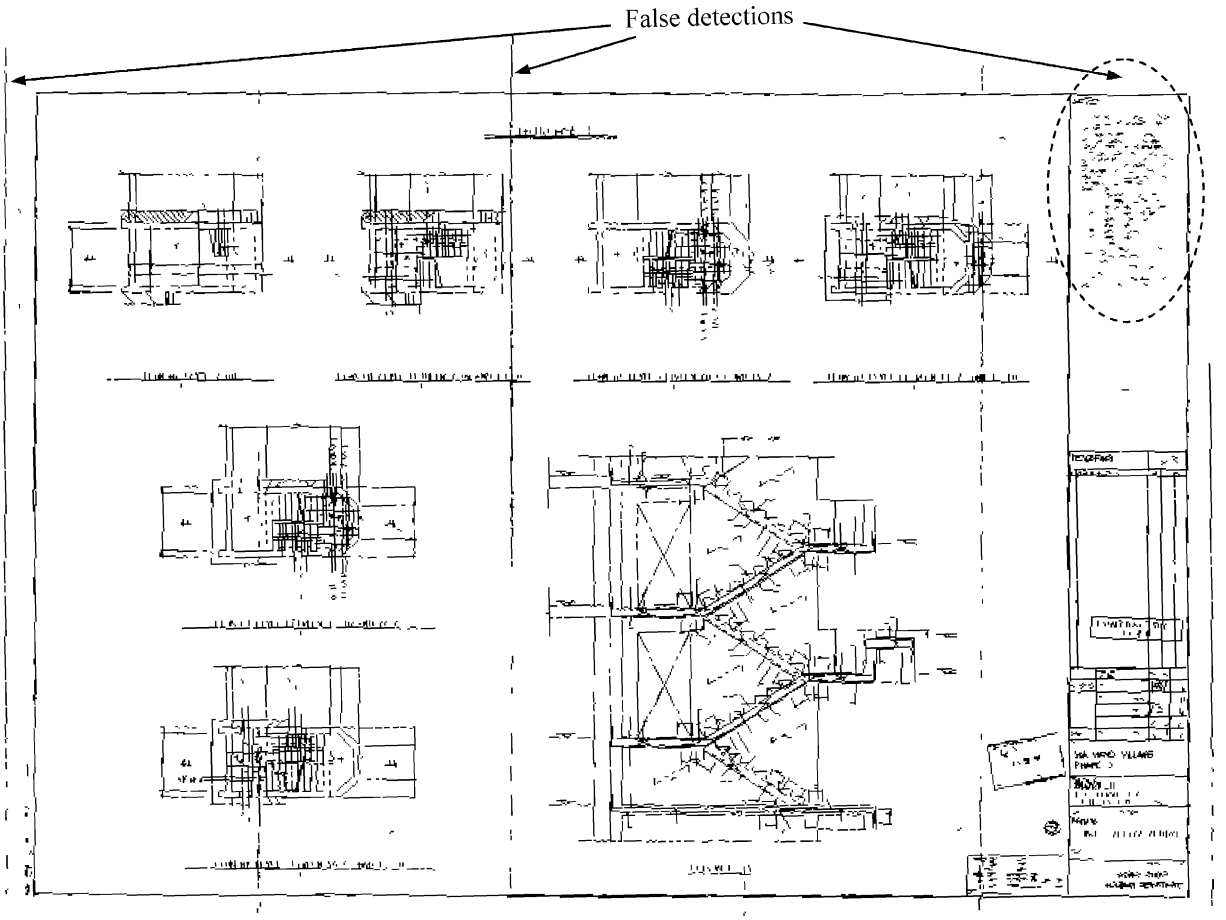

Fig. 9. Recognition result of the image A0 by our proposed method.

algorithm, the proposed method saves averagely about 25% memory space and 94% processing time.

Furthermore, to evaluate the effect of gradient prediction, we measure the performance of the proposed method without gradient prediction. The experimental results are shown in Table 5. The time comparison curves between with and without gradient prediction are plotted in Fig. 12. We observe that the $\boldsymbol{Time_{all}}$ of recognition without gradient prediction increases over that of with gradient prediction by a nearly constant value (Fig. 12a). However, when looking into the three parts of $\boldsymbol{Time_{all}}$, we find interesting facts. For $\boldsymbol{Time_{vote}}$, as expected, voting with gradient prediction speeds up the process more when the image size becomes larger (Fig. 12b). For $\boldsymbol{Time_{sort}}$, surprisingly, using

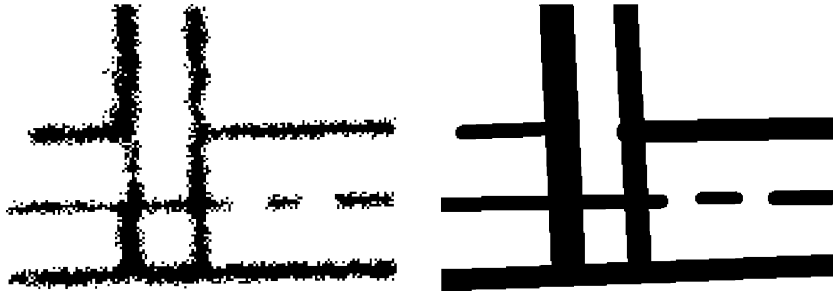

Fig. 10. Line recognition result of a fraction of test image.

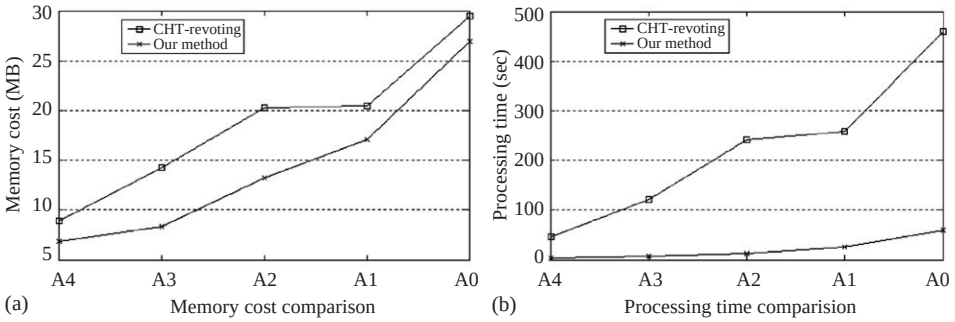

Fig. 11. Performance comparison between our method and the CHT-revoting algorithm.

Table 5
Performance of the proposed method without gradient prediction

		A4	A3	A2	A1	A0
Time (s)	$Time_{all}$	3.72	9.83	17.95	33.12	67.5
	$Time_{vote}$	2.27	6.19	8.67	10.16	15
	$Time_{sort}$	0.09	0.20	0.73	3.66	8.66
	$Time_{verify}$	1.16	3.14	7.63	18.70	42.94
Detection rate (%)		79.1	78.9	77.0	79.0	78.6
False rate (%)		3.6	3.9	13.4	13.5	11.4
Detection accuracy (%)		87.8	87.5	81.8	82.8	83.6

gradient prediction slows down the process, especially for the largest image A0 (Fig. 12c). The reason is that when voting without gradient prediction, the HT parameter space is more "smooth" than voting with gradient prediction because the randomly aligned feature points on a direction also accumulate to a high value. Therefore, the number of detectable peaks (local maxima) decreases. On the other hand, the boundary of a direction may be mistakenly enlarged by randomly aligned feature points, which explains why $\boldsymbol{Time_{verify}}$ of recognition without gradient prediction is much slower for large images (Fig. 12d). Moreover, the false rate of recognition without gradient prediction also increases. In conclusion, the gradient prediction not only accelerates the HT voting process, but also eliminates random aligned noises.

Finally, to evaluate the effect of boundary recorder, we measure the performance of the proposed method without boundary recorder. The experimental results are shown in Table 6. We find that $\boldsymbol{Time_{all}}$ is greatly reduced by using boundary recorder. The magnitude of reduction depends on the image size in a direct ratio (Fig. 13a). The detailed analysis of the three parts of $\boldsymbol{Time_{all}}$ reveals that almost all the acceleration comes from $\boldsymbol{Time_{verify}}$ (Fig. 13b). For the medium-size images A4 and A3, the acceleration is not obvious; however, for the large-size images A2, A1 and A0, $\boldsymbol{Time_{verify}}$'s without boundary recorder are 2.1, 1.8, and 1.7 times of those with boundary recorder, respectively. On the other hand, the detection accuracy is not affected by whether using boundary recorder or not. Therefore, we conclude that the effect of boundary recorder is mainly on

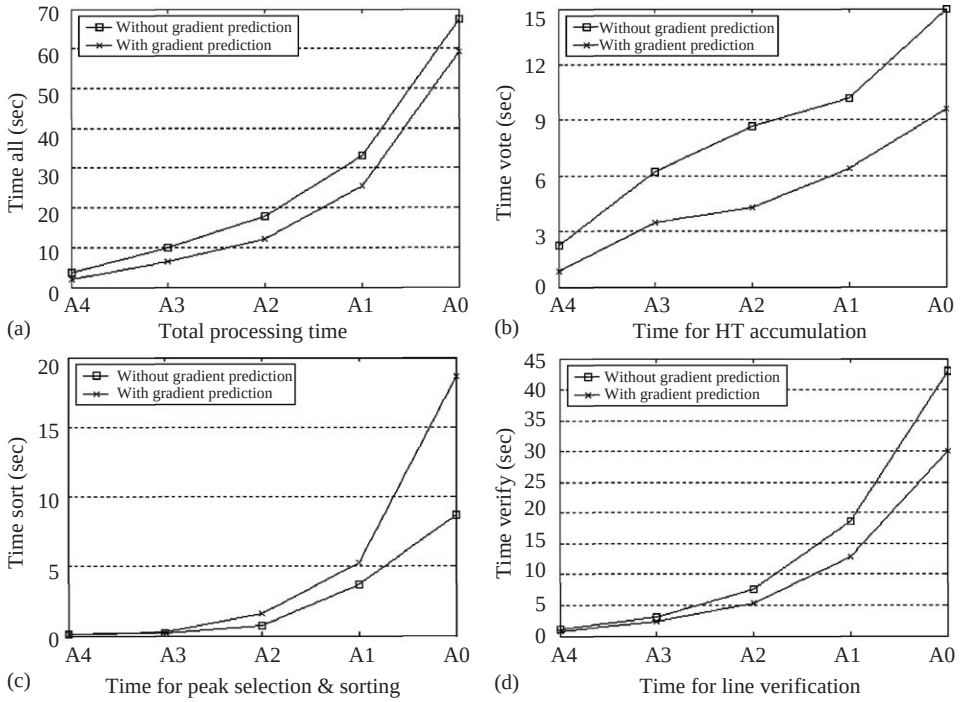

Fig. 12. Performance comparison between with and without gradient prediction.

Table 6
Performance of the proposed method without boundary recorder

		A4	A3	A2	A1	A0
Time (s)	$Time_{all}$	2.88	7.97	17.65	36.16	82.70
	$Time_{vote}$	0.86	3.13	4.08	6.03	9.00
	$Time_{sort}$	0.52	0.55	2.17	6.38	21.20
	$Time_{verify}$	1.38	3.89	11.09	23.52	51.11
Detection rate (%)		78.4	78.1	79.0	77.9	78.3
False rate (%)		3.1	4.0	13.1	13.6	11.7
Detection accuracy (%)		87.7	87.1	83.0	82.2	83.3

speeding up the line verification, especially for large-size images.

5. Conclusions

This paper proposes a new method for recognizing straight-line segments based on HT, aiming at overcoming the long-existing limitations of HT-based methods, including the weakness on handling large-size images and the unawareness of line thickness. The proposed method makes two major contributions. (1) It proposes to utilize the image space throughout the whole recognition process. Several novel image-based techniques are introduced to make the process more efficient. The gradient prediction accelerates the HT accumulation and helps eliminate the random aligned noises. The boundary recorder greatly removes the redundancy of line verification on large-size images. Erasing the pixels belonging to newly-recognized lines avoids overlapping lines effectively. All these techniques work together to significantly speed up the whole recognition process for large-size images, while maintaining high detection accuracy, as confirmed by the experimental results. (2) The image-analysis-based line verification enables the proposed method to detect line thickness correctly, which is critical to many applications. Therefore, the success of the proposed method demonstrates that HT-based methods can also be applied to large-size images,

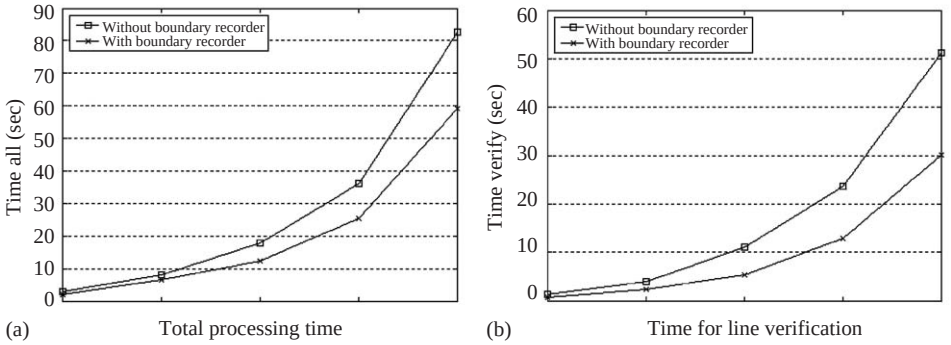

Fig. 13. Performance comparison between with and without boundary recorder.

e.g., engineering drawings, if the image space is properly utilized.

6. Summary

Graphical element recognition plays an important role in various image-understanding applications. Hough Transform is recognized as a powerful tool for graphic element extraction from images due to its global vision and robustness in noisy or degraded environment. However, the application of Hough Transform has been limited to small-size images for a long time. One limitation is the time inefficiency. Besides the well-known heavy computation in the accumulation step, the peak detection step and the line verification step also become much more time-consuming for large-size images. Another limitation is that most existing Hough Transform-based line recognition methods are not able to detect line thickness, which is essential to large-size images, usually engineering drawings. We believe these limitations arise from that these methods only work on the Hough Transform parameter space. This paper therefore proposes a new Hough Transform-based line recognition method, which utilizes both the Hough Transform parameter space and the image space. The proposed method devises an image-based gradient prediction to accelerate the accumulation, introduces a boundary recorder to eliminate redundant analyses in the line verification, and develops an image-based line verification algorithm to detect line thickness and reduce false detections as well. It also proposes to use pixel-level line erasing to avoid overlapping lines instead of rigidly suppressing the $N \times N$ neighborhood. We perform experiments on real images with different sizes in terms of speed and detection accuracy. The experimental results demonstrate the significant performance improvement, especially for large-size images. The proposed method makes two major contributions. First, it proposes to utilize the image space throughout the whole recognition process. Several novel image-based techniques are introduced to make the process more efficient. The gradient prediction accelerates the HT accumulation and helps eliminate the random aligned noises. The boundary recorder greatly removes the redundancy of line verification on large-size images. Erasing the pixels belonging to newly recognized lines avoids overlapping lines effectively. All these techniques work together to significantly speed up the whole recognition process for large-size images, while maintaining high detection accuracy, as confirmed by the experimental results. Second, the image-analysis-based line verification enables the proposed method to detect line thickness correctly, which is critical to many applications. Therefore, the success of the proposed method demonstrates that Hough Transform-based methods can also be applied to large-size images, e.g., engineering drawings, if the image space is properly utilized.

Acknowledgements

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4182/03E).

References

[1] J. Illingworth, J. Kittler, A survey of the Hough transform, Comput. Vision Graphics Image Process. 44 (1988) 87–116.
[2] R.O. Duda, P.E. Hart, Use of Hough transformation to detect lines and curves in pictures, Commun. ACM 15 (1) (1972) 11–15.
[3] F. O'Gorman, M.B. Clowes, Finding picture edges through collinearity of feature points, IEEE Trans. Comput. 25 (4) (1976) 449–456.
[4] T.M. van Veen, F.C.A. Groen, Discretization errors in the Hough transform, Pattern Recognition 14 (1981) 137–145.
[5] P.R. Thrift, S.M. Dunn, Approximating point-set images by line segments using a variation of the Hough transform, Comput. Vision Graphics Image Process. 21 (1981) 383–394.

[6] T. Risse, Hough transform for line recognition: complexity of evidence accumulation and cluster detection, Comput. Vision Graphics Image Process. 46 (3) (1989) 327–345.
[7] H.F. Li, D. Pao, R. Jayakumar, Improvements and systolic implementation of the Hough transformation for straight line detection, Pattern Recognition 22 (1989) 697–706.
[8] J. Princen, J. Illingworth, J. Kittler, A hierarchical approach to line extraction based on the Hough transform, Comput. Vision Graphics Image Process. 52 (1990) 57–77.
[9] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic Hough transform, Pattern Recognition 24 (4) (1991) 303–316.
[10] P.-K. Ser, W.-C. Siu, Sampling Hough algorithm for the detection of lines and curves, Proceedings of IEEE International Symposium on Circuits and Systems, vol. 5, 1992, pp. 2497–2500.
[11] L. Xu, E. Oja, Randomized Hough transform (RHT): basic mechanisms, algorithms, and computational complexities, CVGIP: Image Understand. 57 (2) (1993) 131–154.
[12] D. Shaked, O. Yaron, N. Kiryati, Deriving stopping rules for the probabilistic Hough transform by sequential, Comp. Vision Image Understand. 63 (3) (1996) 512–526.
[13] M.C.K. Yang, J.-S. Lee, C.-C. Lien, C.-L. Huang, Hough transform modified by line connectivity and line thickness, IEEE Trans. Pattern Anal. Mach. Intell. 19 (8) (1997) 905–910.
[14] V. Kamat, S. Ganesan, A robust Hough transform technique for description of multiple line segments in an image, in: Proceedings of International Conference on Image Processing (ICIP'98), 1998, pp. 216–220.
[15] K. Murakami, T. Naruse, High speed line detection by Hough transform in local area, in: Proceedings of International Conference on Pattern Recognition (ICPR'00), 2000, pp. 467–470.
[16] Y.-T. Ching, Detecting line segments in an image—a new implementation for Hough Transform, Pattern Recogn. Lett. 22 (2001) 421–429.
[17] C.W. Niblack, P.B. Gibbons, D.W. Capson, Generating skeletons and centerlines from the distance transform, CVGIP: Graph. Models Image Process. 54 (5) (1992) 420–437.
[18] R.D.T. Janssen, A.M. Vossepoel, Adaptive vectorization of line drawing images, Comput. Vision Image Understand. 65 (1) (1997) 38–56.
[19] C.-C. Han, K.-C. Fan, Skeleton generation of engineering drawings via contour matching, Pattern Recognition 27 (2) (1994) 261–275.
[20] D. Dori, W. Liu, Sparse pixel vectorization: an algorithm and its performance evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 21 (3) (1999) 202–215.
[21] J. Song, F. Su, C.-L. Tai, J. Chen, S. Cai, An object-oriented progressive-simplification based vectorization system for engineering drawings: model, algorithm and performance, IEEE Trans. Pattern Anal. Machine Intell. 24 (8) (2002) 1048–1060.
[22] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, Computer Graphics: Principles and Practice, Addison-Wesley, Reading, MA, 1990.
[23] Available at http://www.cse.cuhk.edu.hk/~jqsong/testing_images.zip.

About the Author—JIQIANG SONG received the B.S. degree in Computer Science and the Ph.D. degree in Computer Science and Application from Nanjing University in 1996 and 2001, respectively. Currently, he is a Postdoctoral Fellow at Department of Computer Science and Engineering, the Chinese University of Hong Kong. His research interests include graphics recognition, automatic interpretation of engineering drawings, image/video processing and video compression. He has published over 30 papers in these areas. He is a member of IEEE.

About the Author—MICHAEL R. LYU received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1981, the M.S. degree in computer engineering from University of California, Santa Barbara, in 1985, and the Ph.D. degree in computer science from University of California, Los Angeles, in 1988. He is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He was with the Jet Propulsion Laboratory as a Technical Staff Member from 1988 to 1990. From 1990 to 1992, he was with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, as an Assistant Professor. From 1992 to 1995, he was a Member of the Technical Staff in the applied research area of Bell Communications Research (Bellcore), Morristown, New Jersey. From 1995 to 1997, he was a Research Member of the Technical Staff at Bell Laboratories, Murray Hill, New Jersey. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, Web technologies, multimedia information processing, and E-commerce systems. He has published over 180 refereed journal and conference papers in these areas. He received Best Paper Awards in ISSRE'98 and ISSRE'2003. He has participated in more than 30 industrial projects, and helped to develop many commercial systems and software tools. He was the editor of two book volumes: Software Fault Tolerance (New York: Wiley, 1995) and The Handbook of Software Reliability Engineering (Piscataway, NJ: IEEE and New York: McGraw-Hill, 1996). Dr. Lyu initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the program chair for ISSRE'96, and has served in program committees for many conferences, including ISSRE, SRDS, HASE, ICECCS, ISIT, FTCS, DSN, ICDSN, EUROMICRO, APSEC, PRDC, PSAM, ICCCN, ISESE, and WWW. He was the General Chair for ISSRE2001, and the WWW10 Program Co-Chair. He has been frequently invited as a keynote or tutorial speaker to conferences and workshops in U.S., Europe, and Asia. He served on the Editorial Board of IEEE Transactions on Knowledge and Data Engineering, and has been an Associate Editor of IEEE Transactions on Reliability and Journal of Information Science and Engineering. Dr. Lyu is a fellow of IEEE.