A New Line Segment Grouping Method for Finding Globally Optimal Line Segments

Jeong-Hun Jang and Ki-Sang Hong

Image Information Processing Lab, Dept. of E.E., POSTECH, Korea
{jeonghun,hongks}@postech.ac.kr

Abstract. In this paper we propose a new method for extracting line segments from edge images. Our method basically follows a line segment grouping approach. This approach has many advantages over a Hough transform based approach in a practical situation. However, since its process is purely local, it does not provide a mechanism for finding more favorable line segments from a global point of view. Our method overcomes the local nature of the conventional line segment grouping approach, while retaining most of its advantages, by incorporating some useful concepts of the Hough transform based approach into the line segment grouping approach. We performed a series of tests to compare the performance of our method with those of other six existing methods. Throughout the tests our method ranked almost highest both in detection rate and computation time.

1 Introduction

A line segment is a primitive geometric object that is frequently used as an input to higher-level processes such as stereo matching, target tracking, or object recognition. Due to its importance, many researchers have devoted themselves to developing good line segment detection methods. The methods proposed up to date can be categorized into following three approaches:

Approach 1: Hough transform based approach [1,2,3]. The biggest advantage of this approach is to enable us to detect collinear edge pixels even though each of them is isolated. Therefore, this approach is useful in finding lines in noisy images where local information around each edge pixel is unreliable or unavailable. Another advantage of this approach is that it has a global nature since the score assigned to each detected line is computed by considering all the edge pixels lying on that line. However, there are several problems in this approach. It requires relatively a large amount of memory and long computation time, and raises the so-called *connectivity problem*, where illusionary lines that are composed of accidentally collinear edge pixels are also detected. To find lines with high accuracy, the approach needs fine quantization of parameter space, but the fine quantization makes it difficult to detect edge pixels that are not exactly collinear. The original Hough transform gives us only information on existence of lines, but not line segments. Therefore, we have to group pixels along each

S.-W. Lee, H.H. Bülthoff, T. Poggio (Eds.): BMCV 2000, LNCS 1811, pp. 397–406, 2000.

detected line, but this process is local, i.e., it does not guarantee the detection of globally optimal line segments.

Approach 2: Line segment grouping approach [4,5,6]. Elementary line segments (ELSs) are obtained by linking edge pixels and approximating them to piecewise straight line segments. These ELSs are used as an input to this approach. Adjacent line segments are grouped according to some grouping criteria and replaced by a new line segment. This process is repeated until no new line segment occurs. This approach overcomes many weaknesses of Approach 1. However, when most of the edge pixels are isolated or the ELSs are perturbed severely by noises so that information on them is almost useless, the approach does not work. Another disadvantage of this approach is that its process is purely local. Repetition of locally optimal grouping of line segments does not guarantee their globally optimal grouping. For example, consider an optimal grouping criterion in which an ELS needs to be grouped only into the longest line segment among detectable ones that can share the ELS. Such a criterion cannot be handled properly with this approach.

Basically, our method belongs to Approach 2. ELSs are used as an input to our method. However, the grouping mechanism of our method is different from that of the conventional approach. While retaining the advantages of Approach 2, in order to overcome its local nature, we adopted the concept of "line detection by voting" from Approach 1. By combining the concepts of both approaches, a more powerful method could be constructed.

Our method consists largely of three steps. In the first step, ELSs are grouped iteratively by measuring the orientation difference and proximity between an ELS and a line formed by ELSs grouped already. This step yields *base lines*, and for each base line, information on the ELSs contributing to the construction of the base line is also given. It should be noted that an ELS is allowed to belong to several base lines simultaneously. Therefore, a base line does not represent a real line, but represents a candidate line around which true line segments possibly exist. In the second step, ELSs around each base line are grouped into line segments along the base line with some grouping criteria. We designed the algorithm of this step so that closely positioned parallel line segments are effectively separated. In the third step, redundant line segments are removed by applying a restriction that an ELS should be grouped only into the longest line segment among ones sharing the ELS. The restriction can be relaxed so that an ELS is allowed to be shared by several, locally dominant line segments. Note that the criterion we use here for optimal grouping is the *length* of a line segment. Other criteria (e.g., the fitness of a line segment) can be applied easily by making a few modifications to the proposed algorithm. After the third step has been completed, if there exist ELSs remaining ungrouped, the above three steps are repeated, where the remaining ELSs are used as an input to the next iteration.

This paper is organized as follows. In Section 2, the proposed algorithms for the above three steps are described in detail. Section 3 shows experimental results on a performance comparison between our method and other conventional ones. Conclusions are given in Section 4.

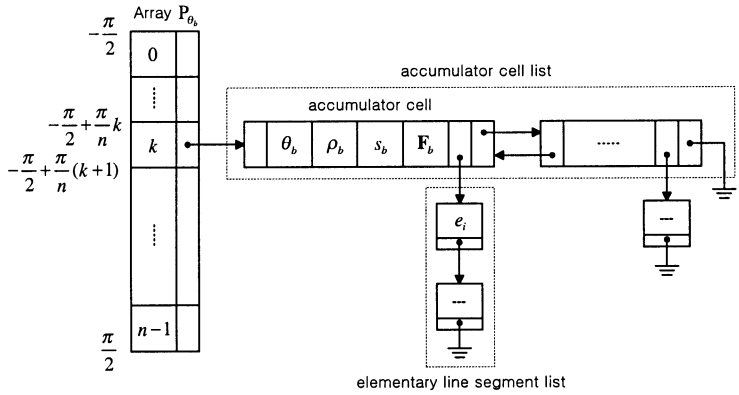

Fig. 1. Basic structure of an accumulator array.

2 The Proposed Method

An input to our method is a set of *elementary line segments*. If an edge image is given, the ELSs are obtained by first linking edge pixels using the *eight-directional following* algorithm, and then approximating them to piecewise straight segments using the *iterative endpoint fit-and-split algorithm* [7].

2.1 Finding Base Lines

Base lines are candidate lines near which true line segments may exist. They are obtained by grouping collinear ELSs. Figure 1 shows a data structure, called an *accumulator array*, which is used to manage base lines. The contents of the accumulator array are updated as the grouping goes on. The accumulator array is composed of *accumulator cells*, each of which contains four parameters (θ_b, ρ_b, s_b, and $\mathbf{F}_b$) concerning the corresponding base line, and an *elementary line segment list* whose elements are ELSs that contribute to the construction of the base line. The parameters θ_b and ρ_b represent two polar parameters of a base line that is given by

$$x\cos\theta_b + y\sin\theta_b = \rho_b, \quad -\frac{\pi}{2} \le \theta_b < \frac{\pi}{2} \tag{1}$$

and the parameter s_b represents a voting score, which is computed by summing all the lengths of the ELSs registered in the ELS list of an accumulator cell. The 3×3 matrix $\mathbf{F}_b$ is used to compute a weighted least-squares line fit to the ELSs in an ELS list. In Figure 1, the array P_{θ_b} is an array of n pointers, where the kth pointer indicates the first accumulator cell of the *accumulator cell list* whose accumulator cells have the values of θ_b in the range of $-\pi/2 + k\pi/n \le \theta_b < -\pi/2 + (k+1)\pi/n$. The array P_{θ_b} is used to find fast the accumulator cells whose values of θ_b are in a required range.

The detailed algorithm for finding base lines is given as follows. Initially, all the pointers of P_{θ_b} are set to null and the accumulator array has no accumulator

cells. Let e_i denote the ith ELS ($i = 1, \ldots, N_e$) and let $\theta(e_i)$ and $\rho(e_i)$ represent two polar parameters of a line passing through two end points of e_i. For each e_i, accumulator cells satisfying the condition

$$|\theta_b - \theta(e_i)| \leq \Delta\theta_b + \frac{\Delta\theta_d}{l(e_i)} \tag{2}$$

are searched for, where $\Delta\theta_b$ and $\Delta\theta_d$ are user-specified parameters and $l(e_i)$ denotes the length of e_i. If no such accumulator cells exist, a new accumulator cell is created and the parameters of the cell are set as follows:

$$\theta_b \leftarrow \theta(e_i), \quad \rho_b \leftarrow \rho(e_i), \quad s_b \leftarrow l(e_i), \quad \text{and} \quad \mathbf{F}_b \leftarrow \frac{l(e_i)}{2}(\mathbf{f}_1^T\mathbf{f}_1 + \mathbf{f}_2^T\mathbf{f}_2)$$

where

$$\mathbf{f}_1 = [x_1(e_i), y_1(e_i), -1], \quad \mathbf{f}_2 = [x_2(e_i), y_2(e_i), -1],$$

and $(x_1(e_i), y_1(e_i))$ and $(x_2(e_i), y_2(e_i))$ represent two end points of e_i. The created cell is registered in an appropriate accumulator cell list according to the value of θ_b. If accumulator cells satisfying Equation (2) exist, a proximity measure $p(e_i, b)$ between e_i and a base line b of each of the accumulator cells is computed, where $p(e_i, b)$ is defined by

$$p(e_i, b) = \sqrt{\frac{1}{l(e_i)} \int_0^1 (x(t)\cos\theta_b + y(t)\sin\theta_b - \rho_b)^2 dt} \tag{3}$$

where

$$x(t) = x_1(e_i) + t(x_2(e_i) - x_1(e_i)) \quad \text{and} \quad y(t) = y_1(e_i) + t(y_2(e_i) - y_1(e_i)).$$

If $p(e_i, b)$ is less than a threshold T_p and e_i is not registered in the ELS list of the accumulator cell corresponding to the base line b, e_i is registered in the ELS list and the parameters of the accumulator cell are updated. First, $\mathbf{F}_b$ is updated by $\mathbf{F}_b \leftarrow \mathbf{F}_b + \frac{l(e_i)}{2}(\mathbf{f}_1^T\mathbf{f}_1 + \mathbf{f}_2^T\mathbf{f}_2)$ and then the normalized eigenvector corresponding to the smallest eigenvalue of $\mathbf{F}_b$ is computed. The elements of the eigenvector represent the parameters (a, b, and c) of a line ($ax + by = c$) that is obtained by weighted least-squares line fitting of the end points of the ELSs in the ELS list. New values of θ_b and ρ_b are calculated from the eigenvector. Finally, s_b is updated by $s_b \leftarrow s_b + l(e_i)$. Note that if the updated value of θ_b of an accumulator cell is outside the range of θ_b of the accumulator cell list the accumulator cell currently belongs to, the accumulator cell should be moved to a new appropriate accumulator cell list so that the value of θ_b of the accumulator cell is always kept inside the range of θ_b of a current accumulator cell list.

If the accumulator array has been updated for all e_i ($i = 1, \ldots, N_e$), the first iteration of our base line finding algorithm is now completed. The above procedure is repeated until no accumulator cell is updated. It should be noted that the creation of a new accumulator cell is allowed only during the first iteration of the algorithm. The repetition of the above procedure is necessary because for some

e_i and b, the orientation difference $|\theta_b - \theta(e_i)|$ and the proximity measure $p(e_i, b)$ may become less than given threshold values as b is continuously updated. The repetition has another important role. It allows an ELS to be shared by several base lines simultaneously. This property, with the two algorithms that will be explained in the next two subsections, makes it possible for our method to find more favorable line segments from a global point of view.

2.2 Grouping ELSs along Each Base Line

Once base lines have been obtained, we have to group ELSs into actual line segments along each base line. According to the algorithm of Subsection 2.1, each base line carries a list of ELSs that participate in the construction of that line. Therefore, when grouping ELSs along a base line, we take into account only ones registered in the ELS list of the line.

Let b_c represent a line segment that is obtained by clipping a base line b with an image boundary rectangle. If a point on b_c is represented by a single parameter s $(0 \leq s \leq l(b_c))$, the value of s of a point obtained by projecting a point (x, y) onto b_c is given by

$$s(x, y) = \frac{x_2(b_c) - x_1(b_c)}{l(b_c)}(x - x_1(b_c)) + \frac{y_2(b_c) - y_1(b_c)}{l(b_c)}(y - y_1(b_c)). \quad (4)$$

Let e_i^b denote the ith ELS $(i = 1, \ldots, N_b)$ registered in the ELS list of the base line b and let $s_1(e_i^b)$ and $s_2(e_i^b)$ $(s_1 < s_2)$ represent two values of s that are obtained by substituting two end points of e_i^b into Equation (4). Then our algorithm for grouping ELSs along the base line b is given as follows.

Procedure:

1. Create a 1D integer array I of length $N_\mathrm{I} + 1$, where $N_\mathrm{I} = \lceil l(b_c) \rceil$. Each element of I is initialized to 0.
2. Compute $p(e_i^b, b)$ for $i = 1, \ldots, N_b$.
3. Compute $s_1(e_i^b)$ and $s_2(e_i^b)$ for $i = 1, \ldots, N_b$.
4. Sort p's computed in Step 2 in a decreasing order of p. Let p_j denote the jth largest value of p $(j = 1, \ldots, N_b)$ and let $e^b(p_j)$ represent an ELS corresponding to p_j.
5. Do $\{\mathrm{I}[k] \leftarrow$ index number of $e^b(p_j)\}$ for $j = 1, \ldots, N_b$ and $k = \lfloor s_1(e^b(p_j)) + 0.5 \rfloor, \ldots, \lfloor s_2(e^b(p_j)) + 0.5 \rfloor$.
6. Create a 1D integer array S of length $N_b + 1$. Each element of S is initialized to 0.
7. Do $\{\mathrm{S}[\mathrm{I}[k]] \leftarrow \mathrm{S}[\mathrm{I}[k]] + 1\}$ for $k = 0, \ldots, N_\mathrm{I}$.
8. Remove e_i^b from the ELS list if $\mathrm{S}[i]/l(e_i^b) < T_m$, where T_m is a user-specified parameter $(0 \leq T_m \leq 1)$. By doing so, we can separate closely positioned parallel line segments effectively.
9. Clear the array I and perform Step 4 once again.
10. Group $e_{\mathrm{I}[i]}^b$ and $e_{\mathrm{I}[j]}^b$ into the same line segment for $i, j = 0, \ldots, N_\mathrm{I}$ $(\mathrm{I}[i] \neq 0$ and $\mathrm{I}[j] \neq 0)$ if $|i - j| < T_g$, where T_g is a user-specified gap threshold $(T_g > 0)$.

2.3 Removing Redundant Line Segments

According to the algorithm of Subsection 2.1, an ELS is allowed to be shared by several base lines. This property of the algorithm causes an ELS to be shared by several line segments after the grouping of Subsection 2.2. This property is very useful as will be shown in Section 3, but it may produce redundant line segments. For example, if most of the ELSs that belong to some line segment g are shared by more dominant line segments (i.e., longer line segments), the line segment g is redundant and should be removed.

Assume that a total of N_g line segments have been found with the grouping algorithm of Subsection 2.2 and let g_i denote the ith line segment ($i = 1, \ldots, N_g$). Here is our algorithm for redundant line segment removal.

Notations:

- $S(g)$: sum of the lengths of ELSs belonging to a line segment g.
- $L(e)$: label assigned to an ELS e.
- $G(e)$: a set of line segments sharing e as a common member ELS.
- $E(g)$: a set of ELSs belonging to a line segment g.

Procedure:

1. Do $\{L(e_i) \leftarrow$ index number of $g(e_i)\}$ for $i = 1, \ldots, N_e$, where $g(e_i) = \text{argmax}_{g \in G(e_i)} S(g)$.
2. Create a 1D array M of length $N_g + 1$. Each element of M is initialized to 0.
3. Do $\{\text{M}[L(e)] \leftarrow \text{M}[L(e)] + l(e)\}$ for all $e \in E(g_i)$ and $i = 1, \ldots, N_g$.
4. Discard g_i if $\text{M}[i]/S(g_i) \leq T_r$, where T_r is a user-specified parameter ($0 \leq T_r \leq 1$).

The procedures of Subsections 2.1, 2.2, and 2.3 are repeated if there exist ELSs remaining ungrouped. The remaining ELSs are used as an input to the next iteration. This iteration is necessary because ELSs that are not grouped into any line segment may appear during the processes of Subsections 2.2 and 2.3.

3 Experimental Results

In this section, experimental results on a comparison between our method and other line segment detection methods are shown. The conventional methods we chose for the purpose of comparison are as follows: Boldt *et al.*'s [4], Liang's [8], Nacken's [5], Princen *et al.*'s [10], Risse's [9], and Yuen *et al.*'s [11] methods, where Boldt *et al.*'s and Nacken's methods are based on the line segment grouping approach and the others are related with the Hough transform based approach. Note that originally, Liang's and Risse's methods are not designed for detecting line segments, but for lines. To make use of their methods for line segment detection, we appended a line segment extraction routine, where two adjacent edge pixels on each detected line are grouped into the same line segment if the distance between them is smaller than a given gap threshold.

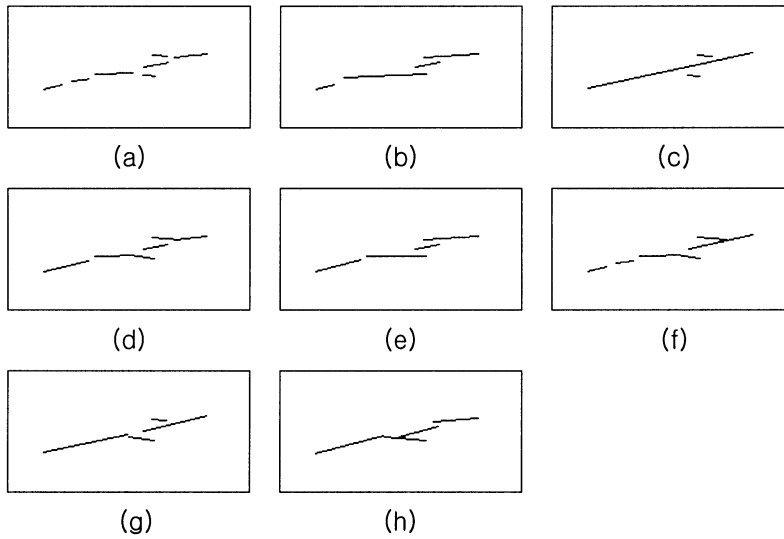

Fig. 2. Line segment detection results of conventional methods and ours for the simple test image: (a) input image, (b) Boldt *et al.*'s method, (c) our method ($T_r = 0.5$ and $\Delta\theta_b = 7.0°$), (d) Liang's method, (e) Nacken's method, (f) Princen *et al.*'s method, (g) Risse's method, and (h) Yuen *et al.*'s method.

To see the characteristics of the methods mentioned above, we made a simple test image as shown in Figure 2(a), where a sequence of ELSs, which looks like a single long line segment that is broken into several fragments, is given from upper right to lower left, and two short line segments are added. Figures 2(b)-(h) show the results yielded by the line segment detection methods. Note that all the methods failed to detect the long line segment except ours. The result of Figure 2(c) illustrates the global nature of our method well. In Boldt *et al.*'s and Nacken's methods, as can be seen in Figures 2(b) and (e), initial wrong grouping of ELSs (though it might be locally optimal) prohibited the methods from proceeding to group the ELSs further into the long segment. Such a behavior is a fundamental limitation of the conventional line segment grouping methods. In Figure 2(a), edge pixels constituting the long segment are not exactly collinear, which is a main reason for the failure of the Hough transform based methods. In such a case, locally collinear pixels may dominate the detection process of a Hough transform based approach. To solve the problem, coarse quantization of parameter space may be needed, but it causes another problem of the inaccurate detection of lines.

Figure 3 shows other detection results of our method for the same test image of Figure 2(a), obtained by adjusting some threshold values. In Figure 3(a), one can see that some ELSs are shared by more than one line segment. Allowing an ELS to be shared by more than one line segment is necessary when two or more real line segments are closely positioned so that some parts of them become

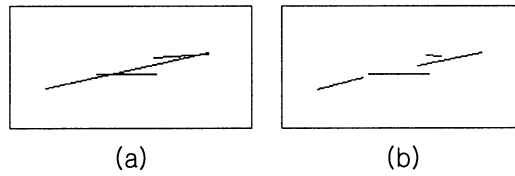

Fig. 3. Other detection results of our method for the same test image of Figure 2(a): (a) $T_r = 0.2$ and (b) $\Delta\theta_b = 3.0°$.

shared after a digitization process. Our method has a capability of allowing such kind of grouping and the degree of sharing is determined by the threshold T_r. The result of Figure 3(b) was obtained by decreasing the value of $\Delta\theta_b$, which has an effect of emphasizing collinearity of ELSs when grouping them.

We carried out four experiments to see the performance of each method. The size of test images used in the experiments is 300×300 and each image contains 18 true line segments that are generated randomly with a length of 40 to 200 pixels. In each experiment, a total of 50 images are generated and tested. In Experiment 1, each test image contains 18 line segments without any fragmentation of them and any noise added (see Figure 4(a)). In Experiment 2, each true line segment is broken into short ones, where the length of each short line segment is 6 to 20 pixels and the gap width is 2 to 6 pixels (see Figure 4(b)). In Experiment 3, a

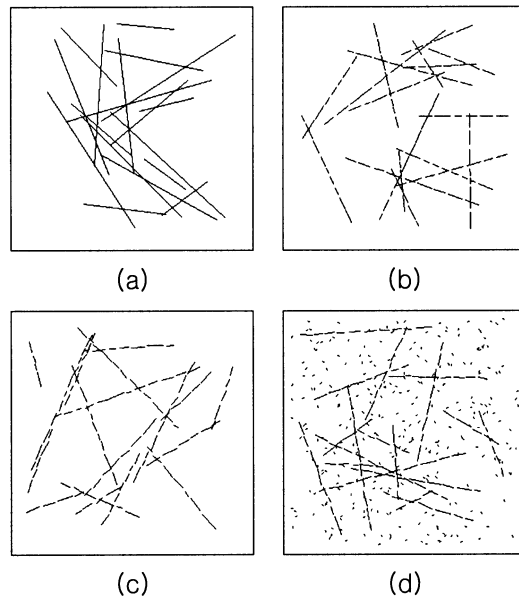

Fig. 4. Sample test images used in the experiments: (a) Experiment 1, (b) Experiment 2, (c) Experiment 3, and (d) Experiment 4.

Table 1. The result of Experiment 1.

match score	Boldt	Ours	Liang	Nacken	Princen	Risse	Yuen
0.0 – 0.9	1.9	1.9	7.1	1.4	3.0	2.4	2.4
0.9 – 1.0	16.5	16.4	13.8	16.3	16.3	16.5	16.0
0.98 – 1.0	14.9	15.2	11.7	14.0	14.0	14.6	13.0
# of line segments detected	18.4	18.3	20.9	17.7	19.3	19.0	18.4
execution time (msec)	153	178	784	1467	4299	6029	737

Table 2. The result of Experiment 2.

match score	Boldt	Ours	Liang	Nacken	Princen	Risse	Yuen
0.0 – 0.9	2.6	1.5	9.4	2.0	3.9	3.7	3.5
0.9 – 1.0	16.0	16.6	11.8	16.0	15.4	15.4	15.3
0.98 – 1.0	14.6	15.5	10.0	14.4	13.4	13.6	13.1
# of line segments detected	18.5	18.2	21.3	18.1	19.3	19.2	18.8
execution time (msec)	202	226	634	6164	3797	5757	701

Table 3. The result of Experiment 3.

match score	Boldt	Ours	Liang	Nacken	Princen	Risse	Yuen
0.0 – 0.9	3.8	2.4	22.7	5.1	8.8	5.5	17.4
0.9 – 1.0	15.1	16.2	5.3	14.4	12.7	14.2	8.5
0.98 – 1.0	13.7	14.8	3.8	12.7	9.1	12.0	5.1
# of line segments detected	18.9	18.5	28.0	19.5	21.5	19.8	26.0
execution time (msec)	200	231	624	6319	3716	5756	915

Table 4. The result of Experiment 4.

match score	Boldt	Ours	Liang	Nacken	Princen	Risse	Yuen
0.0 – 0.9	5.3	11.2	20.0	18.0	11.5	5.8	18.2
0.9 – 1.0	13.8	14.6	5.0	13.8	11.2	13.7	6.9
0.98 – 1.0	10.9	11.0	3.0	11.1	6.74	9.2	3.6
# of line segments detected	19.1	25.8	25.0	31.8	22.7	19.5	25.2
execution time (msec)	1400	833	700	50712	4482	6694	3141

small random perturbation is added to the orientation and the position of each short line segment (see Figure 4(c)). Finally, 300 noisy line segments, each of which is 3 pixels long and has an arbitrary orientation, are added to each test image in Experiment 4 (see Figure 4(d)). Note that the test images were not made incrementally as the experiments proceeded. A completely new set of test images was created for each experiment. We repeated the above experiments several times to obtain good parameter values for each method. It should be noted that the parameters of each method were set equal throughout all the experiments. The experiments were performed on a Pentium II(266MHz) PC and a programming tool used was Microsoft Visual C++ 6.0. The experimental results are shown in Tables 1-4, where given values are average values over 50 test images. In the tables, "match score" represents how much a detected line

segment resembles the true line segment. Thus, "match score = 1.0" means a perfect match between a detected line segment and the corresponding true one. Since a test image always contains 18 true line segments, a good line segment detector would give values close to 18.0 in the third to the fifth row of the tables. The tables show that the performance of our method is almost highest among all the methods tested in terms of detection capability and execution time throughout the entire experiments.

4 Conclusions

In this paper we have proposed a new method for finding globally optimal line segments. Our method overcomes the local nature of a conventional line segment grouping approach while retaining most of its advantages. Experimental results showed that our method is fast and has a good detection capability compared to the existing line segment detection methods.

References

1. J. Illingworth and J. Kittler, "A Survey of the Hough Transform," *Computer Vision, Graphics, and Image Processing*, Vol. 44, pp. 87-116, 1988.
2. V. F. Leavers, "Survey - Which Hough Transform?" *Computer Vision, Graphics, and Image Processing*, Vol. 58, No. 2, pp. 250-264, 1993.
3. H. Kälviäinen, P. Hirvonen, L. Xu, and E. Oja, "Probabilistic and Non-probabilistic Hough Transforms: Overview and Comparisons," *Image and Vision Computing*, Vol. 13, No. 4, pp. 239-252, 1995.
4. M. Boldt, R. Weiss, and E. Riseman, "Token-Based Extraction of Straight Lines," *IEEE Transactions on System, Man, and Cybernetics*, Vol. 19, No. 6, pp. 1581-1594, 1989.
5. P. F. M. Nacken, "A Metric for Line Segments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 12, pp. 1312-1318, 1993.
6. E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, pp. 114-117, 1998.
7. R. Haralick and L. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Vol. 1, pp. 563-565, 1992.
8. P. Liang, "A New Transform for Curve Detection," *Proc. International Conference on Computer Vision*, Osaka, Japan, pp. 748-751, 1990.
9. T. Risse, "Hough Transform for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection," *Computer Vision, Graphics, and Image Processing*, Vol. 46, pp. 327-345, 1989.
10. J. Princen, J. Illingworth, and J. Kittler, "A Hierarchical Approach to Line Extraction Based on the Hough Transform," *Computer Vision, Graphics, and Image Processing*, Vol. 52, pp. 57-77, 1990.
11. S. Y. K. Yuen, T. S. L. Lam, and N. K. D. Leung, "Connective Hough Transform," *Image and Vision Computing*, Vol. 11, No. 5, pp. 295-301, 1993.