A Real-time CAM-based Hough Transform Algorithm and Its Performance Evaluation

Mamoru Nakanishi
NTT Electronics Technology

Takeshi Ogura
NTT LSI Laboratories

E-mail: mamoru@center.nel.co.jp

Abstract

This paper proposes a real-time content addressable memory (CAM)-based Hough transform algorithm for straight line detection and evaluates its performance. Both voting and peak extraction, which compose the Hough transform, are directly executed by CAM. The CAM acts as a PE (Processing Element) array that performs highly parallel processing for the Hough transform and also as a memory for two-dimensional Hough space. To achieve high-speed processing, voting is executed in every scanning line, not every pixel. The Hough space is mapped into the CAM in folded form to reduce the size of the CAM hardware. Moreover, CAM-based weighted voting achieves more accurate line detection in spite of the quantization error and noise in the image space. Simulations of CAM hardware size, processing time and the accuracy of line detection show that a real-time and high-resolution Hough transform for a 256×256 picture can be achieved using a single CAM chip with current VLSI technology. This CAM-based Hough transform algorithm promises to be an important step towards the realization of a real-time and compact image understanding system.

1. Introduction

The Hough Transform [4], HT, is promising for line detection in image understanding and computer vision. The HT is defined by

$$\rho = x\cos\theta + y\sin\theta. \qquad (1)$$

The voting is performed to the points on the curve derived from Eq.(1) in a $\rho-\theta$ space (the Hough space). After this voting stage, lines are extracted by detecting peaks in the Hough space.

The HT is promising, but has two major drawbacks: excessive storage requirements and high computational complexity. To overcome these shortcomings, various methods have been intensively studied from the standpoints of both software and hardware technologies [5]. The software solutions have successfully speeded up the HT, but real-time processing has not yet been achieved. On the other hand, a high-resolution Hough transform has been achieved based on software technology [6].

Hardware solutions for real-time processing have also been proposed. These implementations are based on parallel processors, such as a mesh-connected array of processors [2], a systolic array [1] or a scan line array processor [3]. In general, conventional parallel processors that have sufficient parallelism for practical applications are not compact. Therefore, it is difficult to develop a compact line detection system based on these parallel processors. Moreover, a high-resolution HT based on hardware technology has not been achieved.

In order to attain a real-time and compact line detection system, we propose a CAM (Content Addressable Memory)-based HT algorithm. Both voting and peak extraction, which compose the HT, are directly executed by CAM. The CAM has sufficient parallelism for practical applications. It acts as a PE (Processing Element) array that performs highly parallel processing for HT and also as a memory for two-dimensional Hough space. To achieve high-speed processing, the voting algorithm is modified from conventional HT. Voting is executed in every scanning line, not every pixel. The Hough space is mapped into the CAM in folded form to reduce CAM hardware size. CAM-based weighted voting makes it possible to achieve a high-resolution HT in spite of the quantization error and noise in the image space.

The performance of the CAM-based HT algorithm, i.e., the reduction in CAM hardware size, the processing time and the accuracy of line detection, were evaluated by finite word-length simulation. The results show that real-time (video rate) and high-resolution HT for a 256×256 picture can be achieved using a single CAM chip with current VLSI technology. Moreover, because this CAM-based HT algorithm is scalable, real-time processing for a 512×512 picture is possible using four CAM chips.

2. CAM-based Hough Transform Algorithm

2.1. CAM and CAM-based System

The proposed HT algorithm was established on our CAM and CAM-based system technologies [7], [8]. A CAM achieves not only parallel search but also various types of parallel data processing. As a CAM-based system model, a Highly-parallel Integrated Circuits and System (HiPIC) was proposed for real-time image processing [8]. The HiPIC is an application-specific system that achieves high performance and flexibility.

As shown in Fig. 1, a HiPIC consists of a highly-parallel PE array, a reconfigurable logic element, RISC/DSP and memory. The highly-parallel PE array

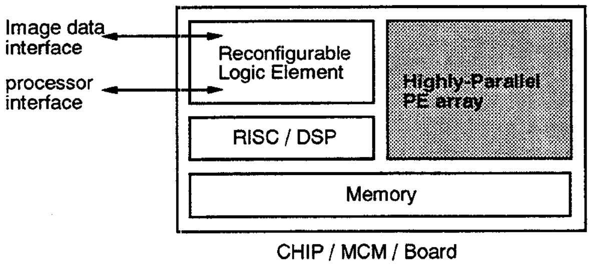


Figure 1. HiPIC: Highly-parallel Integrated Circuits and System.

(the shaded block in the figure) executes SIMD (Single Instruction, Multiple Data stream) processing for high-volume image data. The reconfigurable logic element controls the PE array. The RISC/DSP controls the whole system and executes serial data processing.

To achieve high performance on a HiPIC, sufficient parallelism in the highly-parallel PE array is crucial. CAM is most suitable for a highly-parallel PE array because a large number of PEs can be included in one VLSI chip. The CAM effectively performs general associative and arithmetic operations as a SIMD type PE array. The CAM operations used in our HT algorithm are maskable search, parallel write, partial write, and normal RAM operations. By combining the above operations, relational searches (less than, greater than, maximum, minimum, etc.) and fixed point calculations are executed in parallel.

2.2. Basic Hardware Algorithm

In the proposed HT algorithm, a CAM word corresponds to a quantization point in Hough space. A CAM word acts as a PE, a decision circuit and an accumulator. The CAM word configuration is shown in Fig. 2. The accumlator field is the memory of the Hough

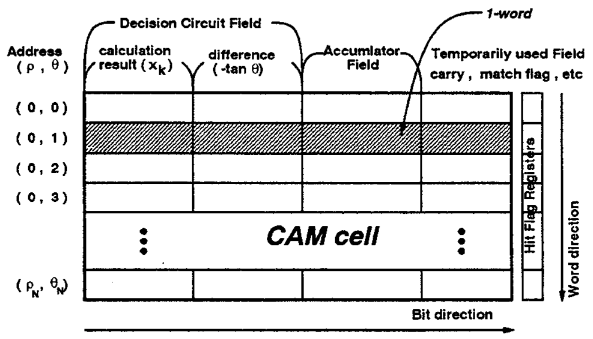


Figure 2. The arrangement of the CAM for the HT.

space. The decision circuit field, where the calculated result and the difference are stored, decides whether voting to the accumulator must be done or not. The decision is carried out by search operation of CAM. The calculation and search operation are carried out in parallel.

By modifying Eq. (1), we get

$$\begin{cases} x_k = x_{k-1} - \tan\theta_j \\ x_0 = \frac{\rho_i}{\cos\theta_j} + \frac{N}{2}\tan\theta_j \end{cases} \quad (0 \leq \theta_j \leq \tfrac{\pi}{4}, \tfrac{3\pi}{4} \leq \theta_j \leq \pi) \tag{2}$$

$$\begin{cases} y_k = y_{k-1} - \cot\theta_j \\ y_0 = \frac{\rho_i}{\sin\theta_j} + \frac{N}{2}\cot\theta_j \end{cases} \quad (\tfrac{\pi}{4} < \theta_j < \tfrac{3\pi}{4}) \tag{3}$$

where N is a size of an input images. In Eqs. (2) and (3), x_0 and y_0 are the initial values, and are set to the calculation result field in CAM at the start. The $(-\tan\theta_j)$ and $(-\cot\theta_j)$ in Eqs. (2) and (3) are set to the difference field in CAM. The x_k and y_k are renewed by Eqs. (2) or (3) in every horizontally or vertically scanning of the input image, respectively. The x_k is the x-coordinate of an intersection of each horizontal scanning line and a line associated with the CAM word. The y_k is the y-coordinate of an intersection of each vertical scanning line and a line associated with the CAM word. By searching words whose calculation result field is the same as the edge pixel coordinate, their match flag field is set in CAM. This process is executed for all x-coordinates of edge pixels in a horizontal scanning line ($y = k$) or y-coordinates of edge pixel in a vertical scanning line ($x = k$). After one-line scanning, voting to the accumulator in the CAM words, whose match flag is set, is carried out in parallel.

The whole sequence of the proposed algorithm is shown in Fig. 3. Computational cost is much reduced by using Eqs. (2) and (3) instead of Eq. (1), because Eqs. (2) and (3) can be iteratively calculated by only addition in every scanning line, not every pixel. Costly multiplication is not necessary. Moreover the voting and renewing coordinates are parallelly executed in CAM. Therefore, processing time is reduced dramatically.

```
1. set initial value to the CAM
2. For y= $-\frac{N}{2}$ to $\frac{N}{2}-1$
3.     For x= $-\frac{N}{2}$ to $\frac{N}{2}-1$
4.         if image (x,y) is an edge pixel
           then search (x) for calculation result field in CAM
                and set to match flags in coincided words
5.     parallel voting to accumlator field
6.     parallel renewing calculation result field
7. extracting local maximum in Hough space
```

Figure 3. Proposed CAM-based parallel HT algorithm.

2.3. CAM-based Weighted Voting

2.3.1. Rectangular-shape Weighted Voting for Quantization Error Suppression

The quantization error in image space is shown in Fig. 4 [6] and is represented by

$$|\delta x| \leq \frac{\Delta s}{2}, \delta y = 0 \quad \text{or} \quad |\delta y| \leq \frac{\Delta s}{2}, \delta x = 0 \qquad (4)$$

where Δs is the quantization size of the image space. Because of the error, incorrect peaks are generated or correct peaks are missed in Hough space by voting. Especially for high-resolution Hough transform, it is necessary to suppress this incorrect peak generation.

The deviation of Hough space ($\delta\rho_{max}$) caused by image space quantization error is represented by

$$\delta\rho_{max} = \frac{\Delta s}{2}\max\Big(|\cos\theta|, |\sin\theta|\Big). \qquad (5)$$

To avoid generating incorrect peaks, it is necessary to vote between $\rho - \delta\rho_{max}$ and $\rho + \delta\rho_{max}$, as shown in

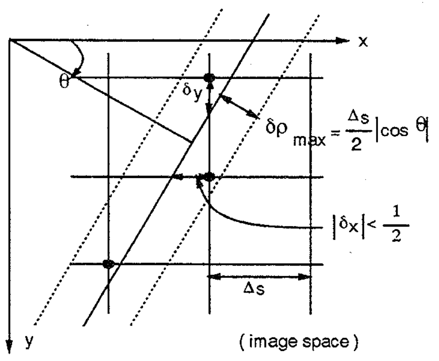


Figure 4. Quantization Error.

Fig. 5(a). The $\delta\rho_{max}$ varies in accordance with θ. It is hard to vote to this variable ρ region. Therefore, the computational complexity for this voting becomes higher in conventional Hough transform methods.

In the proposed CAM-based voting, as shown in Fig. 5(b), each CAM word calculates the coordinate of a crosspoint to the scanning line, and round the value of the calculation result field. Therefore, the CAM words whose calculated value are between $x - 0.5$ and $x + 0.5$ can be selected by search operation in CAM. By adding the constant to the accumlator field of selected words, a rectangular-shape weighted voting is easily achieved. The width of the rectangle is the same as the maximum quantization error size in an image. By this rectangular-shape weighted voting, more accurate line detection is possible.

2.3.2. Pentagonal-shape Weighted Voting for Noise Suppression

In addition to the quantization error in an image, noise in the image space generates incorrect peaks in Hough space.

The pentagonal-shape weighted CAM-based voting, which can suppress incorrect peak generation caused by both quantization error and noise, is also carried out effectively. The decimal value of the calculated value is used as another weighted value for addition to the accumulator, as shown in Fig. 6(a). These calculations are executed without additional hardware and with little additional processing time.

An example of the pentagonal-shape weighted voting is shown in Fig. 6(b). In the CAM words whose integer value coincides with the x-coordinate value 00000100(2), the decimal value is used as another weighted voting value. The decimal value deals with 2's compliment binary. If it is a positive number (MSB is 0), addition is executed. On the other hand, if it is

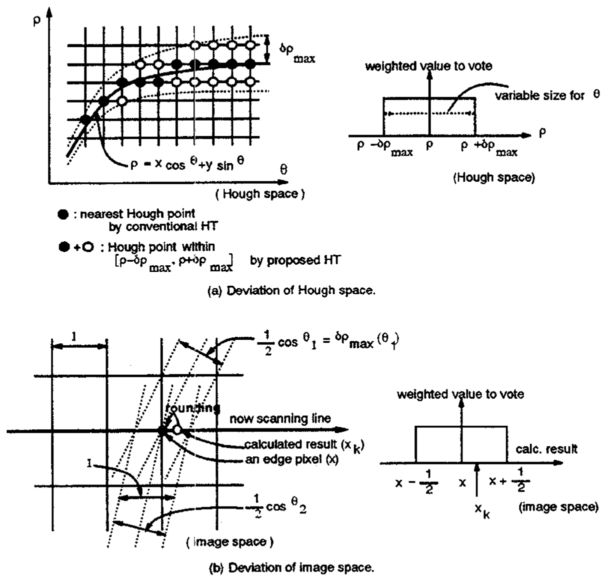
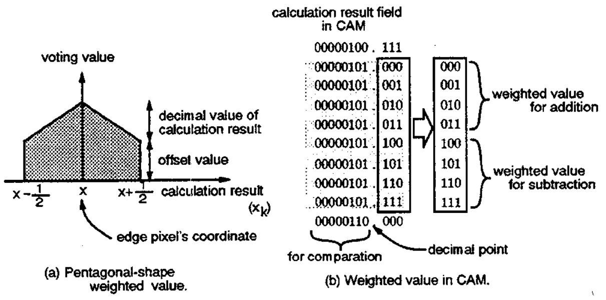


Figure 5. CAM-based rectangular-shape weighted voting.

a negative number (MSB is 1), subtraction is executed for positive weighted addition.



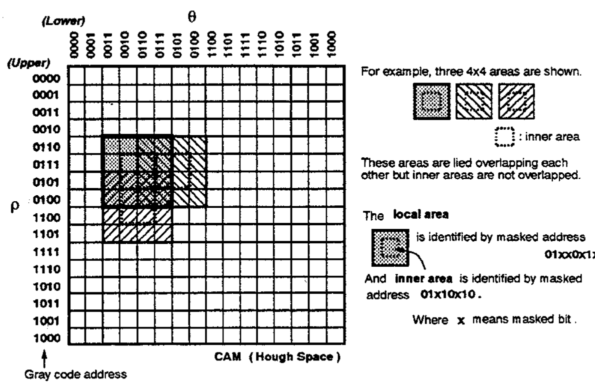
voting value
decimal value of calculation result
offset value
x − 1/2 x x + 1/2 calculation result (xk)
edge pixel's coordinate
(a) Pentagonal-shape weighted value.
calculation result field in CAM
00000100 . 111
00000101 . 000
00000101 . 001
00000101 . 010
00000101 . 011
00000101 . 100
00000101 . 101
00000101 . 110
00000101 . 111
00000110 . 000
weighted value for addition
weighted value for subtraction
for comparation
decimal point
(b) Weighted value in CAM.

Figure 6. Pentagonal-weighted voting.

2.4. Peak Extraction by Local Maximum Search

To detect lines in an input image, it is necessary to extract peaks in Hough space after voting. The computational cost of this sequential peak extraction is high. In our HT algorithm, the computational cost of peak extraction is low because peak extraction is carried out by thresholding and an iterative local maximum search for the accumulator using only CAM operations. If the maximum value is found in the inner part of the local area, this Hough point is regarded as a peak.

To identify the small square areas for local maximum search in the Hough space, Gray coding for the CAM address is introduced. By Gray coding, square areas, which are laid overlapping each other, can be easily identified as shown in Fig. 7.



(Lower)
θ
(Upper)
ρ
0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000
CAM (Hough Space)
Gray code address
For example, three 4x4 areas are shown.
: inner area
These areas are lied overlapping each other but inner areas are not overlapped.
The local area is identified by masked address 01xx0x1x.
And inner area is identified by masked address 01x10x10.
Where x means masked bit.

Figure 7. Identification of the local area by Gray coding for CAM word address.

2.5. CAM Hardware Size Reduction

Using the symmetry of image space, CAM hardware size can be reduced. The coordinate calculated in CAM word (i, j) can be used for the four Hough points: (ρ_i, θ_j), $(\rho_i, \frac{\pi}{2} - \theta_j)$, $(-\rho_i, \frac{\pi}{2} + \theta_j)$, $(\rho_i, \pi - \theta_j)$, which are respectively denoted H1, H2, H3, and H3 in Fig. 8 . The Hough points H1, H2, H3, and H4 correspond to lines L1, L2, L3, and L4 in the image space illustration in Fig. 8.

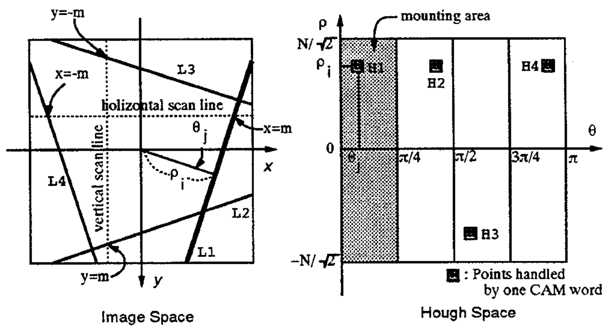


Figure 8. Folded mapping form on CAM and relationship to lines.

To utilize the symmetry, the CAM word configuration is changed in the accumlator field and the match flag field. These fields are divided into four sections.

If the image (m, k) is an edge pixel in horizontal scanning at $y = k$, CAM words whose calculation result is m are selected to vote to accumlator section-1. On the other hand, CAM words whose calculation result is $-m$ are selected to vote to accumlator section-4. In the same way, if the image (k, m) is an edge pixel in vertical scanning at $x = k$, CAM words whose calculation result is m are selected to vote to accumlator section-2, and CAM words whose calculation result is $-m$ are selected to vote to accumlator section-3.

In these ways, the Hough space can be mapped into CAM in folded form. The number of CAM words is reduced to 1/4 by implementing four accumulators to one CAM word. The total CAM hardware size (word×bit) is reduced to about 1/2.

3. Performance Evaluation

3.1. CAM Hardware Size and Processing Time

The evaluated CAM hardware sizes and processing times are summarized in Table 1. The bit-width shown in the table was estimated by finite word-length simulation. Using currently available 0.5-μm CMOS technology, a 336-kbit CAM has already been developed by the authors [8]. A 208-kbit CAM for a 64×64 picture can be integrated on a single chip. By time-sharing processing on the CAM chip, up to a 256×256 picture can be processed in real time, where the cycle time is 25 nsec. Moreover, real-time processing for a 512×512 picture can be achieved using four CAM chips.

Table 1. CAM size and execution time for real-time processing.

size of image space	CAM capacity (words×bits)	execution time (msec)	#CAM chips
64 × 64	4K×52	0.7	1 chip
128 × 128	8K×59	3.3	
256 × 256	16K×66	15.1	
512 × 512	32K×73	19.1	4 chips

3.2. Accuracy of Line Detection

The line detection performance was evaluated by finite word-length simulation. First, we examined the differences in the calculation results between Eq. (1) and Eqs. (2), (3). Then, we compared the peak extraction performance with the conventional method using ten input images (see Fig. 9) that contain fifty random lines. Finally, we totally evaluated the accuracy of line detection of our proposed HT algorithm using ten noisy input image (see Fig. 10) that contain twenty random lines.

The difference in the calculated values using Eq.(1) and Eqs. (2), (3) is summarized in Table 2. Table 2 indicates the ratio of the same quantization Hough points are calculated by Eqs. (2), (3) as by Eq. (1). The difference depends on calculation word length and input image size. Table 2 shows that values calculated from Eqs. (2), (3) with suitable word length are the same as those calculated by Eq. (1). Therefore, Eqs. (2) and (3) can be used instead of Eq. (1) under proper conditions.

Table 2. The ratio of correct quantizaion Hough points by finite word-length simulation.

word-length under decimal point	image size 128	256	512
6	97.4%	85.6%	—
7	100.0%	97.9%	83.6%
8	100.0%	100.0%	97.7%
9	100.0%	100.0%	100.0%

To compare the peak extraction performance between the proposed and conventional methods, we determined the dependency of the hit ratio of detecting lines, which is shown in Table 3. The local area of the proposed method is 4×4, and the inner area is 2×2. On the other hand, for conventional methods, both 3×3 and 5×5 filtering are applied. The hit ratio of the proposed method is almost the same as the conventional ones.

Table 3. Comparison of peak extraction performance between the proposed and conventional methods.

	size of local area		
	4×4	3×3	5×5
	(proposed)	conventional	
hit ratio	92.4%	92.2%	91.8%

To totally evaluate the accuracy of line detection, and estimate the contributions of the CAM-based weighted voting, we determined the dependency of the hit ratio on quantization size ($\Delta\rho$ and $\Delta\theta$) of Hough space, which is shown in Fig. 11. With non-weighted voting (conventional HT), the hit ratio decreases as the quantization size becomes smaller. On the other hand, with CAM-based weighted voting, the hit ratio is kept high. For noisy input image, the hit ratio of the pentagonal-shape weighted voting is higher than that of the rectangular-shape one. The figure shows that the proposed algorithm can achieve high-resolution HT.

4. Conclusion

A real-time CAM-based HT algorithm has been proposed, and its performance has been evaluated by finite word-length simulation. The proposed algorithm is easy to apply to VLSI technology, and is scalable for highly parallel processing.

The simulation showed that real-time and high-resolution straight line detection for a 256×256 picture can be realized using a single CAM chip. Moreover, real-time processing for a 512×512 picture can be achieved using four CAM chips.

The immediate goal is to develop a prototype of real-time and compact line detection system using the developed CAM [8] based on the HiPIC concept. The availability and usefulness of the current CAM-based HT algorithm will be clarified in the near future.

References

[1] R. E. Cypher, J. L. C. Sanz, and L. Snyder. A systolic array for straight line detection by modified hough transform. *Proceedings, IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pages 300–304, 1985.

[2] R. E. Cypher, J. L. C. Sanz, and L. Snyder. The hough transform has o(n) complexity on simd n×n mesh array architectures. *Proceedings, IEEE Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, pages 115–121, 1987.

[3] A. L. Fisher and P. T. Highnam. Computing the hough transform on a scan line array processor. *In Proceedings IEEE Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence*, pages 83–97, 1987.

[4] P. V. C. Hough. A method and means for recognizing complex patterns. *US Patent*, 3,069,654, 1962.

[5] V. F. Leavers. Survey: Which hough transform? *CVGIP: Image Understanding*, 58(2):250–264, 1993.

[6] M. Morimoto, S. Akamatsu, and Y.Suenaga. A high-resolution hough transform using variable filter. *ICPR'92*, 3:280–284, September 1993.

[7] T. Ogura and et al. A 20-kbit associative memory lsi for artificial intelligence machines. *IEEE Journal of Solid-State Circuits*, 24(4):1014–1020, 1989.

[8] T. Ogura, M. Nakanishi, T. Baba, Y. Nakabayashi, and R. Kasai. A 336-kbit content addressable memory for highly parallel image processing. *Custom Integrated Circuits Conference '96*, 1996.

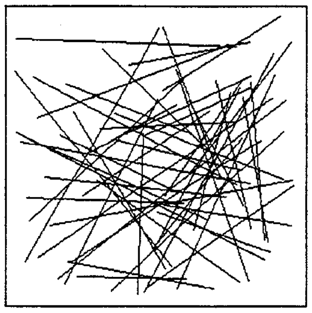


Figure 9. Input edge image without noise which is an example of an input 256×256 binary image with 50 lines.

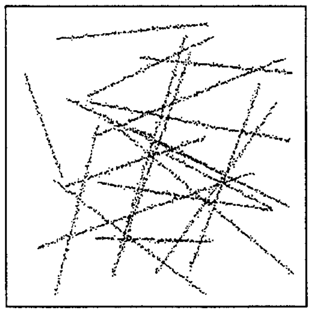


Figure 10. Noisy input edge image,which is an example of a noisy input 256×256 binary image with 20 lines.

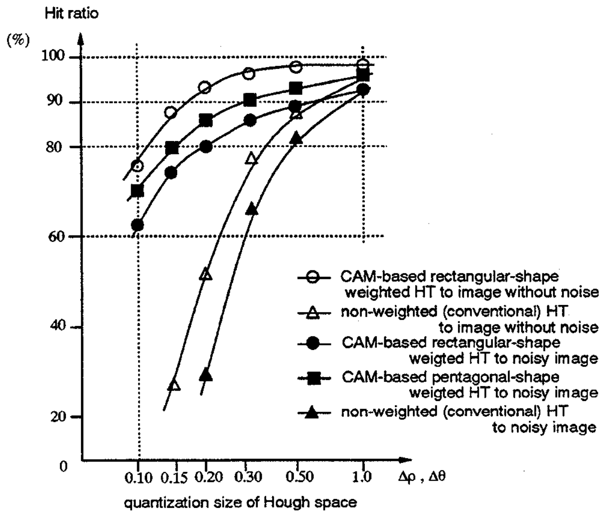


Figure 11. Accuracy of line detection.