Real-time implementations of Hough transform on SIMD architecture

Citation for published version (APA):

He, Y., Zivkovic, Z., Kleihorst, R. P., Danilin, A., & Corporaal, H. (2008). Real-time implementations of Hough transform on SIMD architecture. In *Second ACM/IEEE International Conference on Distributed Smart Cameras, 2008 : ICDSC 2008 ; 7 - 11 Sept. 2008, Stanford University, [Palo Alto, CA]* (pp. 1-8). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/ICDSC.2008.4635716>

DOI:

[10.1109/ICDSC.2008.4635716](https://doi.org/10.1109/ICDSC.2008.4635716)

Document status and date:

Published: 01/01/2008

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

REAL-TIME IMPLEMENTATIONS OF HOUGH TRANSFORM ON SIMD ARCHITECTURE

Yifan He^{1,2}, Zoran Zivkovic¹, Richard Kleihorst¹, Alexander Danilin¹, Henk Corporaal²

¹ NXP Semiconductors, the Netherlands
{yifan.he, zoran.zivkovic}@nxp.com

² Technische Universiteit Eindhoven, the Netherlands
y.he.1@student.tue.nl, h.corporaal@tue.nl

ABSTRACT

Implementation of the Hough Transform (HT) for line detection requires massive computation, large memory space and high bandwidth. Without parallel processing on a proper platform, it can be hardly implemented in real-time, especially with high accuracy on high-resolution images. This paper proposes several efficient methods for implementation of HT on SIMD (Single-Instruction, Multiple-Data) architecture. All lines in an image/frame can be detected in real time with the proposed voting method, the novel Hough Space (HS) structures, and the efficient image “rotation” mechanism. With suggested refinement and tracking approach, we can capture and follow the target lines with very high accuracy. Analysis and comparison are elaborated with real-time implementation on our Wireless Smart Camera (WiCa) platform, which is a powerful image/video processing platform developed by NXP Semiconductors. Moreover, two different applications, lane detection and gesture control, are also developed on this platform.

Index Terms— Hough Transform, Real-Time, SIMD, WiCa, Gesture Control

1. INTRODUCTION

Line detection is a classical problem in the image processing and computer vision field. Extracting target lines efficiently and accurately are two important requirements, especially for real-time applications. The Hough Transform (HT) [1] is a well-established and robust tool to locate straight lines in images in presence of noise and occlusion. Essentially, HT for line detection is a voting process where each feature (edge) point votes for all the possible lines passing through that point, which can be broadly described as a 2-D histogramming. These votes are then accumulated in an accumulator array (Hough Space, HS), and the cells in HS which receive maximum votes are recognized as detected lines. Thus, the HT converts a difficult global detection

problem in image space into a more easily solved local peak detection problem in a parameter space [2].

In general, two sets of parametric representation are used for HT: slope-intercept based (proposed by Hough [1]), and (ρ, θ) based, where ρ is the length of the unique chord from the origin perpendicular to the line, and θ is the angle of that chord clockwise from the positive x axis (suggested by Duda and Hart [3]). The main difference of the first representation with respect to the second is the unbounded parameter space. This is also the main reason why in practice (ρ, θ) based representation is much more popular than slope-intercept based representation.

The Hough Transform can cope with noise, gaps in outlines and partial occlusion, even in complicated backgrounds. In spite of its superb recognition ability, the major drawbacks as massive computation, large memory capacity and bandwidth requirements have prevented it from being used in real-time applications. Aiming at real-time execution, many architectures and mapping methods were introduced for HT in the past 30 years: from systolic array [5, 6], pipelined system [4, 7], 1-D or 2-D SIMD processors [8-10], MIMD (Multiple-Instruction, Multiple-Data), to other dedicated hardware [11]. Basically speaking, implementations on general purpose architectures are more preferred in practice, as ASIC or reconfigurable-logic solutions are used to be too costly for a single algorithm, especially for commercial usage. However, only few of the reported works on general purpose platforms have both real-time and high-accuracy performance on even VGA format [12]. Moreover, the processing time of these methods generally depends on the number of edge points, which is usually not preferred for practical usage.

In this paper, an improved slope-intercept like representation is proposed for efficient implementation of HT on 1-D SIMD architectures. The processing time is independent of the number of edge points or the number of detected lines. Two methods, dual Hough Space and interleaved Hough Space, are suggested to capture lines during each pass of HT. We introduce an efficient image

Fig.1. The WiCa Platform

“rotation” mechanism with the help of an off-chip memory, which solves a main problem in the Twin HT [13] concept. We also propose two refinement & tracking realizations to capture and follow the target lines with high accuracy. All of these proposed approaches are implemented with real-time demos on our Wireless Smart Camera (WiCa) platform, which is a powerful image/video processing platform containing a high-performance SIMD processor (Xetal). Finally, two applications, lane detection and gesture control, are developed based on the proposed implementation approaches.

The rest of this paper is organized as follows. Section 2 gives a brief introduction of the WiCa platform, focusing on the high-performance Xetal SIMD processor. Section 3 contains the details of proposed implementation strategies, as well as the analysis and comparison of them with other reported approaches. Section 4 introduces the refinement & tracking algorithms. Applications and experimental results are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. WIRELESS SMART CAMERA (WiCa) PLATFORM

The WiCa platform (see Fig. 1), developed by NXP Research, is an image/video processing platform which gives very-high pixel processing performance. It mainly consists of a high-performance, power-efficient SIMD processor (Xetal-I/IC3D/Xetal-II) for low-level image processing, a general purpose processor for intermediate and high-level processing and control, a dual-port RAM (DPRAM) for inter-frame processing and (internal) data communication, one or two VGA color image sensors, and a ZigBee transmitter for communication [14, 15].

The Xetal processor is the key component of the WiCa platform. IC3D, the 2nd generation IC in the Xetal family, is a power-efficient, fully programmable IC with a peak performance of 50 GOPS (Giga-Operations Per Second) when working at 80MHz and 1.8V, while dissipating less than 500mW. A linear processing array (LPA) consisting of 320 processing elements (PEs) and 64 320×10-bit on-chip line memory handles the compute-intensive data processing (see Fig. 2). At 80MHz, IC3D memory interface provides

Fig. 2. Block Diagram of IC3D Architecture

1 Tb/s bandwidth. The high computational power and internal data bandwidth of Xetal processor provide a good opportunity for real-time implementation of algorithms like Hough Transform.

3. THE PROPOSED IMPLEMENTATION METHODS

The Hough Transform is a both computation-demanding and bandwidth-demanding algorithm. Take a VGA sequence (640 pixels × 480 pixels, 30 frames/s), $N_\theta = 30$ (the number of quantization in the θ space, 6° accuracy), and CORDIC [16] implementation of $\sin(\theta)$ and $\cos(\theta)$ as an example, the total computation is already above several GOP/s, not to mention higher-resolution sources and higher accuracy. Besides that the frequent accesses to the image and large HS in the memory require huge amount of bandwidth support in a real-time realization. These together prevent HT to be implemented in a general purpose processor in real time. However, the intrinsic parallelism due to pure pixel-based feature of HT makes it extremely suitable for SIMD architectures.

The complete HT for line detection can be divided into a set of subtasks: a) edge detection; b) voting; c) Hough Space post-processing; d) detected line displaying. The Canny operator [17] is used in our implementation for edge detection. As this step is not the focus of this paper, only steps b) ~ d) will be discussed in detail.

In the following sections, we are going to propose several approaches for real-time implementation of HT on 1-D SIMD architectures. A slope-intercept like representation is introduced together with efficient voting process. Two different methods, dual Hough Space and interleaved Hough Space, are suggested to capture all lines in every half plane. We also proposed an efficient image “rotation” mechanism to fulfill the twin HT concept. All approaches below are mapped onto our WiCa smart camera platform with VGA format sequence as input.

3.1. Basic Implementation

Li's Equation: $b = x + y \cdot \tan(\theta)$

Fig. 3. A 1-D SIMD implementation similar to Li's method

Fig. 3 depicts an implementation similar to Li's method [8]. Equation $b = x + y \cdot \text{tg}(\theta)$ is used to represent a line, where x and y are pixel coordinates, θ is the angle between the target line and y axis, and b is the intercept in the x axis. A line (represented by edge points) can be detected by shifting the edge-image line-by-line and then sum the columns in parallel. However, in Li's method, for every value of θ , the whole frame is shifted before summing up the columns. Multiple passes (depended on the number of different θ) on the same image are required, and the whole image has to be stored, which are not very appropriate for a real-time embedded platform. And $\text{tg}(\theta)$ calculation here is also inefficient in both computation time and accuracy. Moreover, Li did not solve the problem of image rotation, which costs huge amount of time and bandwidth in his implementation.

Proposed Equation: $b = x + y \cdot m$ for 1-D SIMD Processors

Instead of using the equation $b = x + y \cdot \text{tg}(\theta)$, $-45^\circ \leq \theta \leq +45^\circ$, we choose to use the $b = x + y \cdot m$, $-1.0 \leq m \leq +1.0$ representation according to the characteristics of the 1-D SIMD architecture. Note that, here m - b does not have the same meaning as the standard slope-intercept representation mentioned in Section 1 (However, we still prefer to call it a slope-intercept like representation). One obvious improvement in this representation is that only a one-cycle MAC (Multiple-Accumulate) instruction is required instead of multiple cycles due to removal of the $\text{tg}(\theta)$ calculation. The accuracy of the calculation results is also improved (Note that number is represented by limited bits in hardware. More calculation on the same data means more accuracy loss. This is especially visible in an embedded processor.)

In our method, a line-based fashion is adopted instead of multi-pass processing on the same image. HS is stored in the line memory of the SIMD processor, and each line memory in the HS stands for a "Hough Line", which has the same m value but different b values in each cell. The voting procedure is done as follows: When an input video line is received (after edge detection step, so the pixel value is 0 or 1), we start from $m = 0$, which merely adds the video line to the corresponding Hough Line. This process costs only 1

Fig. 4. Proposed implementation method on a SIMD Processor

cycle in the Xetal processor. After that m is increased by Δm (say 0.02, roughly 1°), and new shifting amount $(h-y) \cdot m$ is calculated. Here h is the image height, and y is the index of current row. We call it a "hitting the bottom" method, as all edge points on the same line will vote to the cell, whose x coordinate is the same as that of the cross point where the line hits the image bottom (See Fig. 4). (Note that we can also use $y \cdot m$ instead. Then the physical meaning becomes "hitting the top" instead of "hitting the bottom"). The same video line is shifted to the right (for $m > 1$), and a copy is shifted to the left (for $m < 0$). Votes (the shifted video line) are added to the corresponding Hough Line. This process is continued till m reaches the last value (1.0). Then a new video line is read in, and the current video line is sent to the LCD screen for displaying, or stored for other following applications, or simply discarded.

After processing the last video line, the whole HS is built in the line memory, and post-processing can be applied on it. As we only process a limited number of m with step Δm , a practical line could fall in between two processed m values with its voting spreading among several neighboring cells instead of concentrating into one. These lines can still be easily captured by adding the votes of cells in HS to their neighboring ones. With local peak detection and a global threshold, target lines are detected.

The final step is to display the detected line on the LCD screen. An inversed process to the voting procedure is used. Instead of shifting the input video lines, Hough Lines are shifted to reproduce the detected lines. With this method, lines can be displayed on the screen efficiently, independent of the number of total detected lines.

3.2. Capture All Lines in Every Half Plane

In Section 3.1, the basic approach to implement HT on a 1-D SIMD processor is introduced. By applying this on a frame and its 90° -rotated version (Twin HT), all lines are expected to be detected. The precondition is that every pass must be able to capture all of the lines in its half-plane (i.e. $[-45^\circ, +45^\circ]$ for the original pass, and the other half for the

Fig. 5. Some lines are “missing” if we only apply the basic approach

rotated pass). However, this precondition does not hold true if we only apply the basic method introduced in the previous section. Fig. 5 gives an example: Two lines, which are within the range of $[-45^\circ, +45^\circ]$, are “missing” as they can not “hit” the image bottom.

Dual Hough Space

In order to solve this problem, a property of lines in the half-plane is exploited: for any line between -45° and $+45^\circ$ in an image of size $N \times N$, it will either “hit” the top border, or hit the bottom border, or hit both. According to this, a “hitting the top” concept is also introduced together with the “hitting the bottom” method, which is discussed in Section 3.1. For every pass, Dual Hough Spaces are built, which are called “Top Hough Space” and “Bottom Hough Space” respectively. Fig. 6 depicts this idea. By using this approach, the problem described in Fig. 5 can be solved.

Applying both hitting the top and hitting the bottom will not double the processing time, though two HT are called per video line. The most time-consuming processing in our implementation is shifting. The shifting amount for every video line depends on the current row number and parameter m (the maximum value of m is 1.0). Equation (1) and (2) below give the total shifting amount for every video line when “hitting the top” and “hitting the bottom” is used respectively (h is the image height, and Row is the current row number). A multiplication of 2 is because we need to shift the video line both to the right and to the left. By summing up (1) and (2), a constant result $2h$ is achieved, which is the worst-case shifting amount in both (1) and (2). As system performance is decided by the worst case, using both “hitting the top” and “hitting the bottom” will not degrade the performance significantly. Only some overhead like extra addition and multiplication is introduced. The performance results in Table 1 (the “Worst Case” column) in Section 5 show this clearly.

$$\text{Shift (hitting the bottom)} = 2 * (h - \text{Row}) \quad (1)$$

$$\text{Shift (hitting the top)} = 2 * \text{Row} \quad (2)$$

$$\text{Total shift} = 2 * h = \text{constant} \quad (3)$$

Fig. 6. Using both “hitting the top” and “hitting the bottom” methods

Fig. 7. Using odd and even cells to capture the lines which hit the bottom inside or out of the image border respectively

The main drawback of this approach is that the memory requirement for HS is increased. As line memories are usually critical resources in a SIMD processor, the total size of HS (thus, the quantization grain) is limited.

Interleaved Hough Space

A line memory consists of 320 cells in our Xetal processor. Thus, the number of quantization in the parameter b space is 320 (If two line memories are used for every m , then the number of quantization of b is 640, which is not discussed here). If this high number of quantization is not required, an alternative solution for detecting all of the lines in each half plane could be designed with the cost of reduced quantization grain of b . Fig. 7 describes this approach. A line memory (Hough Line) is split into two equal parts (odd and even, 160 cells each). Half of the cells (odd cells, denoted by yellow color in Fig. 7) in a line memory are used to capture the lines which hit the image bottom inside the image border, and the other half (even cells, denoted by blue color) captures the lines which hit the bottom out of the image border. The interleaved fashion of this method makes it very easy for voting, and the cost of shifting is also

minimized. Thus, all lines can be detected, but only one HT is required for each video line. The memory requirement for HS also remains the same as the basic approach. The drawback of this method, comparing to the dual HS solution, is that the processing for each video line is a bit more complex and costs more cycles.

3.3. Twin Hough Transform with Efficient Image “Rotation”

Now we are capable of implementing HT in real time to detect all lines in the half plane ($-45^\circ \sim +45^\circ$). In order to implement the full-plane HT, only a same pass on the 90° -rotated image is required. However, rotation of a whole image usually costs too much time, which makes it a main bottleneck in the twin HT concept.

As introduced in Section 2, a DPRAM is included in the WiCa platform as an off-chip memory shared by the Xetal processor and the 8051 (an off-chip memory is commonly used in an embedded system to supply enough data storage capacity). In our implementation, an efficient image rotation mechanism is proposed with the help of this DPRAM. During processing the first pass on the input frame, the input video line is stored into the DPRAM with a normal write mode (store the frame from left to right, and top to bottom). This is done by the control processor, so transfer time is hidden to the HT. When processing the other pass on the 90° -rotated image, image data is read back from DPRAM line by line with a different mode (from bottom to top, and left to right). Thus, the rotated image data is acquired without really rotating the whole image. This “rotation” mechanism is very efficient, as the transfer time for reading is also done in parallel with the remaining part of the program.

Fig. 8 depicts one of the suggested implementation approach of the twin HT with the efficient image “rotation” mechanism. The normal pass is processed during the even frame period, while the other pass on the 90° -rotated image is processed during the odd frame period. All lines can be captured, and processing time is independent of the image content (number of edge points or number of detected lines), which is also a very important feature.

The drawback of this approach is that only half of the frames, either even or odd, are processed. Down-sampling is applied at the frame level. Another approach is presented in Fig. 9, which is more preferred for applications when working on a sub-sampled image is sufficient enough. In this approach, down-sampling is applied at the row level. The twin HT is applied to every input frame in real time.

Thus, the full implementation of HT is realized on the WiCa platform. Compared to the implementation on IMAP, another 1-D SIMD processor [9], the proposed implementation has the following features: 1) One instruction to generate the vote; 2) Local-addressing is NOT required (to support local-addressing, both area and complexity of the SIMD processor will severely increase due to extra logic);

Fig. 8. Twin HT with efficient image “rotation”: down-sampling at frame level

Fig. 9. Twin HT with efficient image “rotation”: down-sampling at row level

3) Content Independent (processing time are independent of the number of edge points or detected lines); 4) Efficient Shifting and Image “Rotation”; 5) Efficient Line Displaying, which is independent on the number of total detected lines.

4. REFINEMENT & TRACKING

In some applications (e.g. lane detection), we are interested in the dominant lines, and want to capture and track them with high accuracy. However, a simple HT usually outputs too much irrelevant information, and fails to detect the target lines with high accuracy.

Though the “coarse-to-fine” idea has already been proposed by Illingworth and Kittler [18], here we combine it with tracking and realize it in two ways. Fig. 10 presents a refinement and tracking algorithm, which enable us to capture and follow the target lines with high accuracy in real time. During the coarse-grain phase, all lines are detected with relatively low accuracy, but only the dominant lines are picked up for the following fine-grain phase. The fine-grain phase follows the detected line and applies fine-grain HT on a limited parameter space around these detected lines. Only

Fig. 10. Refinement and Tracking algorithms: coarse-grain and fine-grain phases are on different frames.

Fig. 11. Refinement and Tracking algorithms: coarse-grain and fine-grain phases are on the same frame.

if lines are lost in tracking or if the system stays in the fine-grain phase for a preset time, the coarse-grain phase is visited again. The coarse-grain phase will consume one-frame time. However, it is only revisited occasionally.

If only part of the image is of interest, we can also apply the two phases on every frame. The coarse-grain phase is applied on the interesting part of the image. During scanning the non-interesting part, the fine-grain phase is called, which works on the same data used by the coarse-grain phase. Fig. 11 depicts this algorithm.

5. EXPERIMENTAL RESULTS

All the proposed methods discussed above are implemented on our WiCa platform. In this section, we will analyze the

Fig. 12. These are the snapshots from the real-time demos based on the methods discussed in Section 3. Detected lines are overlaid with green color on the (white) edge image

performance of the voting step (which is the most time/resource-consuming step), and present some experimental results. After that, two applications based on the suggested mapping approaches are introduced.

Table 1 shows the resource usage of the three proposed implementation approaches, which are discussed in Sections 3.1 and 3.2. The input is a VGA sequence at 30fps (640 pixels $\times$ 480 pixels), and the step of m is 0.067 (3° precision on average). The total line memory available in IC3D is 64, and the total processing time available between two video lines are about 5500 cycles. Compared to the other two methods, the interleaved HS consumes more cycles. This is mainly because we need a different line memory shifting mode which costs more cycles than the normal shifting mode in our IC3D processor. The twin HT and the refinement & tracking algorithm only apply one of these three methods on each video line, so the resource usage are almost the same (Several extra cycles and line memory are required to initialize the DPRAM operation).

Fig. 12 presents some snapshots from the real-time demos on our WiCa platform based on the methods discussed in Section 3. The accuracy of detected lines here is roughly 3 degrees.

5.1. Lane Detection

The refinement and tracking algorithms could be used to

Table 1. Resource usage (the voting step) of the three proposed implementation approaches

	Average (cycles/video line)	Worst Case (cycles/video line)	Line Memory Usage	Detect All Lines?
Basic	1500	2700	30	No
Dual Hough Space	2985	2985	60	Yes
Interleaved Hough Space	2211	3891	30	Yes

Fig. 13. The lane detection results: the top one is the original image, and the bottom one is the snapshot from the screen with the detected lines displayed in green color ($< 1^\circ$ precision)

Fig. 14. The gesture control application: on the left bottom of each image is our WiCa platform. On the top of the TV screen, 10 solid buttons, filled by red color, are placed. The directions of human arms are detected and displayed on the TV Screen with red lines. When the lines hit the red buttons, they will be activated (turn to a hollow button on the TV screen)

extract the road boundaries. During the coarse-grain phase, the lane borders (dominant lines) are picked up with relatively low accuracy, and the data is fed into the fine-grain phase. High-accuracy HT is applied on the neighboring parametric space of these selected lines, and accurate information ($< 1^\circ$ precision) is achieved during the fine-grain phase. The lane border information is updated at the end of each fine-grain phase, and the new data is fed into the following fine-grain phase, guiding the new searching region. Fig. 13 shows some snapshots of detected results from our demo.

5.2. Gesture Control

Another interesting application based on our implementation approaches is the gesture control, which can be useful in many fields. In our demo, we take the directions of the arms to control the buttons on the TV screen, which means that instead of pushing the buttons, we only need to wave our arms to do remote control. Fig. 14 presents the results from our demo. When the buttons are hit, they will become active. In this implementation, some extra preprocessing, like contour processing, is required. We keep the contour of the body, but remove all the other detailed information.

6. CONCLUSIONS

In this paper, we have shown several real-time implementation methods of HT on 1-D SIMD architecture, including 1) a slope-intercept like implementation which is efficient for voting generation; 2) dual HS and interleaved HS to capture all lines in each half plane; 3) efficient image “rotation” and line displaying mechanisms; 4) refinement & tracking method to capture and follow the target lines with high accuracy. All of these proposed approaches are implemented with real-time demos on our WiCa smart camera platform. Noise is not a main concern in our implementation, as HT algorithm itself is quite robust to noise. Thus, the SNR requirement to the sensors is also not critical.

The main features of our implementations are: 1) real-time; 2) detect all lines; 3) high accuracy ($< 1^\circ$ precision for parameter m , and 1-pixel accuracy for parameter b); 4) one instruction to generate the vote; 5) local-addressing is NOT required; 6) content independent (processing time is independent of the number of edge points or the number of detected lines); 7) efficient image “rotation”; and 8) efficient line displaying.

Two applications, lane detection and gesture control, are developed on the WiCa platform. The experiment results prove that the proposed implementation approaches are suitable and efficient for 1-D SIMD architecture with real-time, high-accuracy performance. We also show that the low-cost, low-power WiCa platform is extremely powerful for computation-demanding image/video processing.

7. REFERENCES

- [1] P. Hough, "Method and Means for Recognizing Complex Patterns," U.S. Patent No. 3,069,654, (1962).
- [2] J. Illingworth, and J. Kittler, "A Survey of the Hough Transform. Computer Vision," *Graphics, and Image Processing*, Vol. 44 (1988) pp. 87-116.
- [3] R. Duda, and P. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Communications of the ACM*, Vol. 15 (1972) pp. 11-15.
- [4] K. Hanahara, T. Maruyama, and T. Uchiyama, "A Real-Time Processor for the Hough Transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10 (1988) pp. 121-125.
- [5] L. Costa, and M. Sandler, "A Binary Hough Transform and its Efficient Implementation in a Systolic Array Architecture," *Pattern Recognition Letters*, Vol. 10 (1989) pp. 329-334.
- [6] B. Schmidt, M. Schimmler, and H. Schroder, "Morphological Hough Transform on the Instruction Systolic Array," *The 3rd Euro-Par Conference on Parallel Processing* (1997) pp. 798-806.
- [7] A. Majumdar, "Design of an ASIC for Straight Line Detection in an Image," *Proceedings of the 13th International Conference on VLSI Design* (2000) pp. 128-133.
- [8] Z. Li, F. Tong, and R. Laughlin, "Parallel Algorithms for Line Detection on a 1xN Array Processor," *International Conference on Robotics and Automation* (1991).
- [9] N. Yamashita, Y. Fujita, and S. Okazaki, "An Integrated Memory Array Processor with a Synchronous-DRAM Interface for Real-Time Vision Applications," *Pattern Recognition*, Vol. 4 (1996) pp. 575-580.
- [10] A. Rosenfeld, J. Ornelas, and Y. Hung, "Hough Transform Algorithms for Mesh-connected SIMD Parallel Processors," *Computer Vision, Graphics, and Image Processing*, Vol. 41 (1988) pp. 293-305.
- [11] M. Mahmoud, M. Nakanishi, and T. Ogura, "Hough Transform Implementation on a Reconfigurable Highly Parallel Architecture," *Computer Architecture for Machine Perception*, (1997) pp. 186-194.
- [12] M. Albanesi, M. Ferretti, and D. Rizzo, "Benchmarking Hough Transform Architectures for Real-Time," *Real-Time Imaging*, Vol. 6 (2000) pp. 155-172.
- [13] T. Risse, "Hough Transform for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection," *Computer Vision, Graphics, and Image Processing*, Vol. 46 (1989) pp. 327-345.
- [14] R. Kleihorst, B. Schueler, and A. Danilin, "Architecture and Applications of Wireless Smart Cameras (Networks)," *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 4 (2007) pp. 1373-1376.
- [15] R. Kleihorst, A. Abbo, et al., "Xetal: A Low-Power High-Performance Smart Camera Processor," *IEEE International Symposium on Circuits and Systems*, Vol. 5 (2001) pp. 215-218.
- [16] J. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Transactions on Electronic Computing*, Vol. EC-8 (1959) pp. 330-334.
- [17] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8 (1986) pp. 679-698.
- [18] J. Illingworth, and J. Kittler, "The Adaptive Hough Transform," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9 (1987) pp. 690-698.