Energy-Efficient Hardware Implementation of Road-Lane Detection Based on Hough Transform with Parallelized Voting Procedure and Local Maximum Algorithm

Jungang GUAN†, Fengwei AN††, Xiangyu ZHANG†, Lei CHEN††, Nonmembers, and Hans Jürgen MATTAUSCH†, Member

SUMMARY  Efficient road-lane detection is expected to be achievable by application of the Hough transform (HT) which realizes high-accuracy straight-line extraction from images. The main challenge for HT-hardware implementation in actual applications is the trade-off optimization between accuracy maximization, power-dissipation reduction and real-time requirements. We report a HT-hardware architecture for road-lane detection with parallelized voting procedure, local maximum algorithm and FPGA-prototype implementation. Parallelization of the global design is realized on the basis of θ-value discretization in the Hough space. Four major hardware modules are developed for edge detection in the original video frames, computation of the characteristic edge-pixel values (ρ, θ) in Hough-space, voting procedure for each (ρ, θ) pair with parallel local-maximum-based peak voting-point extraction in Hough space to determine the detected straight lines. Implementation of a prototype system for real-time road-lane detection on a low-cost DE1 platform with a Cyclone II FPGA device was verified to be possible. An average detection speed of 135 frames/s for VGA (640x480)-frames was achieved at 50 MHz working frequency.

key words: Hough transform, parallelized voting procedure, local maximum, video-based road-lane detection, energy efficient

1. Introduction

Real-time road-lane detection is of great importance for traffic safety as it is the major contributor to a lane-departure-warning (LDW) system, which is a basic and necessary part of an advanced driver assistant system (ADAS) [1]. LDW systems use the information from road-lane detection to warn the driver of imminent lane departure, so that the driver can correct the route to avoid potential accidents. Due to its significance, many researchers pay attention and work on the development of LDW systems to support the advances in self-driving technologies. It is a general consensus that a real-time, stabilized and accurate road-lane detection method is of high necessity for autonomous vehicle navigation.

At present, the lane detection approaches can be mainly divided into two categories, sensor-based approaches and image-processing-based approaches [2].

As the name suggests, the sensor-based approaches use devices such as radar, laser, and even global positioning system (GPS) to mark the locations of road lane and vehicle by analyzing the captured sensor information. From the analysis results the danger of road-lane departure is estimated in real time. These sensor-based approaches can provide high reliability, especially under bad weather conditions. However, their accuracy for detecting the road-lane positions is usually a concern, so that researchers often combine sensor data with specific eigenvalue algorithms to obtain an increased accuracy. For example, Li et al. used a multi-sensory scheme with local Otsu thresholding method and PPHT (Progressive probabilistic Hough Transform) to execute a conditional lane detection algorithm for detecting the optimal drivable region [3].

The image-processing-based approaches use various features of captured images such as gradient [3], [4], color [5]–[7], histogram [8], or texture (edge) for road-lane detection. Yoo et al. [4] proposed a linear discriminant analysis (LDA)-based gradient-enhancing conversion method for road-lane detection, which aims at solving the problem of changing gradient values between lanes and road due to different illumination, thus degrading the performance of the road-lane detection system. This method requires approximately 50 ms for frame processing with an average detection rate of 96%. Chiu et al. [5] proposed another method applying color-based image segmentation with a quadratic-function analysis to find out and approach the road-lane boundary in a selected region of interest. As color-based segmentation is sensitive to ambient light and requires additional processing to avoid undesirable effects, it is not suitable for roads with complex illumination environments. Munajat et al. [8] combined RGB histogram filtration with boundary classification to obtain a new approach for road-lane detection. The RGB histogram filtration is used to process the camera output by the color-based segmentation for determining the road area. On the other hand, the boundary classification is used to map roads and their frequently changing environments based on the RGB indexing. Finally, Hough transform (HT) and Canny edge detection are used to look for the road-lane boundaries.

Application of HT is one of the image-processing-based approaches for road-lane detection, as it is robust against many problems like line gaps or noise under real-world conditions.
Generally, HT-based road-lane detection is composed of four processing steps. Firstly, the input-frame image captured by the camera is down sampled or unimportant parts are removed. The goal of this step is to reduce the computational load e.g. by choosing a region of interest (ROI) for the follow-up processing [9]–[12]. The ROI should of course contain the necessary parts of the road lanes in the original input image. Another method of reducing computational load is based on specific filters to eliminate the influence of noise pixels by the way of smoothing the input image. The commonly used filters are median filter [13] and KNN filter [14]. In the next step graying and edge detection are applied for the determined ROI to extract the contained edge pixels for HT execution. The number of edge pixels directly determines the HT-computation amount. A suitable edge-detection algorithm often leads to a better result of the road-lane detection. Sobel [15], Canny [16] and Morphology [3] are the popular operators for edge detection. The third and fourth processing steps are HT itself and then road-lane tracking. The function of HT is to determine the polar coordinates connected to straight lines in the image frame, while the road-lane tracking part labels the detected lanes based on these polar coordinates.

The core HT part can be viewed as a mathematical approach for gathering straight-line evidence from the edge pixels in an accumulator array for polar coordinates, which is followed by a voting process dependent on the accumulation level. Based on specific mathematical rules, HT defines a mapping process for edge pixels from Cartesian coordinate space to Polar coordinate space (Hough space). For this purpose, HT needs large computing and memory resources. Due to these requirements, a software implementation [17]–[19] based on general purpose CPUs is not suitable for real-time HT applications.

Focusing on the operational process and the voting procedure of HT, many enhancements and optimization algorithms with parallelization were proposed to accelerate the HT towards better suitability for real-time applications [20]. Nakanishi et al. [21] proposed a real-time HT algorithm based on Content Addressable Memory (CAM), which acts as a memory for Hough space. The CAM executes directly the voting process and peak extraction and can perform highly parallel processing tasks for the HT. Another effort of parallel algorithm execution, based on a linear array with reconfigurable pipeline bus system (LARPBS), is proposed by [22], which can complete HT in $O(1)$ time using $mn^2 \cdot (n \times n \text{ pixels image with } m \Delta \theta)$ processors and thus can improve operation speed significantly. To reduce the cost of computational delay for HT, the work in [23] proposes software implementations for parallelization of HT on a graphics processing unit (GPU), which obtain a 7-times faster speed than traditional CPU implementations. However, the problems of accuracy decrease and huge power-consumption are not easily avoided, because most hardware implementations require large numbers of memory and processors.

In particular, due to the limitations of on-board power-up system in a car, the implementations for road-lane detection have strict requirements on accuracy, processing speed and power consumption. Therefore, an optimized design of road-lane detection systems is necessary for real-time applications.

The organization of the paper is as follows: In Sect. 2, the HT algorithm is briefly described and common points of HT criticism are explained. Section 3 proposes a hardware architecture for road-lane detection on the basis of HT implementation with parallelized voting procedure and local-maximum-selection algorithm. In Sect. 4, the performance results of developed prototype systems on the DE1 and the DE4 FPGA platforms are presented. In the finally Sect. 5, we summarize the conclusions of our investigation of road-lane-detection systems.

### 2. Straight-Line Extraction with Hough Transform (HT)

One of the problem facilitations of road-lane detection is that the lanes have to be detected only for a short distance ahead. A relatively simple HT-based algorithm, which does not employ any tracking or image-to-real-world reasoning, can thus solve the road-lane-detection problem in roughly 90% of the highway cases [24].

\[
\rho = x \cdot \cos \theta + y \cdot \sin \theta
\]  

Equation (1) represents all possible straight lines through a Cartesian-space point $(x, y)$ in the form of a parametric equation, while in polar space $(\rho, \theta)$ a sinusoidal curve is represented by Eq. (1). As a consequence, all sinusoidal curves defined by Eq. (1) for the points on a straight line in the Cartesian space have to pass through an identical point in $(\rho, \theta)$ space.

A discretized $(\rho, \theta)$ space, named Hough space, is used by the HT to exploit the above describe properties in combination with a four step algorithm. In the first step the Hough space is initialized with zero values. In the second step the coordinates $(x, y)$ of a still unused edge pixel from the processed image frame are selected. In the third step, which is the main HT step, $\theta$ is changed in discrete increments of $\Delta \theta$ from 0 to $\theta_{\text{max}}$. For each discretized $\theta$, Eq. (1) is then used to calculate the related $\rho$ value, to identify the corresponding Hough-space discretization bin and to increase the vote value for this bin. In step 4, the unprocessed edge pixels are checked. If there are further unprocessed pixels, the algorithm continues with step 2. Otherwise, the detected straight lines are outputted and algorithm goes to step 1 for the next image frame.

The HT is commonly criticized for its burden of the large necessary computational efforts as well as memory requirements. The main factors, which influence the
magnitude of this burden, are the increments $\Delta \rho$ and $\Delta \theta$ for Hough-space discretization because they determine the number of necessary $\rho$ calculations and the size of Hough-space, i.e., the memory requirements. For the coordinates $(x, y)$ of each edge point in an image frame, the $\rho$ values are obtained according to Eq. (1) for $\theta$ value increments from 0 to $\pi$. Besides the effect of $\Delta \rho$ and $\Delta \theta$ parameter choices on computational burden and Hough-space size, these choices also affect the accuracy of straight-line detection. In fact there is a trade-off between the practical requirements of high accuracy on one hand and low computational burden as well as memory resources on the other hand. It should be noted, that in addition to the dependence of the HT-computational cost on the chosen increment $\Delta \theta$ in the parameter space, there is also a proportional dependence on the edge-pixel number ($p$) of the image frame. Therefore, the computational cost shows an $O(p \cdot \frac{\theta_{max}}{\Delta \theta})$ increase. Consequently, even though computational requirements and memory usage can be improved by reducing the Hough-space resolution, this will normally be accompanied by lower accuracy of straight-line detection. In view of the demanding requirements for accuracy and operating speed of the road-lane-detection application, finding an optimized trade-off between computational burden, memory usage and detection accuracy is the inevitable challenge of the reported research.

3. Hardware Architecture for Road-Lane Detection

The complete design of HT can be generally divided into 4 major parts, which are shown in Fig. 1. The preprocessing is used to choose edge pixels by some algorithms of noise reduction and edge detection. The number of edge pixels, for which the HT is carried out, is a major factor which determines the HT’s computational cost. In this design, we apply a median filter (arranging the pixel values from small to large) to eliminate the influence of noise pixels, due to its fewer requirements of logic-elements and only a slight increase of computing burden. The computing unit and voting-procedure unit, which are introduced in next subsection, determine the efficiency of HT. The last part is the lane-marking unit, which is used to highlight the road lanes according to the polar-coordinates of peak-points in Hough space. It’s an inverse of the computing-unit operation.

3.1 Parallelized and Pipelined Computational Operation

A major problem of the computing-process in HT is that $\sin \theta$ and $\cos \theta$ can not be calculated directly in hardware by a simple logical hardware solution. CORDIC (Coordinate Rotational Digital Computer) [25] is an efficient algorithm to calculate $\sin$ and $\cos$ and has previously been used in HT implementations. However, due to the iteration process of CORDIC, it leads to accuracy losses and complexity in computation. To improve the HT-operating speed and guarantee the accuracy, we applied a look-up-table (LUT) solution to replace the time-consuming $\sin \theta$ and $\cos \theta$ computation without any accuracy loss. In each LUT, the $\sin \theta$ or $\cos \theta$ fractional value is scaled by a certain power-of-2 factor and represented in two’s complement notation. The result for $\rho$ according to Eq. (1) is divided by the same factor to obtain the correct computation result.

For hardware implementation of the computing process as illustrated in Eq. (1), a computing unit with $n$ parallel parts and additional pipeline architecture, as shown in the Fig. 2, is applied. Here the hardware parallelism $n$ represents a divisor of the chosen number of discrete $\theta$-values for the angle coordinate of the Hough space. Each parallel part is composed of two multipliers, two LUTs (implemented as RAM) for $\sin \theta$ and $\cos \theta$, one adder, and pipeline registers between the part-internal computing units.
In this design of road-lane detection, the power-of-2 factor for fractional value scaling is set as 8192 (2^{13}), while the computing unit is divided into \( n = 9 \) parallel parts. Both the precision and the operational speed can be kept on a high level by this choice, so that a good trade-off is achieved.

3.2 Voting Process with Parallelized Initialization of Hough-space

Generally, the voting procedure for HT is implemented by increasing the vote value at the corresponding location \((\rho, \theta)\) of the Hough space. To avoid address conflicts due to the \( n \)-fold parallelism applied for this voting procedure, the 2-dimensional Hough space with \((\rho, \theta)\) is mapped onto 1-dimensional memory blocks.

The Hough-space initialization is one of the challenges of the real-time implementation. As the Hough space (implemented by RAM) cannot be initialized by one control signal, e.g. a reset signal. The simplest way is to initialize the Hough space after finishing the voting procedure for the current frame. This means some non-negligible additional time has to be spent for this purpose and to be added to the processing time for an image frame. In this paper, we apply our previously proposed parallelized initialization method\[26\] to solve this problem. A double-clock operation for the write port is utilized for hidden initialization of the Hough space already during the voting procedure.

The parallelized initialization method follows the algorithm shown in Fig. 3.

Additional 1-bit flags are added to each image frame and each voting location in Hough space to avoid conflicts between voting process and initialization. The frame-flag starts with 0 for the first frame, and alternates between 1 and 0 for the following frames, as illustrated in Fig. 3. All vote-flags in Hough space are initialized with 0 for the first frame. Based on different conditions concerning voting address \((VA)\) and initialization address \((IA)\), there are 4 independent ways for dealing with the vote values, which are stored in Hough space (implemented by dual-port memory).

a.) For the \( k \)th frame image, when \( VA_i \) latches to \( address_i \) in Hough space, if the vote flag \((VF)\) belonging to the data stored in \( address_i \) is same as current frame flag \((FF)\), this means that the Hough-space location has already been initialized for the current frame and also that the new vote has to be accumulated at this address. Therefore, only the stored vote value is increased by 1.

b.) When \( VA_i \) latches to \( address_i \) in Hough space, if the \( VF \) is different from the current \( FF \), this means that the stored data is still from the previous frame and that the storage location should be initialized for the current frame. In addition, the new vote has to be accumulated at this address. Consequently, \( VF \) is changed to the current frame flag and the vote value is set to 1.

c.) When \( IA_n \) latches to \( address_n \) in Hough space, if the \( VF \) is same with the current \( FF \), this means that this storage location has already been initialized for the current frame and can therefore be skipped without any action.

d.) When \( IA_n \) latches to \( address_n \) in Hough space, if the \( VF \) is different from current frame flag, this means that the stored data is still from the previous frame and should be initialized for the current frame. Therefore, the vote flag is changed to the current frame flag and the stored data is reset to 0.

3.3 Combined Threshold and Local Maximum Algorithms for Straight-Line Determination

One of the contributions of this proposed architecture is to shorten the processing time to approximately the voting-process time, by executing not only Hough-space initialization but also peak-point extraction in parallel. In the traditional way, the peak-point extraction has to be carried out after the voting process for the image frame, thus leading to an additional time loss. In this work, we combined the local maximum algorithm with the applied threshold-value method, through an updating procedure, storing only the meaningful peak-points in a dual-port memory, which can shorten the processing time.

The threshold-value method is the most popular way to decide on the polar coordinates of straight lines as it does not require much additional computational effort. For the implementation of HT with threshold-value method, the vote value and the corresponding \((\rho, \theta)\) Polar-coordinate pair are outputted, when the vote value at this Hough-space location is larger than the pre-defined threshold value, which is supplied from the voting-procedure module. These polar
Fig. 4 Distribution examples of vote values in the Hough space. (a) and (b) refer to the same images but with different Hough-space resolution. Obviously, the number of polar coordinates around the peak points with vote values above the threshold is much smaller for (b) with lower Hough-space resolution than for (a) where resolution is 4 times higher. Therefore, a smaller window size for reducing the interferential polar coordinates can be used in case of (b) as compared to (a).

Fig. 5 Algorithm of the dynamic combinational local-maximum search for finding the polar coordinates of straight lines.

coordinates are then considered to define a detected straight line. However, due to unavoidable distortions and noise, as e.g. the quantization error caused by resolution parameters $\Delta \rho$ and $\Delta \theta$, the votes are usually distributed over a small range around the polar coordinates of the correct straight line. This effect of localized vote distribution can lead to the generation of multiple peak points which cross the threshold value in the distribution area, as shown in Fig. 4, i.e. to the detection of multiple interfering straight lines around the straight line which is actually present in the image frame.

To decrease the number of interfering noise-like straight lines and to find a meaningful set of distinct peak-points in Hough space, we apply the threshold-value method in combination with a local-maximum algorithm using parameters for defining the neighborhood window. A local maximum, also called relative maximum, is the absolute maximum within this neighborhood. In the traditional way, the neighborhood is usually limited by a Hough-space window of size $A \times A$. The Hough space is divided into many overlapped windows of this size, and then traversed through all of these windows to find the local maximum values one by one. Using the described method has the consequence that the process for finding the polar coordinates of these local maxima, i.e. the reduction of the number of the noise-like straight lines, must be carried out after the voting procedure for the image frame, thus leading to an additional time loss.

In this research, we propose a dynamic combinational method for finding the local-maximum peak points in Hough space, which is parallelized with the voting processing. The method uses another dual-port memory and follows the below described algorithm shown in Fig. 5:

1. For each frame image, wait for an input-vote-value (IVV), larger than the fixed threshold. When an IVV is provided, read the first address of the dual-port RAM, which is used to store potential local-maximum vote
values (PLMVVs) and their Hough-space location, corresponding to existing straight lines. Then go to step 2.

2. If the current PLMVV, stored in the current dual-port-RAM address, is equal to zero or “null”, meaning that no IVV had been stored previously, directly write the IVV (including Hough-space location) into this address, then reset the dual-port memory address to the first address and return to step 1. Otherwise, go to step 3 for comparing the stored PLMVV with the IVV.

3. Judge whether the IVV is within the $A \times A$ range of the current PLMVV.

   A) If the IVV is in the $A \times A$ range of the current PLMVV and greater than this PLMVV, overwrite the current PLMVV with IVV (including Hough-space location). Then reset the address counter to the first dual-port RAM address and go to step 1.

   B) If the IVV is in the $A \times A$ range and smaller than current PLMVV, abandon this IVV, reset the address counter to the first dual-port RAM address and go to step 1.

   C) If the IVV is not in the $A \times A$ range of the current PLMVV, read the next dual-port-RAM address and go to step 2.

4. Continue steps 1 to 3 for all the IVV until the voting process for all edge pixels in the image frame is completed. Finally, output all PLMVVs stored in the dual-port RAM, the Hough-space location of which represents the detected straight lines in the image frame, and initialize the dual-port RAM for the processing of the next image frame.

According to the described algorithm, the polar-coordinate screening module is implemented by one dual-port RAM, an address counter, a state machine, a local-maximum estimation unit and a few of logic elements, as illustrated in Fig. 6. A global control signal “start” enables the state machine and the dual-port RAM. After a potential local-maximum coordinate $ITR$ (input-theta-rho) and the corresponding IVV in Hough-space are inputted to this module, the “Local-max Estimation Unit” triggers one of four states (NULL, GREATER_THAN, LESS_THAN, OUT) in the state machine, based on the comparison between the IVV and PLMVV, according to the four conditions in step 2 and 3 of above algorithm. These states control the write enable and the address counter of the dual-port RAM. Obviously, the IVV should be written into the dual-port RAM when the state of the state machine becomes “NULL” or “GREATER_THAN”. When the current state becomes “OUT”, the address counter is triggered for the comparison with the PLMVV stored at the next dual-port RAM address. The “dual-port RAM” unit with four 28-bit words is utilized for storing the peak voting points in Hough space. The top 10 bit (27-18 bit) are used to store the PLMVV, the next 8 bit (17-10 bit) are used to store the corresponding $\theta$ and the final 10 bit (9-0 bit) are used to store the corresponding $\rho$. As we use a 9-fold parallel structure to implement the voting procedure in the present FPGA design, the Hough space is accordingly divided into 9 parts based on the corresponding range subdivision of $\theta$ (see Fig. 2), applying 9 polar screening modules of the type shown in Fig. 6. Therefore, the total number of detectable straight lines is $36 = (4 \times 9)$ in the current FPGA design, which is enough to deal with road-lane detection in actual applications.

The resolution of the image frame has an important influence on the optimum window size for local-maximum search. As apparent in Fig. 4, with a high image resolution, the number of polar coordinates with vote values above threshold (interferential points) around the peak point with highest vote value will also be high. Higher resolution image frames therefore require larger optimum-window size for reducing the number of interferential, noise-like polar coordinates of detected road lanes.

### 3.4 Prototype FPGA System for Road-Lane Detection Based on HT

Figure 7 shows the designed FPGA-based prototype system for road-lane detection. The system’s D8M-GPIO video camera operates at 60 fps and has VGA-resolution (640x480 pixels). The chosen low-cost DE1-SoC platform board contains a Cyclone V (5CSEMA5F31C6) Altera FPGA device and is used to implement the core circuitry of the system. The result of the road-lane detection is displayed on a MTL2 multi-touch panel.

The core circuitry of the prototype system is implemented of the FPGA and consists of three major modules, which perform the straight-line detection in the inputted video sequence. The implemented modules execute the three steps necessary for HT-based straight-line detection. The preprocessing part for image-edge extraction is carried out by the first module, which applies the Morphology

![Fig. 6](image-url)
operator [27]. The first module’s main objective is to determine the pixels, which constitute the important edges of the image, and to output the edge-pixel coordinates via a FIFO to the second actual-HT module. The number of these edge pixels is a major factor which determines the HT’s computational cost. Precise edge-pixel detection is therefore very important for computational-cost minimization of the system. The input image’s geometrical center is chosen as origin for the Cartesian coordinates of the edge pixels. The FIFO serves as an edge-pixel buffer memory and as serial serial-input source of edge pixels to the second module for HT-execution. The HT-execution module is the most complex core module of this prototype system, which is made up of the parallelized units for polar-coordinate computing, voting procedure, and local-maximum peak-point searching. The finally transformation of the polar coordinates ($\rho, \theta$) of the detected straight lines to the Cartesian-coordinate space is done by the third module for line drawing, including output of the lines to the MTL2 multi-touch panel.

4. Experimental FPGA-Prototype Evaluation

4.1 Performance Analysis

Before the implementation of the road-lane detection on the FPGA board, we firstly have to define the Hough-space discretization parameters ($\Delta \rho$ and $\Delta \theta$) and the window size for local maximum searching. The resolution of these discretization parameters directly affects Hough-space size and corresponding memory consumption. Furthermore, the incremental quantity for $\Delta \theta$ also determines the iteration number for each edge pixel. As evident from Fig. 8, after the edge detection (a2) by the Morphology operator and subsequent binarization (a3), different $\Delta \theta$ values show different results of the straight-line detection, where the threshold is set as ($\frac{\rho_{\text{max}}}{2}$). The results from (b1) to (b3) in Fig. 8 are attained by the threshold-value method for finding the peak Hough-space points, which determine the detected straight lines. It can be seen that larger $\Delta \theta$ ((b2) and (b3)) causes more interfering lines so that the detection results become worse. Whereas, the local maximum algorithm can decrease the number of interfering lines effectively, especially for a larger $\Delta \theta$, as shown in (c2), (c3), (d2), and (d3). In particular, the result with $\Delta \theta = 2^\circ$ and $5 \times 5$ sub-window for local maximum searching is better than that with $\Delta \theta = 1^\circ$ and only thresholding method. Meanwhile, the memory usage in the case of $\Delta \theta = 2^\circ$ is only half compared to the case of $\Delta \theta = 1^\circ$. In general, a larger sub-window size for the local maximum algorithm is more effective in the case of larger $\Delta \theta$. However, there is also a tradeoff with detection resolution, because the resolution of detected straight lines becomes worse. Eventually, we have used an incremental quantity $\Delta \theta = 2^\circ$ and a sub-window size $5 \times 5$ as the optimized choice under these trade-off conditions.

As mentioned previously, in this developed prototype system for road-lane detection, the parallelized Hough space for vote accumulation is divided into 9 parts. To increase the arithmetic speed, the incremental angle quantity $\Delta \theta$ is chosen to be 2 degrees. In addition, the coordinate origin of the input image is set at the geometrical image center. Therefore, the range of $\rho$ in the case of a video sequence with 640x480 pixels per frame is $400 = \sqrt{\frac{640^2+480^2}{2}}$, and the range of $\theta$ is defined from 0° to 360°. Furthermore, non-negative $\rho$ values are obtained for the complete $\theta$ range. As a result, the total requirement for Hough-space storage becomes $792000 = (400 \cdot \frac{360}{2})$ bits, i.e. a number of $400 \cdot \frac{360}{2}$ Hough-space addresses where vote values with up to 11-bit word length can be stored.

The number of edge pixels determines the
computation amount of HT. How to reduce the redundant edge pixels is one of the challenges for the preprocessing of HT. The traditional way is to select a region of interest (ROI) from the original image by a suitable image-processing algorithm [9]–[12]. Generally, the number of edge pixels is a small proportion of whole image pixels. In the case of road-lane detection, this percentage can be expected to be even smaller than in other types of input images. We have extracted 3800 image frames from a highway video with 1024x768 pixel resolution, to analyze the percentage distribution of the obtained edge pixels. The determined proportion of edge-pixels is in the range between 4.5% and 8.5%, while the average is around 6%. Therefore, we decided to apply only a median filter to reduce the appearance of noisy edge pixels, instead of selecting a ROI, because the ROI selection will also increase the computing burden.

The hardware resource usages on the DE1-SoC board with an Altera Cyclone V FPGA device are listed in Table 1. The embedded system utilizes 17% of the available logic and 49% of the available memory for the whole design. The parallel voting and initialization procedure based on a doubled SRAM clock cycle utilizes 2 PLLs (phase-locked loops). In particular, the parallel computing unit consumes 18 (2 \times 9) multipliers in the HT module. Due to these system decisions, our developed low-cost prototype system achieves an average processing speed of 135 frames/s for HT-based road-lane detection at 50 MHz working frequency, which is clearly sufficient for real-time application.

### Table 1

| Resource                  | HT (Used/Available/Utilization) | Total (contains interface) (Used/Available/Utilization) |
|---------------------------|---------------------------------|--------------------------------------------------------|
| Total block memory bits   | 1875/4065 (46%)                 | 1985/4065 (49%)                                       |
| (Kbit)                    |                                 |                                                        |
| Total PLDs                | 0/6 (0%)                        | 2/6 (33%)                                             |
| Total registers           | 2210/166036 (=<2%)              | 5781/166036 (3%)                                     |
| Logic utilization         | 2889/32070 (9%)                 | 5460/32070 (17%)                                    |
| Embedded 18x18 multipliers| 18/224 (8%)                     | 18/224 (8%)                                           |

4.2 Accuracy Analysis

The main influence factors for the accuracy of the road-lane detection are interfering straight-lines around the real lines (boundaries of the road lanes) during the detection operation. The approach of combining the threshold method for straight-line determination with the developed local maximum algorithm can remove these unwanted noisy lines effectively. The threshold value is defined by the maximum possible vote value \( \rho_{\text{max}} \) and a fraction factor \( p \), as illustrated in Eq. (2). The detection results with different threshold values, calculated by different fraction factors \( p \), are shown in Fig. 9. When \( p \) is smaller than one half, there are many noisy lines in the detection results, while for a fraction factor \( p \) larger than one half, some of the real straight-line information is lost during the detection process. Therefore, the optimum \( p \) is about one half and accordingly the threshold for 640x480-pixel images becomes \( 200 = \frac{1}{2} \cdot \sqrt{640^2 + 480^2} \).

\[
\text{threshold} = \rho_{\text{max}} \cdot p = \frac{\sqrt{x^2 + y^2}}{2}
\]

(2)

Based on the above analysis results, the efficiency of the local maximum algorithm in terms of noise-line (i.e. false-positive line) reduction is investigated in Fig. 10 for different threshold settings and window sizes of the local maximum algorithm. The upper histogram shows the results for the number of incorrectly detected (false-positive) straight lines. The local maximum algorithm can reduce the false-positive line number efficiently, but after the detection threshold becomes larger than 50% of the maximum vote value \( \rho_{\text{max}} \), the effect of the local maximum algorithm becomes smaller and the false-positive line reduction is only about 20%, as can be seen in the lower curve, mainly also because the number of false positive lines is already substantially reduced. This finding also indirectly proves
Another point worth noting is that a 7x7 window has a similar effect on straight-line false-positive reduction as a 5x5 window, in other words, continued increase of the window size does not lead to a better straight-line detection result, which once again verifies that the optimized window size for the local maximum algorithm is 5x5 pixels. The accuracy of road-lane detection due to the application of the local maximum algorithm is verified to show an average improvement of 41%, compared with a threshold only implementation, according to the comparison results in Fig. 10.

In addition, the detection performance is also evaluated by a Receiver Operating Characteristic curve (ROC curve) in Fig. 11, which illustrates the accuracy performance of the hardware architecture for road-lane detection via True Positive Rate (TPR) and False Positive Rate (FPR) by varying the threshold for binarization during preprocessing. Consistent with previous analysis, the local maximum algorithm is verified to reduce the FPR for the same TPR. This results in a detection accuracy (the area under the ROC curve) for the case “threshold with 5x5 window” which is much higher than the accuracy for the case “threshold only”. Therefore, the local maximum algorithm can not only reduce the noisy straight lines, but also can significantly improve the accuracy for road-lane detection, i.e., the detection of the two straight lines which mark the two sides of the road lane. The accuracy of the case “threshold with 5x5 window” is about 94%, while about 89% are achieved for the case “threshold only”.

Comparing with other object detection solutions (pedestrian detection, gesture recognition or face recognition), a learning process is not necessary for lane detection based on HT. However, the threshold of HT becomes one of the decisive factors. That’s because the features (edge, texture, gradient or the combination of these features) of general object-detection solutions based on feature extraction are irregular or abstract, so that a learning process is required in these cases to increase the accuracy of the detection. On the other hand, for lane detection based on HT, the feature required is only a regular straight-line. Thus, combined with the robustness and high-accuracy of HT, the detection rate can be maintained at a very high level, under an optimum threshold for HT. Nevertheless, in the real-time detection, the high accuracy may be decreased by the influence of brightness in the real-world environment. But this problem can also occur in other real-time object-detection applications.

### 4.3 Comparison of Hardware Resource Usage

The hardware resource usage is compared to previous state-of-art works [12], [28], [29], which are implemented on comparable FPGA platforms.

A modified Canny-Hough road-lane detection algorithm is proposed in [12] for achieving real-time implementation. In order to relax the complexity of the algorithm, the authors simplified both the gray-scale conversion and the Canny-edge detection, separately. For the graying, they used just one-third of the sum of RGB, instead of multiple floating-point operations required in the exact graying equation. For the edge detection, on the other hand, the authors introduced three stages (Gaussian smoothing, Sobel edge tracking and sharpening) to simplify the Canny-edge detection. A super-resolution reconstruction algorithm is additionally applied for selecting a ROI to reduce the edge-pixel number and the highly time-consuming complexity of the HT. In this way, the presented FPGA-based system in [12] achieves a processing speed of 24 frames/s, as listed in Table 2. A particular aspect to note is that the authors of [12] have to use an external SDRAM to store the converted gray-scale image and the corresponding edge-detection result, so that their design requires a very large memory usage, which is not mentioned as a negative aspect of the work in [12].

In [28], a robust road-lane detection and tracking system is presented, which features two main parts. One part contains a Sobel filter plus HT for road-lane detection,
and the other part applies a Kalman filter for dealing with road-lane tracking. The authors of [28] further used the Coordinate Rotation Digital Computer (CORDIC) algorithm to obtain $\sin \theta$ and $\cos \theta$ for the HT implementation. CORDIC is a relatively simple and efficient algorithm to calculate trigonometric functions, which can reduce memory usage when compared with the LUT method. However, besides the dependence on the number of fractional bits, the accuracy losses of CORDIC depend additionally on the iteration number. In both [28] and our work, each image frame is captured by a camera and then stored in SRAM before being sent to the FPGA. However, it is worth noting that although the image resolution is larger in [28] than in our work, the total number of memory bits is smaller than in our work. The reason lies in the employment of LUT memories for HT to avoid CORDIC in our FPGA implementation. Further, the processing speed of [28] with 25 frames/s is more than a factor 5 slower than our achieved prototype-implementation results.

Chen et al. [29] applied a run-length encoding algorithm with a block-based processing element to compute $\rho$ and $\theta$ for all pixels in the block, which results in a reduction of the computing complexity. An edge detection procedure is not included in the hardware-prototype system of [29]. The total-memory usage of [29] is about 1.5 times larger than in our work, because an off-chip memory is used to pre-store the binary-feature image of the edge-detection result. The hardware system of [29] achieves a higher operation speed of 352 frames/s for an image resolution of 512x512 pixels under a higher working frequency of 200 MHz.

For fair comparison, our developed prototype architecture has been also implemented on a higher-performance DE4-230 platform board with a Stratix IV EP4SGX230KF40C2 FPGA-device. The real-time detection results under different conditions are shown in Fig. 12. This implementation has been evaluated under the same working frequency as in [29]. In addition, the operation speed has been transformed to a normalized speed, which illustrates the computational capacity of processed edge pixels in one millisecond. As listed in Table 2, this work is superior to [29] with a two times higher normalized speed under the same working frequency of 200 MHz.

The hardware resource usage and power consumption of each FPGA board for the developed prototype system is shown in Table 3. As mentioned before, the resolution of the captured video for the DE1 FPGA board is 640x480 (VGA-size) pixels at 60 fps. For the DE4 FPGA board, the captured video has 1024x768 pixel resolution (XGA-size) at 30 fps. The higher resolution means that the number of edge pixels detected from the XGA-size video should be more than for the VGA-size video used on the DE1 FPGA board. Therefore, under application of the same algorithm and the same hardware architecture, the utilization of logic

### Table 3 Usage of FPGA-hardware resources in the DE1- and DE4-platform-based prototype systems for road-lane detection.

|                | DE1 | DE4 |
|----------------|-----|-----|
| Logic elements | 5356 | 14597 |
| Registers      | 5700 | 7246 |
| Memory (Kbit)  | 1985 | 2047 |
| DSP block      | 52   | 96   |
| Total PLLs     | 2    | 3    |
| Thermal power (W) | 0.505 | 3.867 |
elements based on the DE4 FPGA board is as expected higher than on the DE1 FPGA board. However, as the other architecture parameters of the two implementations are defined to have the same size, the usage of memory is nearly equal. In particular, the power consumption (thermal power) of the low-cost DE1 implementation is only 0.505 W, which is less than one seventh of the power consumption of the high-performance DE4 implementation.

5. Conclusions

In this paper, we reported a hardware architecture for real-time road-lane detection by Hough transform (HT) in combination with a local-maximum algorithm for finding the maximum vote-value coordinates in localized Hough-space regions and for suppressing neighboring lower noise-like maxima. This enhances the detection accuracy to meet the accuracy requirements of real-time applications for road-lane detection. The prototype design was carried out with both low-cost DE1 and high-performance DE4 FPGA-board platforms, enabling different resolution sizes of the video-based implementations. During real-time verification, both prototypes, based on high-end FPGA board (Stratix IV) as well as low-cost FPGA board (Cyclone II), proved successful to satisfy the requirements of accuracy and power consumption for application in an on-board advanced driver assistant system (ADAS). In particular the DE1 FPGA-board implementation with lower cost and power consumption, while maintaining real-time performance and detection accuracy, is a practically attractive solution.

References

[1] H. Jung, J. Min, and J. Kim, “An efficient lane detection algorithm for lane departure detection,” IEEE Intelligent Vehicles Symposium (IV), pp.976–981, 2013.
[2] T.M. Hoang, H.G. Hong, H. Vokhidov, and K. Park, “Road lane detection by discriminating dashed and solid road lanes using a visible light camera sensor,” Sensors, vol.16, no.8, p.1313, 2016.
[3] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, “A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios,” IEEE Trans. Veh. Technol., vol.63, no.2, pp.540–555, 2014.
[4] H. Yoo, U. Yang, and K. Sohn, “Gradient-enhancement conversion for illumination-robust lane detection,” IEEE Trans. Intell. Transp. Syst., vol.14, no.3, pp.1083–1094, 2013.
[5] K.-Y. Chiu and S.-F. Lin, “Lane detection using color-based segmentation,” IEEE Proc. Intelligent Vehicles Symposium, pp.706–711, 2005.
[6] T.-Y. Sun, S.-J. Tsai, and V. Chan, “HSI color model based lane-marking detection,” IEEE Intelligent Transportation Systems Conference, pp.1168–1172, 2006.
[7] Y. He, H. Wang, and B. Zhang, “Color-based road detection in urban traffic scenes,” IEEE Trans. Intell. Transp. Syst., vol.5, no.4, pp.309–318, 2004.
[8] M.D.E. Munajat, D.H. Widiantoro, and R. Munir, “Road detection system based on RGB histogram filtering and boundary classifier,” IEEE International Conference on Advanced Computer Science and Information Systems (ICACSI), pp.195–200, 2015.
[9] X. Li, Q. Wu, Y. Kou, L. Hou, and H. Yang, “Lane detection based on spiking neural network and Hough transform,” IEEE Image and Signal Processing (CISP) 8th International Congress, pp.626–630, 2015.
[10] Ali, Mohd, A. Radzi, and H.M. Saad, “A new approach to highway lane detection by using Hough transform technique,” Journal of ICT, vol.16, no.2, pp.244–260, 2017.
[11] D. Ding, C. Lee, and K. Lee, “An adaptive road ROI determination algorithm for lane detection,” TENCON 2013-2013 IEEE Region 10 Conference (31194), pp.1–4, 2013.
[12] S. Hwang and Y. Lee, “FPGA-based real-time lane detection for advanced driver assistance systems,” IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp.218–219, 2016.
[13] Srivastava, Sukriti, M. Lamb, and R. Singal, “Lane detection using median filter wiener filter and integrated Hough transform,” Journal of Automation and Control Engineering, vol.3, no.3, 2015.
[14] Z. Zheng, P. Li, M. Hu, W. Zhang, and Y. Li, “Drivers’ lane-changing maneuvers detection in highway,” Springer Singapore Man-Machine-Environment System Engineering, pp.21–29, 2016.
[15] Y. Kortli, M. Marzougui, B. Bouallegue, J.S.C. Boste, P. Rodrigues, and M. Atri, “A novel illumination-invariant lane detection system,” IEEE International Conference on Anti-Cyber Crimes (ICACC), pp.166–171, 2017.
[16] W. Rong, Z. Li, W. Zhang, and L. Sun, “An improved CANNY edge detection algorithm,” IEEE International Conference on Mechatronics and Automation (ICMA), pp.577–582, 2014.
[17] Z. Xu, B.-S. Shin, and R. Klette, “Accurate and robust line segment extraction using minimum entropy with Hough transform,” IEEE Trans. Image Process., vol.24, no.3, pp.813–822, 2015.
[18] Y. Li, L. Chen, H. Huang, X. Li, W. Xu, L. Zheng, and J. Huang, “Nighttime lane markings recognition based on Camy detection and Hough transform,” IEEE International Conference on Real-time Computing and Robotics (RCAR), pp.411–415, 2016.
[19] Cho, J. Hyun, Y.M. Jang, and S.B. Cho, “Lane recognition algorithm using the Hough transform with applied accumulator cells in multiple-channel ROI,” The 18th IEEE International Symposium on Consumer Electronics (ISCE), 2014.
[20] P. Mukhopadhyay and B.B. Chaudhuri, “A survey of Hough transform,” Pattern Recognition, vol.48, no.3, pp.993–1010, 2015.
[21] M. Nakanishi and T. Ogura, “Real-time CAM-based Hough transform algorithm and its performance evaluation,” Machine Vision and Applications, vol.12, no.2, pp.59–68, 2000.
[22] L. Chen, H. Chen, Y. Pan, and Y. Chen, “A fast efficient parallel Hough transform algorithm on LARPBS,” The Journal of Supercomputing, vol.29, no.2, pp.185–195, 2004.
[23] G.-J. Braak, C. Nugteren, B. Mesman, and H. Corporaal, “Fast Hough transform on GPUs: exploration of algorithm trade-offs,” Springer International Conference on Advanced Concepts for Intelligent Vision Systems, pp.611–622, 2011.
[24] A. Borkar, M. Hayes, M.T. Smith, and S. Pankanti, “A layered approach to robust lane detection at night,” IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems, pp.51–57, 2009.
[25] J. Bruguera, N. Guil, T. Lang, J. Villalba, and E. Zapata, “CORDIC based parallel/pipelined architecture for the Hough transform,” VLSI Signal Process, vol.12, pp.207–221, 1996.
[26] J. Guan, F. An, X. Zhang, L. Chen, and H.J. Mattausch, “Real-time straight-line detection for XGA-size videos by Hough transform with parallelized voting procedures,” Sensors, vol.17, no.2, pp.270–283, 2017.
[27] Z. Yu, G. Wei, C. Zhen, et al., “Medical images edge detection based on mathematical morphology,” IEEE-EMBS Annual International Conference of the Engineering in Medicine and Biology Society, pp.6492–6495, 2006.
[28] I.E. Hajjouji, A.E. Mourabit, Z. Asrih, S. Mars, and B. Bernoussi, “FPGA based real-time lane detection and tracking implementation,” IEEE International Conference on Electrical and Information Technologies (ICEIT), pp.186–190, 2016.
[29] Z.-H. Chen, A.W.Y. Su, and M.-T. Sun, “Resource-efficient FPGA
architecture and implementation of Hough transform,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.20, no.8, pp.1419–1428, 2012.

Jungang Guan received the M.S. degree in engineering from Hiroshima University, Hiroshima, Japan, in 2014, and Ph.D. degree in 2018. Now he is a researcher at HiSIM research center in Hiroshima University. His research interests include embedded systems, computer vision and image processing for real-time applications.

Fengwei An received the B.S. from Qingdao University of Science and Technology in 2006, the M.S. and Ph.D. from Hiroshima University, Japan, in March 2010 and March 2013 respectively. He has been an Assistant Professor at Hiroshima University in the graduate school of engineering since 2013 to 2017. His research interests include energy-efficient image recognition algorithms and low-power circuits for embedded systems. Dr. An is a member of IEEE.

XiangYu Zhang received the B.S. from Tianjin University, China, in 2013 and M.S. from Hiroshima University, Japan, in March 2016. Now she is a doctor student of the TAOYAKA Program at Hiroshima University. Her main research is on the hardware development for energy-efficient image recognition algorithms.

Lei Chen received the B.S. and the M.S. from Qingdao University of Science and Technology in 2006 and 2009, respectively, the Ph.D. from Hiroshima University, Higashi-Hiroshima, Japan, in 2012. She pursued her post-doctoral research at the HiSIM Research Center, Hiroshima University from October 2012 to March 2017. Her research interests include circuit design based on organic thin-film transistors and hardware algorithm design for image processing.

Hans Jürgen Mattausch received the doctor degree from the University of Stuttgart, Stuttgart, Germany, in 1981. From 1982 to 1996 he was with Siemens AG in Munich, Germany, where he was involved in the development of CMOS technology, memory and telecommunication circuits, power semiconductor devices, chip-card ICs and compact models. Since 1996 he is a Professor with Hiroshima University researching in the fields of VLSI design, nano-electronics and compact modeling. Dr. Mattausch is a senior member of IEEE and a member of IEICE.