A New Automatic Real-Time Crop Row Recognition Based on SoC-FPGA

SHAN LI, ZHIBIN ZHANG, FANG DU, AND YUCHEN HE

School of Computer Science, Inner Mongolia University, Hohhot 010021, China
Key Laboratory of Wireless Networks and Mobile Computing, School of Computer Science, Inner Mongolia University, Hohhot 010021, China
Corresponding author: Zhibin Zhang (cszhibin@imu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 31760345.

ABSTRACT

With the development of artificial intelligence technology, agricultural robot plays a significantly important role for agricultural intelligence. Crop row line detection is a critical and fundamental step for agricultural robot navigation. Although there are some crop row lines detection methods, few of them can meet the real-time requirement for agricultural robot under complex fields conditions. In view of this, a real-time crop detection system implemented on a SoC FPGA (System-on-a-Chip Field Programmable Gate Array) is first proposed in this paper, which contains crop segmentation and crop row detection, where we design parallel pipeline architecture to enhance real-time performance by using line buffer and sliding windows technologies. At the same time, the fixed point representation is used to reduce the memory resource in this system. The proposed system is evaluated and implemented on Xilinx Zynq UltraScale+ MPSoC ZCU102 SoC-FPGA. The experimental results show that the proposed system can process an image with 1920 × 1080 resolution only within 210 ms with the average accuracy of 89.7%, which satisfies the real-time requirements of the crop rows recognition.

INDEX TERMS

Crop row recognition, real-time image processing, SoC FPGA.

I. INTRODUCTION

Traditional agriculture needs more manpower and material resources. With the developments of electronics and computer technology, autonomous implementation of various agricultural tasks has become possible under field conditions [1]. Especially, the price of high-resolution camera falls, making the machine vision technologies be used widely in various field operations, including agricultural robot guidance in relatively complex field environments. The autonomous agricultural robot based on machine vision guidance can improve efficiency and quality of agricultural production, and has become one of the hot topics in agricultural engineering field [2].

However, the field environment is complicated against crop row recognition; precise control of robot navigating in real-time still has been a bottleneck problem. Accurate detection of crop rows is one of principal problems to be solved, and large number of scholars has achieved a lot in this aspect [3]. However, existing crop row recognition methods are caught in a dilemma between time consumption and recognition accuracy. Therefore, real-time field crop row recognition is necessary for accurately controlling the robot to move in field environments without crushing the crop.

Besides, for reducing time consumption, some of existing algorithms use small-resolution images, or split large-resolution image into small regions of interest area before detecting image crop row line [4], [5]. This mainly results in the image information not to be fully utilized, and the accuracy of crop row recognition cannot be guaranteed.

Nowadays, FPGA has a good development prospect in the image processing, and many image processing algorithms have been proposed based on FPGA architecture [6]–[9]. Some researchers [10] also developed embedded graphics processing unit (GPU) for precision weed management. But in field robot vision guidance, few literatures have been reported about FPGA. Aiming at the above problems, in this paper, we propose a method for automatically detecting the crop rows lines in field image, where the parallel and pipeline architecture is designed and implemented on the SoC FPGA in order to meet the real-time navigation requirements of agriculture robot. It is based on the crop row structure initially proposed by Zhao and Zhang [11], and improves crop...
rows recognition, being about 49 times faster than a personal computer.

The rest of this paper is organized as follows: in section II, we introduce the background and related work of the crop row detection method. Section III describes the modified crop row detection method. Section IV describes our proposed FPGA-based hardware design in detail. Later, in section V, we analyze and discuss the experimental results. The last part is the conclusion in section VI.

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION TO THE CROP ROW MODEL

A common assumption about crop row structure is that crop rows can be represented by parallel straight lines, the green lines $P_v$, as shown in Fig. 1(a) [4], where we define a crop row reference frame of the camera buffer. The origin of the reference frame is lens center (the optic axis of lens coincides with the z-axis of camera coordinate); the z-axis of the reference frame is parallel to the optic axis of lens; its x-axis and y-axis are parallel and vertical to the corresponding coordinate axes of the camera, respectively. A set of parallel row lines is projected on an image plane, where the row lines intersect at a vanishing point, shown as in Fig. 1(b). Then, we assume a sequence of equidistant intersections ($M_1, M_2, \ldots, M_8$) between $P_v$ and the straight line $BL$, which are located on a line parallel to the x-axis direction of the camera system, shown as Fig. 1(a). Therefore, they definitely project onto the corresponding rows in a crop image. At the same time, since these points have same z-coordinates and equal spaced in the x-direction, their projections in image are also equally spaced. That is, if the crop rows $P_v$ in Fig. 1(a) are parallel to the y-axis of the camera’s coordinate system, the projections in image $P_v$ are shown in Fig. 1(b) where their projections theoretically are equidistant in image, denoted as $d_{(v,v-1)}$. In the Fig. 1(b), the red dotted line passing through the center of the image is called as the base line ($BL$) [11], which will be discussed in detail in section III; the subscript $v$ is the $v$-th crop row of the image; $d_{(v,v-1)}$ is the distance between the two adjacent crop rows, denoted as the $v$-th and the $(v-1)$-th rows, and $c_v$ is the distance between $v$-th crop row and the image center.

B. RELATED WORKS

Owing to unevenness of crop growth and out-of-flatness soil background, along with lighting variations, the outdoor field environments are more complicated than indoor scenes. However, crop row pattern is still stable. Therefore, there have been various techniques proposed for detecting crop row in field images by using pattern recognition. For instance, one of early typical methods for crop row detection was using the Hough transform [12]. In [13], the authors proposed an improved Hough transform, called random Hough transform (RHT). It was tested in the case of sparse and intensive plant distributions and its accuracy is overtly higher than that of the Hough transform. A crop row line detection method was proposed, using the index (COM) [14] to segment crop image and a template to match row lines, and then, the Theil-Sen method was employed to correct the crop lines, with a high accuracy of 93.1% [15]. The method in [16] based on Theil-Sen and least squares with the accuracy of 88.3%. The algorithms in [4] was an effective crop row line detection, where the dynamic programming techniques was used, combining the geometry and the prior knowledge of crop row image to detect curved crop line. The work reported in [5] introduced a detection method based on the statistics properties of green crop pixel point distribution, which can detect both straight and curved crop row lines. The approach proposed in [17] used a crop row line detection algorithm based on region of interest (ROI), where the multiple ROIs of crop row image were constructed, and then, the linear regression was used to fit crop row lines. The final detection rate reaches 93% in the case of high weed rate or low vegetation cover. There have been also some scholars to use machine learning method to detect crop row line. The work in [18]
utilized the image sequences and labelled pixels of the images into crop and weeds, so that the proposed classification system is robust to the substantial environmental changes with respect to weed pressure and weed types, growth stages of the crop, and soil conditions. The method in [19] proposed a crop row centerline detection method based on machine vision and genetic algorithm, which is robust to different crops and different growth stages. The solution proposed in [20] used an automatic method for detecting straight or curved crop rows lines of potatoes based on micro-ROI concept, achieving acceptable results in terms of accuracy and processing time.

In summary, researchers have studied the field-based crop row line detection algorithms by using vision technology, and each is relatively effective to obtain crop row lines under special field conditions. However, most of existing algorithms are implemented in ROI of image or the offline status, and the time consumption of crop row line detection with high-resolution image is none the less a bottleneck problem, especially under complex field conditions. At present, no scholar has combined FPGA with image processing methods in the research field of agricultural robot vision guidance. Therefore, this paper proposes an FPGA-based real-time crop row detection algorithm, focusing on modifying the existing algorithm and designing an adaptive parallel processing and pipeline FPGA system, and will improve the processing rate of crop row line detection in order to meet the real-time requirement of the agricultural robot navigation system.

III. APPROACH

A. OVERVIEW OF ALGORITHM

The method proposed in this paper based on the field crop row model, combines the crop row detection with characteristics of the FPGA architecture. This section mainly introduces the specific details of the implementation of the crop row detection method. The specific steps of implementing crop row detection on Soc FPGA by using parallel pipeline architecture are described in section IV.

Crop row detection consists of two main linked phases: image segmentation and crop rows detection. Fig. 2 shows the whole scheme proposed according to the literature [5]. The modules designed in this paper are conveniently integrated into the FPGA architecture. Additionally, some prior knowledge about crop rows structure is applied to search for crop rows lines. Meanwhile, we assume that the intrinsic and extrinsic parameters of the cameras are known to the visual system of agricultural robot.

B. CROP ROW SEGMENTATION

Crop segmentation aims to emphasize information on green channels in the original crop images. Common methods include ExG (Excess Green index) [21], ExGR (Excess Green minus Excess Red) [22], VEG (Vegetative index) [23], CIVE (Color Index of Vegetation Extraction) [24], NDI (Normalized Difference Index) [25] and COM (Combined index) [14], etc. Since the robustness and good illumination, the ExG index is chosen to calculate the greenness of green crop from the crop row images in RGB space, shown in Fig. 3(c) and Fig. 3(d), and the corresponding original images are shown as in Fig. 3(a) and Fig. 3(b), where the age of the maize is about 2 weeks; the age of the strawberry is about 4 weeks. Our group data set is obtained through using two types of cameras (One is MV-VS030FC camera made in Shanxi Microvision, for instance, Fig. 3(a-b) grasped; the other is Balser A301fc camera made in Germany, for instance, Fig. 16(c-d) grasped), which have the same resolution of 640 × 480 and lens focal length of 12 mm.

Threshold method is available for the binarization of the corresponding greenness images of green crop images. This paper adopts the largest inter-class variance method (Otsu) [26], which is particularly suitable for complex field crop images. We obtained the binary image of green crop images based on the Otsu’s method, shown in Fig. 4(a) and Fig. 4(b) where the pixels having greater values than the
threshold given by the Otsu are labeled as black, i.e. these pixels contain a higher greenness degree.

**(C. CROP ROW DETECTION)**

1) CROP ROW SPACE

When a field crop row image is processed through the above steps, its original image is converted into a corresponding binary image. The target object points \(P\) can be easily expressed as

\[
P = \{p_i, i = 1, 2, 3, \ldots , n, n \in N\} \tag{1}
\]

where \(p_i\) stands for \(i\)-th target point in binary image: \(N\) is a set of natural number.

Referring to the literature [11], we define a base line (called \(BL\), a red dotted line as shown in Fig. 1(b), which is passing through a multi-rows image paralleling to the x-axis of image coordinate. For a certain pixel \(O(x, y)\) belonging to \(BL\), we calculate the density of an arbitrary line \(L^o_\theta\), intersecting at this point with the angle \(\theta\) with \(BL\), according to Eq. (2).

\[
d^o_\theta = \frac{\sum p_i}{\left( \sum_{j=1}^{N_j} v_j + \sum_{i=1}^{N_i} p_i \right)} \tag{2}
\]

where \(p_i\) is a target point, \(p_i \in L^o_\theta\), with the total amount \(N_i\); \(v_j\) is the \(j\)-th background point, \(v_j \in L^o_\theta\), with the total amount \(N_j\). If the angle \(\theta\) varies from 45 to 135 degree, the density space is formed about pixel \(O(x, y)\), expressed as Eq. (3).

\[
S_o = \{d^o_\theta : \theta \in [45, 135], o \in BL\} \tag{3}
\]

where \(S_o\) is a density space of the lines passing through the point \(O\). In this way, when the point \(O\) is traversing \(BL\) in a multi-rows image, the whole density space based on \(BL\) is formed, denoted as \(S_{BL} (S_o \in S_{BL})\). Meanwhile, the \(S_{BL}\) contains the potential row lines densities. When we choose the local maximum values of \(S_{i,\text{max}}(i \in \text{BL})\), these local maximum values obtained in \(S_{BL}\), form a certain pattern, as shown in Fig. 5.

![Binary image and filtering results of Fig. 3.](image)

![Density space of crop rows image.](image)

From Fig. 5, we obviously find that the peaks locate where the rows appear; the intervals between the peaks in the images can reflect the number of the crop rows [11].

2) SIMPLE CROP ROW DENSITY MODEL

Generally, the crop rows of binary image hold some intrinsic properties. For instance, the distance \(d_{\text{int}}\) between the consecutive peaks in \(S_{BL}\) in a multi-rows image can be defined a feature parameter, which corresponds to a constant, called as inter-row space. For eliminating the disturbances of row boundaries, the row pattern like Fig. 5 can be abstracted into a simple model, shown in Fig. 6, where some definitions are as following: 1) the local maximum values stand for the corresponding crop rows, denoted as the sign \(r\); 2) the local minimum values stand for the soil background, denoted as the sign \(g\). Thus, there exit some values in \(S_{BL}\), denoted as the sign \(e\), which are larger than the average value \(Z_{\text{avg}}\), but are less than the local maximum values. They can be considered as the edges of crop rows. On the other hand, for a multi-rows image, there always is a global maximum \(Z_{\text{max}}\),

![Simple crop row density model.](image)
representing the first crop row, which shows good robustness in the recognition of crop rows.

According to prior knowledge of multi-rows structure, if there are two neighboring rows which are detected by using pattern recognition method, the guidance parameters for controlling agricultural robot can be extracted automatically [11]. Therefore, this paper divides the crop row detection algorithm into three modules in FPGA architecture: density calculation, the first-crop row recognition and the second-crop row recognition, as shown in Fig. 7, where the inputs of algorithm are the first-crop row recognition and the second-crop row recognition, and the outputs are the position and angle of crop rows. Both the first and second crop row detection module can produce the position and angle of crop rows for robot navigation. The crop row detection module includes two sub-modules: centerline detection and boundary detection.

Suppose a pixel point $O(x, y)$ on the $BL$ at angle $\theta$, it can form a line $L_o^o$, as shown in Eq. (4). The line $L_o^o$ intersects the image edge at the points $(x_0, y_0)$ and $(x_1, y_1)$ corresponding to its upper edge and lower edge, respectively. For simplifying the computation, we used the temporary parameters $temp_0$ and $temp_1$ to convert intersection computation into if-else method, which are expressed as Eq. (5-9).

\[ temp_0 = \frac{y}{\tan \theta + x} \quad (4) \]

\[ y_0 = \begin{cases} y - x \cdot \tan \theta & \text{if } (temp_0 < 1 \text{ and } temp_0 > width) \\ 1 & \text{else} \end{cases} \quad (5) \]

\[ x_0 = \begin{cases} width & \text{if } (temp_0 < 1) \\ temp_0 & \text{else} \end{cases} \quad (6) \]

\[ temp_1 = \frac{\text{height} - y}{\tan \theta + x} \quad (7) \]

\[ y_1 = \begin{cases} y - x \cdot \tan \theta & \text{if } (temp_1 < 1 \text{ and } temp_1 > height) \\ 1 & \text{else} \end{cases} \quad (8) \]

\[ x_1 = \begin{cases} 1 & \text{if } (temp_1 < 1) \\ \text{height} & \text{else if } (temp_1 > \text{height}) \\ temp_1 & \text{else} \end{cases} \quad (9) \]

where the sign $\text{width}$ and $\text{height}$ represent the length and the width of image, respectively; and the parameter $\theta$ ranges from 45 to 135 degree, which is the angle of $L_o^o$ with $x$-axis passing through the current pixel point $(x, y)$. By counting the number of non-zero value of pixel point $p_i$ (i.e., number of green crop pixels) and total number of pixels $N_i$ on the line, we can obtain the density of $L_o^o$ according to Eq. (2).

Thus, the maximum density can easily obtain, denoted as $d_o^o_{\text{max}}$, which corresponds to the first-row. And its boundary lines can also be obtained in order to calculate the row spacing. Since the boundary lines nearly are parallel to its center line of crop row, shown as Fig. 1, we empirically take the angle values, ranging from $-10$ to $10$ degree in the experiment, and the experimental result also validates them, as shown in Fig. 8. According to Fig. 6 the smallest densities are greater than the average density, which are corresponding to the right and the left boundaries of the crop row, denoted respectively as the sign $\text{rightBL}$ and $\text{leftBL}$. Moreover, the distance between the right boundary and left boundaries of crop row is called the row spacing $L_o$, which can be calculated by using $L_o = \text{rightBL} - \text{leftBL}$. In addition, $\text{rightBL}$ and $\text{leftBL}$ can be detected by the crop row recognition method (see Algorithm 1 in Appendix).

\[ \text{Distribution of angles between the left boundary line and centerline.} \]

\[ \text{Distribution of angles between the right boundary line and centerline.} \]

**FIGURE 8.** Distribution of the angles of crop rows in the images sets.

Fig. 8(a) shows the changes of the angles between the left boundary lines and the corresponding centerline, ranging from $-5$ to $5$ degree. Fig. 8(b) shows the changes of the angles between the right boundary lines and the corresponding centerlines ranging from $-10$ to $10$ degree. Therefore, in order to accurately acquire the boundary lines of the crop rows, the constraint of the angle parameter $\theta$ is limited to the range: $(-10, 10)$ degree in the experiments.

Then we can search the second crop row line with its boundary lines in crop row images by using the same pattern.
method according to Eq. (10) where \( x \) is starting point position in \( BL \); \( x_{\text{boundary}} \) is boundary point position in \( BL \); \( L_w \) is the row spacing; \( width \) is the length of image. Meanwhile, \( BL \) intersects the right boundary line right\(_{BL} \) or the left boundary line left\(_{BL} \) at \( x_{\text{boundary}} \).

\[
x = \begin{cases} 
  x_{\text{boundary}} + \frac{L_w}{2} & \text{if } x_{\text{boundary}} \geq \frac{width}{2} \\
  x_{\text{boundary}} - \frac{L_w}{2} & \text{else if } x_{\text{boundary}} < \frac{width}{2}
\end{cases}
\] (10)

IV. ALGORITHM IMPLEMENTATION BASED ON FPGA

A. FRAMEWORK BASED ON FPGA

The block diagram of the developed SoC FPGA circuits is shown in Fig. 9 where the architecture from the abstract level to the interface layer adapts to the modified algorithm of crop rows detection in this paper. Reversely, the modified algorithm module can be logically embedded in the SoC-FPGA system, where the processing system (PS) transfers the image data streams to the programmable logic (PL) part, and then, they are processed there by the modified algorithm.

Fig. 10 illustrates the diagram of the abstract-level of the modified algorithm where the image data stream is captured from a camera, and then transferred to the logic circuit; the processed results from the crop row detection module are passed to the HDMI interface. Especially, the parallel processing and the pipeline structure are designed to improve the efficiency of the whole system. To this end, it consists of different implement modules marked by different colors shown in Fig. 10, such as the blue image pre-processing, the yellow crop row segmentation, the light blue image pre-processing and the brown position of crop row. Additionally, FG/BG mask and crop row mask can mark the crop rows recognized from the foreground (FG) and the background (BG) to display in the monitor through the interface HDMI transmission.

The main modules with their interconnections between the modules are integrated into system architecture in this paper to meet the requirement of field robot navigation, shown in Fig. 7. At the same time, each module corresponds to the basic step of the algorithm introduced in section III. The specific functioning modules are mainly including:

1) IMAGE PREPROCESSING MODULE

It mainly finishes the green crops segmentation from the soil backgrounds in images, in which the greenness calculation combined with the Ostu’s thresholds can obtain good segmenting results. At the same time, the median filtering method is employed to eliminate the isolated or small patch noises in the backgrounds for the convenience of the following steps of crop rows detection.

2) IMAGE STORAGE MODULE

To meet the further processing, the binary image streams are stored into a block random access memory (BRAM). We use 1-bit to represent a pixel rather than 8-bits in order to reduce resource consumption and save time. Moreover, the detailed method is described in section IV.C.
3) CROP ROW DETECTION MODULE
It consists of three parts including the density calculation, the first crop row detection and second crop row recognition. These three parts correspond to three different processes, respectively. Among them, the density calculation module is a crucial step of the whole algorithm. On obtaining the density of each potential row line, we can use the global maximum value to determine the first crop row position (described in section III.B), then we can recognize another crop row line described in the section III.C. For this goal, we design a line buffer and a sliding window to realize the pipeline structure strategies of data flow processing among the modules and the parallel calculation to improve the running time of the whole algorithm.

In order to design a real-time and memory-efficient crop row recognition system based on FPGA architecture, some important issues are considered as follows:

4) DATA PATH AND INTERFACE
For an external port, an efficient solution is to use a dedicated system platform port. We use a USB interface to read the data stream and an HDMI interface to visualize them in the experiment. For an internal port, an AXI interface can be used. The SoC-FPGA system in this work is interfaced with an AMBA AXI version 4 to interconnect PS and PL, whose AXI interface consists of four 64/32-bit high-performance AXI (HP AXI) slave interfaces, two 32-bit AXI master interfaces, two 32-bit AXI slave interfaces, a 64-bit AXI accelerator coherency port (ACP) and an extended multiplexed I/O (EMIO) interface. These interfaces ensure the data stream traffic connections between the designed modules.

5) PARALLEL AND PIPELINE ARCHITECTURE
In order to improve the efficiency of our algorithm, the FPGA-based parallel and pipeline architecture is considered as one of the good choices because it allows the multiple different images data as inputs to be processed simultaneously in a certain clock cycle. Especially, we propose a parallel recursive scheme, which simultaneously calculates the densities of the line segments passing through a current pixel point on the BL at different angles to save the processing rate of the algorithm. Then, we use the parallel architecture of multiple pipelines to determine the locations of crop row lines in each pipeline.

6) MEMORY CONSUMPTION
One of our goals is to improve the real-time performance of the whole crop row recognition, which is implemented by buffering image data in BRAM in the FPGA. For example, for a buffering BRAM module of 32.1Mb size of the Xilinx FPGA device, when storing a 32-bit full-HD size image, 15.82Mb space is needed. In order to reduce the image storage resources, this paper proposes a data cutting method to convert 8-bit data into 1-bit. Thus, the image data become one-dimension format, replacing the entire image with serial density values that represent different row line segments with 90 different angles passing through each pixel point on the BL. This schedule ensures that the resource occupancy rate can be reduced effectively.

B. DATA PATH AND INTERFACE
In this study, for constructing data stream traffic, we use two HP AXI interfaces to connect the processor to the vision kernel module, video display module and video capture module, respectively. Furthermore, the memory-mapped communication is set. Meanwhile, for controlling each block, the AXI master interface is also connected to the vision kernel and the processor.

The development platform (Zynq UltraScale + MPSoC ZCU102 Evaluation Board [28]) is used, which contains an ARM processor, FPGA and multiple common interfaces, such as USB and SD memory readers. To read and write back the image stream which can be further processed by vision kernel, a dedicated video DMA (called VDMA) is connected to the DDR3 register. It operates in circular mode and automatically restarts when a new frame enters.

The ACP is the interface between the AXI4-Stream interconnect and the host processor (ARM CPU). The ACP is a 64-bit AXI slave interface on the snoop control unit (SCU), which provides an asynchronous cache-coherent access directly from the PL to the host processor subsystem. This means that there is a low-latency path between the PS and the accelerator embedded in the PL. The host processor needs only to initialize the cache, after that, the input stream (RGB image data) is transferred in a completely independent manner to the VDMA. As a result, the more cycles remain available on the host processor for higher-level processing.

AMBA AXI version 4 is used internally to interconnect the processor cores with the FPGA. The platform is also equipped with two off-chip memory interfaces, DDR3 and LPDDR2. The ARM processor reads a raw data stream from the SD card or the USB interface, and the data stream is made accessible to the PL for further processing. The RGB and depth data are stored separately in two blocks of the VDMA as the crop row segmentation and detection kernel. After processing, the outputting result is passed to the HDMI, a displaying module, shown in Fig. 11. The clock module in our architecture outputs a 100 MHz clock, which can produce 120 MHz and 148 MHz signals for communication and computation, respectively. For controlling all signals, an AXI-Lite interface is used to configure and initialize all the modules and the sub-modules in the PL.

C. PARALLEL AND PIPELINE ARCHITECTURE
In the revised algorithm (described in the section III), the majority of computational load is responsible for the recognition crop row. Since each pixel participates in the recognition process, resulting in needless repeating computation, the latency in the steps is inevitable.
By identifying an unrolled loop, we can improve the utilization of the massive computational resources of the FPGA (see Image Pre-processing method in Appendix Algorithm 2).

For conveniently processing, it is feasible to transform the RGB image into gray image according to the greyness index value calculated based on Eq.(11) on the FPGA circuit where the shift operation instead of division boosts speeds up the image processing. Our algorithm of the crop row recognition consists of the multiple loops where all variables depend on the loop iterator \(k\) in Image Pre-processing method (see Appendix Algorithm 2). Although the operations are dependent on each other, as a whole, they can be processed in parallel. An appropriate circuit for implementing these methods is illustrated in Fig. 12 and consists of a shift register of \(\text{wh} \times \text{ww} \times 1\)-bit size, which is accessible by each cell in parallel and a RAM block of \(\text{Width} \times (\text{wh} - 1) \times 8\)-bits size. The parameters \(\text{wh}, \text{ww}\) are the size of window of the median filter, their values being 5 and 4 in this work, respectively. The parameter \(\text{width}\) corresponds to the width of the input image frames. By using the designed circuit depicted in Fig. 12, the median value of the pixels of interest region (the center pixel of the window) can be obtained in two clock cycles. And the values of the interest pixels and its neighbors are updated in only one clock cycle.

\[
ExG = [G \gg 1] - R - B \quad (11)
\]

Next, we focus on the binary images obtained in the pre-processing stage to obtain the second-row lines. For each pixel on the \(BL\), the lines that pass through it intersect the image boundaries with the angles ranging from 45 to 135 degree. Thereby, we detect their upper and lower endpoints on the image boundary by parallel calculation. Fig. 13 shows the circuit structure designed for this goal, which determines the upper and lower endpoints by the parallel adders and the comparators, as shown in the light green and light blue portions of Fig. 13. After that, the 1-bit image data stored in the BRAM is read by VDMA, and the green crop points (i.e., black pixel points) and the non-green crop points (i.e., white pixel points) on the line are counted by a simple divider. During the crop row lines detection, the densities of lines passing through the current pixel point in \(BL\) can be obtained in parallel.

![FIGURE 11. Diagram of software module design of frame.](image)

![FIGURE 12. Register Transfer Level (RTL) representation of median filtering module.](image)

![FIGURE 13. Register Transfer Level (RTL) representation of density computation.](image)

![FIGURE 14. Parallel process of row boundary detection.](image)
TABLE 1. Comparisons of all methods on the dataset.

| Algorithms                                         | Implement platform       | Execution time | Accuracy |
|----------------------------------------------------|--------------------------|----------------|----------|
| Template Matching followed by Global Energy Minimization (TMGEM) [4] | Intel i5-3350 3.30 GHz   | 862.2 ms       | 73.7%    |
| Detection by Accumulation of Green Pixels (DAGP) [5]     | Intel i7 2.0 GHz          | 712 ms         | 83.6%    |
| Sector-scan-based crop row detection (SC-CRD) [11]      | Intel i3-2130 3.2 GHz    | 5824 ms        | 94.3%    |
| **Our proposed method**                              | **ZCU102 Evaluation Board** | **52.4 ms**    | **89.7%** |

BRAM in the previous step, where there are multiple density values (typically, 90) per pixel on BL. When the position and the angle of center line of a crop row are confirmed, the angle range of the boundary lines in this step is also determined. This step can be completed in only one clock cycle. Then these density values are sent to the comparator. Similarly, the minimum density at the current angle is obtained in one clock cycle. The read density and density comparison only cost two cycles, which are implemented in pipeline and the density value of the next pixel point can be read without waiting for the density comparison completion.

D. MEMORY CONSUMPTION

The platform that we use in this work sharing memory between the host processor and the accelerator (co-processor) is an L2 cache with low latency. The original image pixel data are converted from 8-bits to 1-bit. Compared with the 8-bit data stored, the image storage occupancy is reduced to one-eighth of the original. At the same time, when calculating the densities of the row lines in images, the original 32-bits float type is converted into a 18-bits fixed point type. By reducing the amount of data, the consumption of the entire system storage resources is reduced, shown in Fig. 15. Fig. 15(a) shows that the accuracy of the density calculation results varies with the data bit widths. Obviously, when the data bit widths are 18-bits, it can achieve the best accuracy. Fig. 15(b) shows that the resource occupancy varies with the data bit widths, displaying that the cost of storage resource in 18-bits is less than in 32-bits. Therefore, choosing 18-bits is an effective way to reduce the memory consumption in our research.

V. RESULTS AND DISCUSSION

In this section, we validate the superiority of our system scheme from two perspectives. Firstly, we evaluate its performance in a relatively simple software environment to compare the proposed algorithm with those in the literature [4]. Secondly, we evaluate the performance of the proposed crop row recognition in terms of memory consumption and speed running on FPGA device.

The proposed crop row recognition is tested on the images with different resolutions, which were captured in Northern China. The simple software environment includes a PC with Intel Core i3-2130 @ 3.2 GHz (2 core) CPU. The algorithms are implemented in C language. In the FPGA environment, we use Xilinx Vivado HLS 2017.4 development tools and Xilinx XCZU9EG−2FFVB1156E MPSoC containing 274,080 FFs, 2,520 DSP48Es, 548,160 lookup tables and 32.1Mb BRAM blocks with 2 speed levels as our target devices.

We also test the designed system on our group dataset and the dataset in the literature [4]. The results over the typical examples of the two datasets are shown in Fig.16 and Fig.17, respectively, where the neighboring two rows lines can be recognized to meet the requirements of navigation information extraction of agricultural robot (the first crop row line is labeled in red and the second crop row line labeled in blue). The precision of crop row detection in our algorithm and other three methods (TMGEM, DAGP, SC-CRD) are shown in Tables 1. The experimental results show that our proposed algorithm can effectively detect the crop rows in the images.
It should be pointed out that the highest accuracy of 94.3% is reached on a PC with floating-point variables, whereas in this work we obtain the accuracy of 89.7% by utilizing fixed-point variables.

From Table 1, it can be seen that our approach outperforms the other crop row detection algorithms in term of execution time over the dataset [4]; it only consumes 52.4 ms per image with the resolution of $640 \times 480$. However, the approach proposed in this paper is unable to accurately detect crop rows with a large of weeds in images, or having low color differences between the green crop and the soil background, as shown in the Fig.18 where the first crop row line labeled in red is wrong recognized, shown in Fig.18(b); the second crop row line labeled in blue is wrong recognized, shown in Fig.18(a-b).

The comparisons of the performance are implemented between the proposed system based on SoC-FPGA and the software-based system, as shown in Table 2. The different threads comparisons are accomplished by using QT Creator 5.6.0 with full-HD image resolution. Obviously, we find that the time consumption declines as the thread number increased, and the decrease in time consumption is significant in our SoC-FPGA-based system with only 210.8 ms for the images with the resolution of $1920 \times 1080$.

Overall, our SoC-FPGA-based implementation achieves a 109.19 times speedup over a software implementation with
FIGURE 18. Wrong recognizing results on the typical examples in the dataset [4].

TABLE 2. Performance analysis: Software-based implementation on Intel CPU (3.40 GHz) and SoC-FPGA. (Unit: ms).

| Image resolution | Crop row segmentation | | |
|------------------|---------------------|-------------|-------------|
|                  | Single-thread | 16th-thread | Our method |
| 640×480          | 6990.7        | 3410.6      | 52.4        |
| 1280×720         | 14370.8       | 7030.9      | 121.3       |
| 1920×1080        | 22930.5       | 10370.8     | 210.8       |

TABLE 3. Resource consumption with image resolution of 1920 × 1080 for our approach.

| Resource | Image pre-processing | Crop row detection | Total(%) |
|----------|----------------------|-------------------|----------|
| FFs      | 3,042                | 12,305            | 5.60%    |
| LUTs     | 6,604                | 25,981            | 5.94%    |
| BRAMs    | 4                    | 156               | 8.77%    |

Based on the internal blocks of the device, the resource utilization is mainly caused by LUTs, block RAMs (BRAMs) and Flip-flop (FFs). Our proposed approach consists of two computational parts: the image pre-processing and the crop row detection. Particularly, we choose the images with high resolution of 1920 × 1080, the resource utilization of them is evaluated, as shown in Table 3, and the total resource utilization of the system on the Xilinx XCZU9EG SoC-FPGA is 5.60% for the LUTs, 5.94% for the FFs, and 8.77% for the BRAMs. Undoubtedly, an image with the high resolution of 1920 × 1080 also can be processed in parallel by our approach. Table 4 illustrates the resource utilization of our approach in three different image resolutions. The experimental results demonstrate that the resource utilization of BRAM goes up with the increase of image size; the highest utilization of BRAMs is 160 for the images with resolution of 1920 × 1080, which is two times that of the images with resolution of 1280 × 720 and three times that of the images with single-thread, and a 49.38 times speedup over a software implementation with 16-threads for the images with the resolution of 1920 × 1080.

TABLE 4. Resource utilization with different image resolutions in our SoC-FPGA-based system.

| Resources | FFs | LUTs | BRAMs |
|-----------|-----|------|-------|
|           | Avil | Util. | Avil  | Util. | Avil  | Util. |
| 640×480   | 15,131 | 5.52% | 32,457 | 5.92% | 54    | 2.96% |
| 1280×720  | 15,271 | 5.57% | 32,513 | 5.93% | 114   | 6.25% |
| 1920×1080 | 15,347 | 5.60% | 32,585 | 5.94% | 160   | 8.77% |

Algorithm 1 Crop Row Detection

Function GetCenterLine (Img)
For k from 1 to width do
For angle from 45 to 90 do
    greenNum ← GetGreenPixelNum(Img, angle, k)
    entireNum ← GetEntirePixelNum(Img, angle, k)
    density ← greenNum / entireNum;
    centerLine, centerAngle ← max(density, angle, k)
End for
End for
Return centerLine, centerAngle
End function

Function GetBoundaryLine (Img)
For k from centerLine to width do
For angle from centerAngle-10 to centerAngle+10 do
    greenNum ← GetGreenPixelNum(Img, angle, k)
    entireNum ← GetEntirePixelNum(Img, angle, k)
    density ← greenNum / entireNum;
    rightLine, rightAngle ← min(density, angle, k) & density<average(density)
End for
End for
For k from centerLine to 1 do
For angle from centerAngle-10 to centerAngle+10 do
    greenNum ← GetGreenPixelNum(Img, angle, k)
    entireNum ← GetEntirePixelNum(Img, angle, k)
    density ← greenNum / entireNum;
    leftLine, leftAngle ← min(density, angle, k) & density<average(density)
End for
End for
Return rightLine, rightAngle, leftLine, leftAngle
End function
Algorithm 2 Image Pre-Processing

Firstly, we use iterator \( k \) to iterate the whole image. During the median filter method, we use the neighbors’ value to calculate the center value. Where \((x, y)\) is the position of current pixel in image, the corresponding position of the pixel in the filter window is \( x_{\text{window}}, y_{\text{window}} \), and which values are obtained by the shift function.

\[
\text{Function MedianFilterMethod (Img)}
\]

For \( k \) from 1 to \( \text{height} \times \text{width} \) do

\[
x_{\text{window}} \leftarrow x + \text{windowsize}_x/2; \\
y_{\text{window}} \leftarrow y + \text{windowsize}_y/2; \\
\text{Img}_{\text{neighbor}} \leftarrow \text{shift (Img, } x_{\text{window}}, y_{\text{window}}) \\
\text{CenterValue} \leftarrow \text{Median (Img}_{\text{neighbor}})
\]

End for

Return \text{CenterValue}

End function

Function Image PreProcessing(Img)

\[
\text{GreenInforcement(Img)} \\
\text{OstuThreshold(Img)} \\
\text{MedianFilterMethod(Img)}
\]

Return \text{Img}

End Function

Algorithm of crop row recognition can be implemented on SoC FPGA to overcome the problems unsolved in this paper.

APPENDIX

See Algorithms 1 and 2.

ACKNOWLEDGMENT

(Shan Li and Zhibin Zhang are co-first authors.)

REFERENCES

[1] S. Yaghoubi and N. Akbarzadeh, “Autonomous robots for agricultural tasks and farm assignment and future trends in agro robots,” \emph{Int. J. Mech. Machtron. Eng.}, vol. 13, no. 3, pp. 1–6, 2013.
[2] T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten, “Autonomous navigation using a robot platform in a sugar beet field,” \emph{Biosyst. Eng.}, vol. 109, no. 4, pp. 357–368, Aug. 2011.
[3] H. Mousazadeh, “A technical review on navigation systems of agricultural autonomous off-road vehicles,” \emph{J. Terramech.}, vol. 50, no. 3, pp. 211–232, Jun. 2013.
[4] I. Vidović, R. Cupec, and Ž. Hocenski, “Crop row detection by global energy minimization,” \emph{Pattern Recognit.}, vol. 55, pp. 68–86, Jul. 2016.
[5] I. García-Santillán, J. M. Guerrero, M. Montalvo, and G. Pajarès, “Curved and straight crop row detection by accumulation of green pixels from images in maize fields,” \emph{Precis. Agricult.}, vol. 19, no. 1, pp. 18–41, Feb. 2018.
[6] F. Aydin, H. F. Uğurdağ, V. F. Levent, A. E. Guzel, N. F. R. Annafianto, M. A. Orkan, T. Akgun, and C. Erbas, “Rapid design of real-time image fusion on FPGA using HLS and other techniques,” in \emph{Proc. IEEE/ACSSC 15th Int. Conf. Comput. Syst. Appl. (AICCSA)}, Aqaba, Jordan, Oct. 2018, pp. 1–6.
[7] J. Lei, Y. Li, D. Zhao, J. Xie, C.-I. Chang, L. Wu, X. Li, J. Zhang, and W. Li, “A deep pipelined implementation of hyperspectral target detection algorithm on FPGA using HLS,” \emph{Remote Sens.}, vol. 10, no. 4, p. 516, Mar. 2018, doi: 10.3390/rs10040516.
[8] F. D. Robinson, L. H. Crockett, W. H. Nailon, and R. W. Stewart, “High-level synthesis for medical image processing on systems on chip: A case study,” in \emph{Proc. 26th Int. Conf. Field Program. Logic Appl. (FPL)}, Lausanne, Switzerland, Aug. 2016, pp. 1–2.
[9] C. Li, Y. Bi, Y. Benezech, D. Ginhac, and F. Yang, “High-level synthesis for FPGA: Code optimization strategies for real-time image processing,” \emph{J. Real-Time Image Process.}, vol. 14, no. 3, pp. 701–712, Mar. 2018.
[10] V. Partel, S. Charan Kakarla, and Y. Ampatzidis, “Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence,” \emph{Comput. Electron. Agricul.}, vol. 157, pp. 339–350, Feb. 2019.
[11] S. Zhao and Z. Zhang, “A new recognition of crop row based on its structural parameter model,” \emph{IFAC-PapersOnLine}, vol. 49, no. 16, pp. 431–438, 2016.
[12] V. Hough and C. Paul, “Method and means for recognizing complex patterns,” U.S. Patent 3 069 654, Dec. 18, 1962.
[13] R. Ji and L. Qi, “Crop-row detection algorithm based on random Hough transformation,” \emph{Math. Comput. Model.}, vol. 54, nos. 3–4, pp. 1016–1020, Aug. 2011.
[14] J. M. Guerrero, M. Guijarro, M. Montalvo, J. Romeo, L. Emni, A. Ribeiro, and G. Pajarès, “Automatic expert system based on images for accuracy crop row detection in maize fields,” \emph{Expert Syst. Appl.}, vol. 40, no. 2, pp. 656–664, Feb. 2013.
[15] J. M. Guerrero, G. Pajarès, M. Montalvo, J. Romeo, and M. Guijarro, “Support vector machines for crop/weeds identification in maize fields,” \emph{Expert Syst. Appl.}, vol. 39, no. 12, pp. 11149–11155, Sep. 2012.
[16] J. M. Guerrero, J. R. Ruiz, and G. Pajarès, “Crop rows and weeds detection in maize fields applying a computer vision system based on geometry,” \emph{Comput. Electron. Agricul.}, vol. 142, pp. 461–472, Nov. 2017.
[17] G. Jiang, X. Wang, Z. Wang, and H. Liu, “Wheat rows detection at the early growth stage based on Hough transform and vanishing point,” \emph{Comput. Electron. Agricul.}, vol. 123, pp. 211–223, Apr. 2016.
[18] P. Lottes, J. Behley, A. Milio, and C. Stachniss, “Fully convolutional networks with sequential information for robust crop and weed detection in precision farming,” \emph{IEEE Robot. Autom. Lett.}, vol. 3, no. 4, pp. 2870–2877, Oct. 2018.
[19] Q. Meng, X. Hao, Y. Zhang, and G. Yang, “Guidance line identification for agricultural mobile robot based on machine vision,” in Proc. IEEE 3rd Adv. Inf. Technol., Electron. Automat. Control Conf. (IAEAC), Chongqing, China, Oct. 2018, pp. 1887–1893.

[20] J. García-Santillán, D. Peluffo-Ordoñez, V. Caranqui, M. Pusdá, F. Garrido, and P. Granda, “Computer vision-based method for automatic detection of crop rows in potato fields,” in Proc. IEEE 3rd Adv. Inf. Technol., Electron. Autom. Control (ICITS), vol. 2018, pp. 355–366.

[21] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen, “Plant species identification, size, and enumeration using machine vision techniques on near-binary images,” in Proc. Int. Soc. Opt. Eng. (SPIE), Boston, MA, USA, 1993, pp. 208–219.

[22] J. C. Neto, “A combined statistical-soft computing approach for classification and mapping weed species in minimum tillage systems,” Nebraska Univ., Lincoln, NE, USA, 2004.

[23] T. Kataoka, T. Kaneko, H. Okamoto, and S. Hata, “Crop growth estimation system using machine vision,” in Proc. IEEE/ASME Int. Conf. Adv. Intell. Mech. (AIM), Kobe, Japan, Jul. 2003, pp. 1079–1083.

[24] T. Hague, N. D. Tillett, and H. Wheeler, “Automated crop and weed monitoring in widely spaced cereals,” Precis. Agricult., vol. 7, no. 1, pp. 21–32, Mar. 2006.

[25] M. Guijarro, G. Pajares, I. Riomoros, P. J. Herrera, X. P. Burgos-Artizzu, and A. Ribeiro, “Automatic segmentation of relevant textures in agricultural images,” Comput. Electron. Agricult., vol. 75, no. 1, pp. 75–83, Jan. 2011.

[26] D. Kaur and Y. Kaur, “Various image segmentation techniques: A review,” Int. J. Comput. Sci. Mobile Comput., vol. 3, no. 5, pp. 809–814, 2014.

[27] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, Digital Image Processing Using MATLAB, 2nd ed. NJ, USA: Gatesmark Publishing, 2009.

[28] Xilinx. UG1182-ZCU102 Board User Guide (Version 1.6). Accessed: Feb. 21, 2020. [Online]. Available: https://www.xilinx.com/support/documentation/boards_and_kits/zcu102/ug1182-zcu102-eval-bd.pdf

SHAN LI received the B.S. degree in computer science and technology from the Inner Mongolia University of Technology, Huhhot, China, in 2015, and the M.S. degree in computer science and technology from Inner Mongolia University, Huhhot, in 2019. Her current research interests include computer vision and embedded systems based on FPGAs, SoC design, and computer architecture.

ZHIBIN ZHANG is currently an Associate Professor and a Master’s Supervisor. His research interests include computer vision, embedded systems, and smart agriculture.

FANG DU received the B.S. degree in network engineering from the Inner Mongolia University of Technology, Huhhot, China, in 2019. She is currently pursuing the master’s degree in computer science and technology with Inner Mongolia University. Her current research interests include computer vision and three-dimensional reconstruction.

SHAN LI received the B.S. degree in computer science and technology from the Inner Mongolia University of Technology, Huhhot, China, in 2015, and the M.S. degree in computer science and technology from Inner Mongolia University, Huhhot, in 2019. Her current research interests include computer vision and embedded systems based on FPGAs, SoC design, and computer architecture.

YUCHEN HE received the B.S. degree in computer science and technology from the Inner Mongolia University of Technology, Huhhot, China, in 2019. She is currently pursuing the M.S. degree in computer science and technology with Inner Mongolia University, Huhhot. Her current research interests include computer vision and robot navigation.

* * *