Optik 126 (2015) 1545–1550

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

Real-time edge detection and range finding using FPGAs

Tariq M. Khan[a,*], D.G. Bailey[b], Mohammad A.U. Khan[c], Yinan Kong[a]

[a] Department of Engineering, Macquarie University, Sydney, Australia
[b] Massey University, Private Bag 11-222, Palmerston North, New Zealand
[c] Department of Electrical and Computer Engineering, Effat University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 23 January 2014
Accepted 26 January 2015

Keywords:
Embedded image processing
VLSI
FPGAs
Image normalization
Edge detection

ABSTRACT

The objective of this paper is to design a model which can integrate a range sensor with real time image processing. An FPGA provides physical solutions for most image processing problems and offers a new hardware acceleration opportunity. In this paper we implemented image normalization along with edge detection on real time images and then integrated the range sensor. The model is validated using Spartan-3 FPGA.

1. Introduction

Real-time image processing has been used for variety of applications, e.g., video surveillance systems, authentication systems, and traffic surveillance systems. A very high computation power is typically required to perform these operations [9]. For such applications, normally application specific hardware implementation is required which offers much greater speed than software implementation [10]. Due to advances in VLSI technology, hardware implementation has become an attractive solution for real-time applications [2,6].

To design hardware using VLSI, two types of technologies are available:

- Full custom hardware design also called Application Specific Integrated Circuits (ASICs)
- Semi custom hardware design, these are programable devices like Field Programable Gate Arrays (FPGAs)

Full custom ASIC design offers high performance, but the associated design cost and complexity is very high. On the other hand, semi custom hardware designs, like FPGAs, are ideal in many embedded system applications [6], as they have several desirable features like low power consumption, a large number of I/O ports, small size, and a large number of computational logic blocks. The use of FPGAs has increased as an implementation platform for image processing applications, particularly real-time image/video processing, as they have grown in size and functionality [12].

FPGA programming is significantly different from conventual single-processor programming. In FPGA-based designs, one needs to not only design the algorithm but also the architecture on which it is implemented. FPGA-based designs generally contain a large number of simple processors for parallel processing. Traditionally, FPGAs have been configured by hardware engineers using a Hardware Description Language (HDL). There are two principal languages in use, Verilog HDL (Verilog) and Very High Speed Integrated Circuits (VHSIC) HDL (VHDL), which allows design at various levels of abstraction.

In real-time image processing, finding the accurate distance between objects and the camera has many applications. For example it can be used for real time face recognition where we can define a region in which a face can accurately be recognized, specifically it can be used for iris recognition where an accurate image can only be scanned if the eye location from camera is in the defined region. Given the importance of range finding in real-time image/video processing and their significance in embedded systems, this work presents real-time edge detection and range sensing of objects. To get better results for edge detection, image normalization is done before edge detection. This is done because of the fact that the input image obtained from the sensor may have imperfections or poor quality due to non-uniformity [8]. In the literature, different techniques have been used to cope with this problem. These techniques are broadly categorized into two types: global normalization

* Corresponding author. Tel.: +61 420300045.
E-mail addresses: tariq045@gmail.com (T.M. Khan), d.g.bailey@massey.ac.nz (D.G. Bailey), yinan.kong@mq.edu.au (Y. Kong).

http://dx.doi.org/10.1016/j.ijleo.2015.01.024
0030-4026/

methods, and local normalization methods. Global normalization may contain gamma intensity correction, histogram equalization, histogram matching and normal distribution. The local normalization methods are: local histogram equalization, local histogram matching, and local normal distribution. Global methods, while usually efficient to implement in hardware, often have difficulty with variation in illumination or contrast across the image. Local normalization techniques are quite efficient under different scenarios but their hardware implementation is expensive. In order to cope with this problem, in this paper, a new local normalization technique is presented which is well suited for hardware implementation. For edge detection, the Sobel edge detector is used as its horizontal and vertical filters are separable and can be decomposed, which makes its hardware implementation efficient. The overall system is composed of the following blocks:

1 Image normalization: This module is used to normalize the input image stream. The input images are obtained from sensors may have imperfections due to non-uniformity. If we reduce these imperfections before edge detection we can get a more detailed gradient image than without image normalization.
2 Sobel Edge Detector: This module detects the edges of the video by convolving a sliding window using the Sobel operator to get an edge detected image.
3 Range sensor: The purpose of this module is to send a trigger out to the SRF05-ultra-sonic ranger and read the echo from it. The measure of the echo would give us the distance of the object detected by the sensor.
4 Number displayer: The distance in cm measured by the SRF05-ultra-sonic ranger must be converted to BCD. This would make it easy to show it on screen as well as on the FPGA. A module segment display is thus created within the range sensor to show the value of the converted number on the FPGA board.

The rest of the paper is organized as follows. Section 2 details the proposed method for image normalization. In Section 3, hardware structure for overall system is described. Experimental results are presented in Section 4. Section 5 is about the concluding remarks.

2. Image normalization

The input images which are obtained from sensors may have imperfections or poor quality due to non-uniformity. To cope with this problem, a local normalization algorithm based on the local property of the given image is proposed. The block diagram of the proposed algorithm is shown in Fig. 1.

A Gaussian filter of size $6 \times \sigma$ is used, where the chosen $\sigma = 4.2$ corresponds with the width of the dominating structure present in the input image. The application of the filter results in blending the dominating structure with the background and provides a blurred image that contains the slowly varying illumination pattern. The filtered image is then subtracted from input image to result in a constant-background image. The background subtraction process can also be thought of as a high-pass filter which allows the structure of interest to pass. Though, the filter provides us with uniform background image, its contrast can vary significantly throughout the image. Therefore, contrast enhancement has to be performed next to boost object intensities in relation to the background. For this, we save the sign of the pixels to be used at the later stages. The magnitude obtained by applying absolute operator on the image pixels will be used to estimate the local contrast. The power-law-transformation with $\gamma = 0.75$ is employed here to compress the high contrast pixels relative to those with low contrast. Then another Gaussian filter, with $\sigma = 4.2$, is applied on the power-law-transformed image to average that locally. The resultant image is clipped in between 50 and 255 to avoid over enhancing noise and to retain the relative strength of already high contrast regions. This provides a measure of the local contrast within the image. We recall the absolute-operated image and divide it by this local contrast. The result of this division is multiplied with the sign image and a factor of 150. A constant 128 is added to enable negative values to be elevated for accommodation in the dynamic range of the monitor. In the end, the image is clipped to the allowed pixel range 0–255 to come up with the output normalized image. Fig. 2 depicts the results of proposed normalization on a real-time image.

Fig. 2. (a)Input image and (b) normalized image.

3. Proposed hardware

The aim of this work is to integrate a range sensor with real time image processing application. For this purpose a new method for image normalization is proposed which is efficiently implemented on hardware and then the Sobel edge detector is used for the gradient image. Fig. 3 show the block diagram of the proposed scheme. In this paper, only three main blocks are explained as the detail of remaining blocks is already given by Ref. [2].

3.1. Image normalization hardware structure

In the image normalization block, the 2D Gaussian filter is implemented as a cascade of one dimensional Gaussian filters (1×25 and 25×1). The details of Gaussian filter implementation are given in next subsection. For power law transformation, a look up table is created. The remaining operations for image normalization are

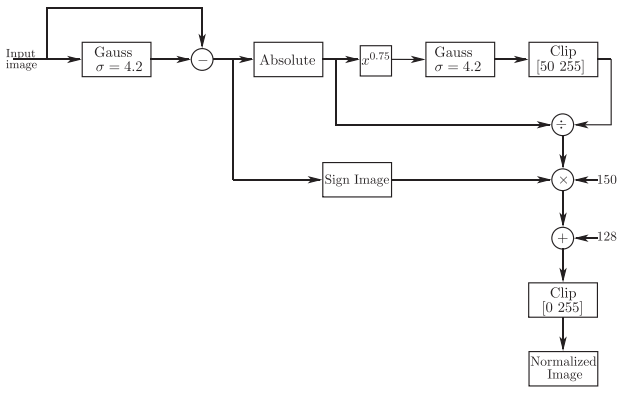


Fig. 1. Block diagram of proposed local normalization algorithm.

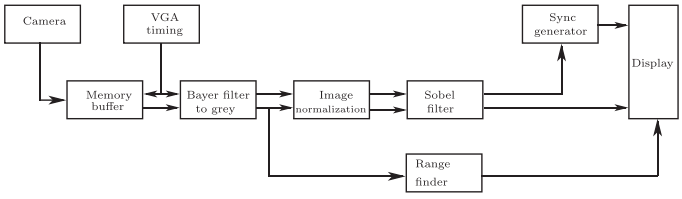


Fig. 3. Block diagram of the proposed hardware.

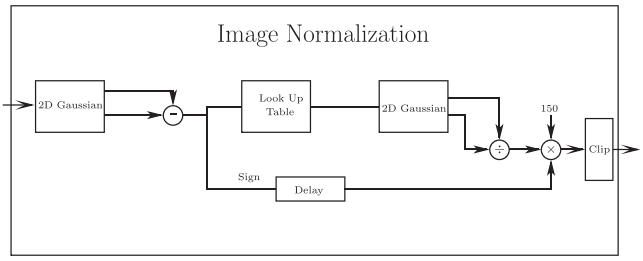


Fig. 4. Block diagram of proposed local normalization algorithm for hardware implementation.

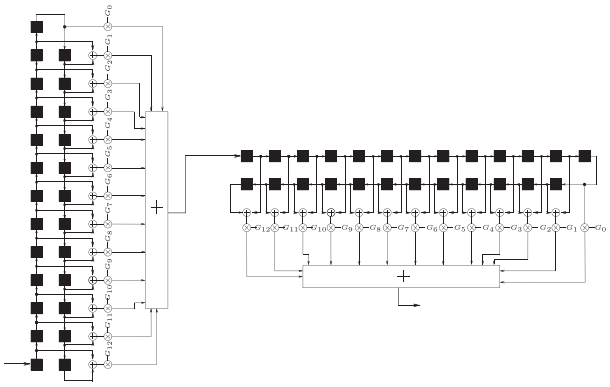


Fig. 5. Hardware implementation of a (1×25 and 25×1) Gaussian filter with $\sigma = 4.2$. The G_x are filter coefficients. For the vertical filter, the boxes represent row buffers.

quite simple. Fig. 4 shows the block diagram of the proposed local normalization algorithm hardware.

3.1.1. Gaussian filter implementation

A Gaussian filter is used for image blurring and removing noise or high frequency components of the image. In two dimensions, the Gaussian function is:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (1)$$

For large σ, the size of filter increases significantly which makes the hardware implementation too expensive. For hardware implementation, a size of $\sigma = 4.2$ is used which gives a 25×25 mask. As the Gaussian is separable, this allows the 25×25 filter to be implemented as cascade of one dimensional Gaussian filters (1×25 and 25×1). The implementation of this cascade filter is shown in Fig. 6. For large windows, several decompositions can be used, for example [7] approximates a large circularly symmetric filter by octagons. Fig. 5 shows the implementation of (1×25 and 25×1) Gaussian filter with $\sigma = 4.2$. Although the filter can be decomposed to only use adders [2], the need of such decomposition is less important on modern FPGAs where high speed pipelined multipliers are plentiful. The optimized hardware multipliers are hard to out-perform with relatively slow adder logic of the FPGA fabric [13].

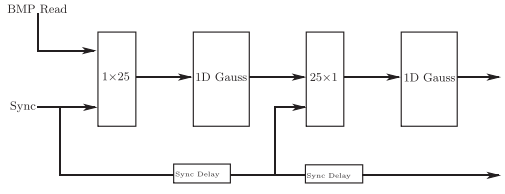


Fig. 6. Block diagram of Gaussian filter implementation.

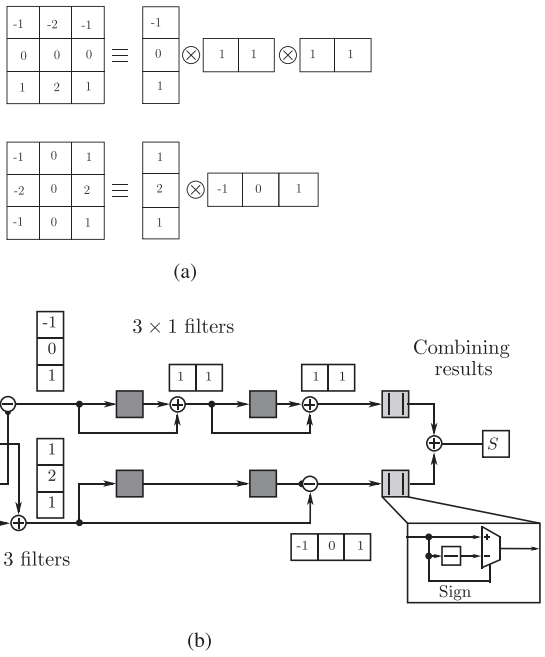


Fig. 7. Sobel filter implementation: (a) filter decomposition and (b) implementation using Eq. (4)

3.2. Edge detection

The edges of an image are considered to be the most important image attributes that provide valuable information for human image perception [5,3]. Image edge detection is a process of locating the edges of objects within an image. In edge detection, the approximate gradient magnitude at each point of an input image is determined. The problem of getting an appropriate magnitude for edges lies in the method used. In the Sobel operator, a 2-D spatial gradient measurement on images is performed. A pair of 3×3 convolution masks is used, one estimating gradient in the horizontal direction and the other estimating gradient in vertical direction. Let H be the horizontal gradient and V be the vertical gradient. The magnitude of the two-dimensional gradient is ideally given by:

$$S = \sqrt{H^2 + V^2} \quad (2)$$

If only edge strength is required then Eq. (2) is quite expensive. For this purpose, two simpler alternatives are commonly used [1]:

$$S = \max(|H|, |V|) \quad (3)$$

or

$$S = |H| + |V| \quad (4)$$

One example of a Sobel filter implementation is given by Ref. [4]; which directly implements the two linear filters and combines the result. However, a simpler implementation can be used to reduce the number of calculations by exploiting separability. Both the horizontal and vertical filters are separable and may be decomposed as shown in Fig. 7(b). The two filters are combined using Eq. (4) to reduce the complexity. Fig. 7(a) shows the hardware structure implementation using Eq. (4). As the orientation is also required in our application, for this purpose a CORDIC unit using the vectoring mode [2] calculates both the arctangent and Eq. (2). Fig. 8 shows the block diagram of the unrolled CORDIC iteration.

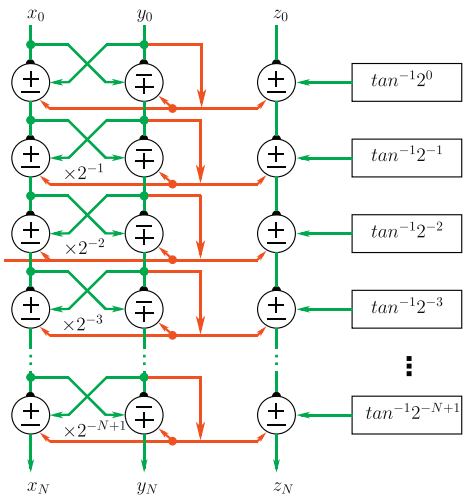


Fig. 8. Block diagram of the unrolled CORDIC iteration.

3.3. Range sensing

For sensing the range of objects SRF05-ultra-sonic ranger is used. It has two modes. Mode1 uses echo and trigger on separate pins. Mode2 uses echo and trigger on the same pin. We use mode1. A trigger of atleast 10 μs is provided to the ultra-sonic ranger. This activates the sensor and sends an echo. When the echo hits any object it stops. The farthest object that this sensor can detect is approximately 4 m. To get the distance of the object the time for which echo remained high is measured. The schematic diagram of range sensing is shown in Fig. 9.

3.4. Process trigger

A trigger of 15 μs is provided to the ultra-sonic ranger. As the Spartan-3 FPGA development board runs at 50 MHz, the time for each clock cycle is 20 ns. So a counter is created that count 15 μs/20 ns = 750 clock cycles. During this time the trigger remains high. After every 35 ms the trigger is applied. Therefore, for resetting the counter to zero 35 ms/20 ns = 1,750,000 clock cycles are required. So in 1 s approximately 28 triggers are created.

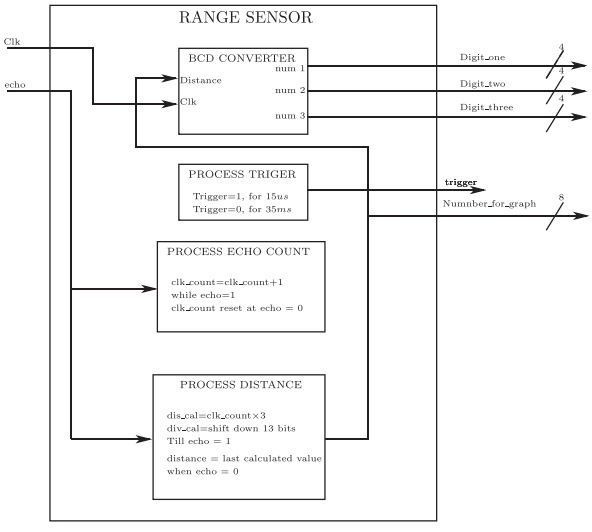


Fig. 9. Range sensor schematic.

3.5. Process echo

In this process the width of the echo pulse is measured and the distance is calculated. A clock counter is made that increments one per clock cycle till echo is high. The clock cycle is divided by 50 to convert the delay to microseconds.'

$$1\ clk\ cycle = \frac{1}{50}(\mu s) \tag{5}$$

so 'x' clk cycles are

$$x\ clk\ cycle = \frac{x}{50}(\mu s) \tag{6}$$

The round trip distance is given by $2d = tc$ where c is the speed of sound in air (343 m/s, or 0.0343 cm/ms). To get the answer in cm the echo pulse width need to be divided by 2/0.0343 = 58. Therefore,

$$x\ clk\ cycle = \frac{x}{2900}(\text{cm}) \tag{7}$$

Division is complex in any hardware. It needs to be avoided because division leads to hardware implementation that is expensive in both hardware resources and system delay. To approximate the division by a power of 2 would inevitably create large calculation errors because 2900 is not close to a power of 2. A much closer approximation (about 5% error) is

$$\frac{3x}{8192} = \frac{3x}{2^{13}} \tag{8}$$

The count of clock cycles the echo is high for is multiplied by 3 and the 13 LSBs are truncated giving 9 bits for the distance. When the echo turns low, the clock counter is stopped and reset to zero. Finally, the last calculation is taken and passed as the signal distance.

3.6. Process BCD

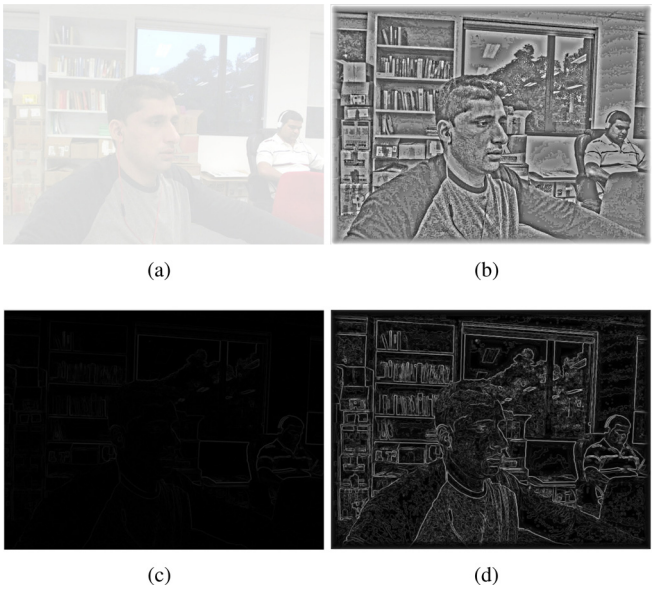
To convert the 9 bit number coming in from the range sensor the "double dabble" algorithm is used [11]. In this algorithm, the 9 bit number is saved in the lower bits of a signal that is 20 bits long. The rest bits are initialized to zero. The signal is left-shifted and iterated for 8 times. On every iteration it checks bits 20–17, 16–13 and 12–9. If any one of these bit combinations is greater than or equal to 5 three is added and the bits are over written. Otherwise all the bits are shifted to left for 8 iterations. At the end of 8 iterations we have 100s in bits 20–17, tens in bits 16–13 and ones in bits 12–9.

4. Results and discussion

In this work the distance of the object from the camera is accurately found and this range finder is successfully integrated with the famous Sobel edge detector. The aim of this work was to integrate the range finder with a real-time image processing algorithm, which can be used for other applications like real-time iris recognition, or face recognition. Fig. 10 shows the comparison of Sobel edge detector with image normalization and without image normalization. It can be seen that if we use image normalization before edge detection we can get more detailed edge image than without image normalization. Figs. 11(a), 12(a) and 13(a) are three examples of poor illumination images. We can see that the gradient images after normalization contain more details than without normalization. Especially, if we compare Fig. 13(c) with (d) we can see some of the text information is not visible in Fig. 13(c), because of poor illumination, but if we use normalization along with gradient we can recover some information, as shown in Fig. 13(d). Fig. 14 shows the integration of range sensor with the edge detector. In Fig. 14, the digital number represents the distance of object from the range sensor and graph at the bottom is the histogram of the image.

(a) (b)

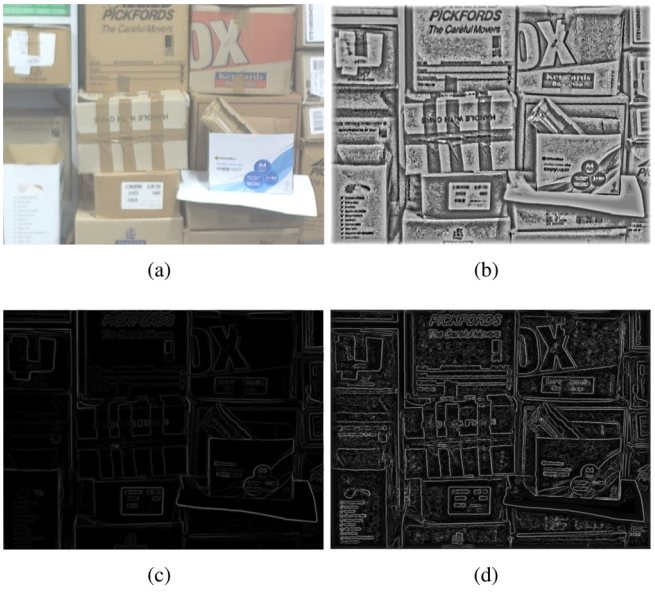
Fig. 10. Experimental results: (a) gradient image without normalization and (b)gradient image with normalization.



(a) (b)

(c) (d)

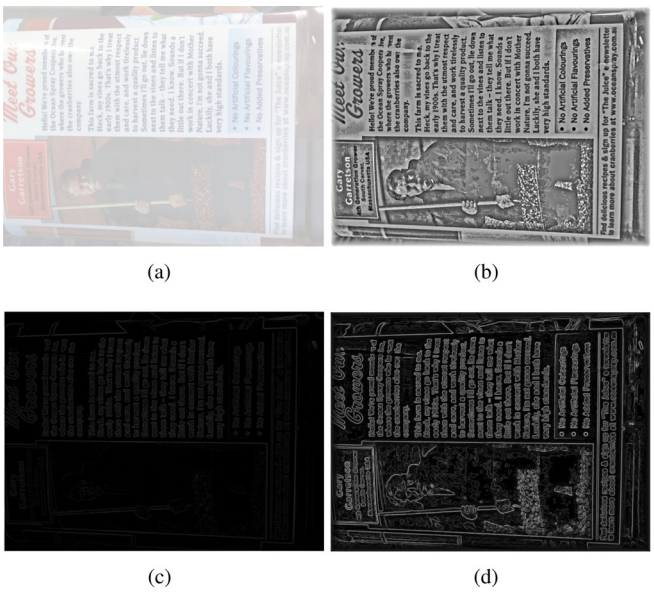
Fig. 11. Experimental results: (a) noisy low contrast image, (b) normalized image, (c) gradient image without normalization and (d) gradient image with normalization.



(a) (b)

(c) (d)

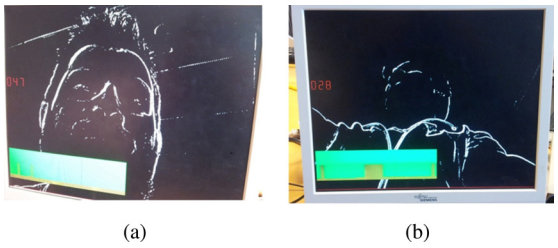
Fig. 12. Experimental results: (a) noisy low contrast image, (b) normalized image, (c) gradient image without normalization and (d) gradient image with normalization.



(a) (b)

(c) (d)

Fig. 13. Experimental results: (a) noisy low contrast image, (b) normalized image, (c) gradient image without normalization and (d) gradient image with normalization.



(a) (b)

Fig. 14. Experimental results: (a) screen shot 1 and (b) screen shot 2.

5. Conclusion

The main aim of this work is to integrate a range finder with real-time image/video processing. Image normalization along with edge detection is successfully executed for video. The range sensor is successfully integrated on FPGA to find the distance from the lens of the camera to the object. From the experimental results, it is found the using image normalization before the edge detection can give more detailed gradient as compare to the one without using image normalization.

References

[1] I.E. Abdou, W. Pratt, Quantitative design and evaluation of edge enhancement/thresholding edge detectors, in: Proceedings of the IEEE, 1979, pp. 753–763.

[2] D.G. Bailey, Design for Embedded Image Processing on FPGAs, John Wiley & Sons (Asia) Pty. Ltd., 2011.

[3] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Pearson Education, Inc., 2002.

[4] S. Hezel, A. Kugel, R. Manner, D.M. Gavrila, FPGA-based template matching using distance transforms, in: Symposium on Field-Programmable Custom Computing Machines, Napa, CA, USA, 2002, pp. 89–97.

[5] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Inc., 1989.

[6] C.T. Johnston, D.G. Bailey, P. Lyons, A visual environment for real-time image processing in hardware (VERTIPH), EURASIP J. Embed. Syst. (2006) 1–8.

[7] S. Kawada, T. Maruyama, An approach for applying large filters on large images using FPGA, in: International Conference on Field Programmable Technology, Kitakyushu, Japan, 2007, pp. 201–208.

[8] M.A.U. Khan, T.M. Khan, R.B. Khan, A. Kiyani, M.A. Khan, Noise characterization in web cameras using independent component analysis, Int. J. Comput. Commun. Control (2012) 302–311.

[9] A. Manan, Implementation of image processing algorithm on FPGA, AKEC J. Technol. (2011) 25–28.
[10] H.S. Neoh, A. Hazanchuk, Adaptive edge detection for real-time video processing using FPGAs, Glob. Signal Process. 2 (2004) 25–28.
[11] Binary to BCD Converter, Double-Dabble Binary-to-BCD Conversion Algorithm, 2012 http://edda.csie.dyu.edu.tw/course/fpga/Binary2BCD.pdf
[12] J. Villasenor, B. Hutchings, The flexibility of configureable computing, IEEE Signal Process. Mag. 15 (1998) 67–84.
[13] R. Zoss, A. Habegger, V. Bandi, J. Goette, M. Jacomet, Comparing signal processing hardware-synthesis methods based on the Matlab tool-chain, in: 6th International Symposium on Electronic Design, Test and Applications, Queenstown, New Zealand, 2011, pp. 281–286.