Proceedings of the 2011 IEEE
International Conference on Mechatronics and Automation
August 7 - 10, Beijing, China

A High Speed Robot Vision System with GigE Vision Extension

Wenhao He, Kui Yuan, Han Xiao and Zhengdong Xu

Institute of Automation
Chinese Academy of Sciences
Beijing 100083, China

{wenhao.he, kui.yuan, han.xiao & zhendong.xu}@.ia.ac.cn

***Abstract* - High speed image and video processing is becoming increasingly important in many applications, especially in robotics. To boost the computing speed of traditional robot vision system, a FPGA and DSP based robot vision system is developed. Considering about the high throughput image acquisition is the premise of high speed processing, a GigE vision interface is also extended. The configuration and some important characteristics of this robot vision system, which can not only capture images rapidly but can also process images using different algorithms in real-time, are described in this paper. Experiment results are also given to show that the newly developed vision system is much faster and more suitable for robot vision applications.**

Index Terms – High Speed, GigE Vision, FPGA.

I. INTRODUCTION

Vision sensor can provide more information for a mobile robot about the environment than most of other sensors, and then it is attracting more and more attentions [1]. Traditional robot vision systems are usually composed of an image grabber and a single-CPU-based computer, such as PC, IPC, notebook computer, etc. [2]. These kinds of vision system always process the image data using the single CPU through software. Due to the great amount of data, the speed of vision system is limited, and the whole function of robot is affected. Therefore, using a single processor and sequential software are not recommended for robot vision system and other real-time vision applications.

With advances in the VLSI technology, FPGA implementation vision system has become an attractive alternative [3,4]. Assigning complex computation tasks to hardware and exploiting the parallelism and pipelining in algorithms yield significant speedup in running times. In this paper, an intelligent image card based on FPGA and DSP and a robot vision system using this intelligent image card is introduced. Considering about the high throughput image acquisition is the premise of high speed processing, a high speed GigE vision interface is also extended in this card. The intelligent card can not only capture images rapidly but also can do the high speed of image processing. So that the speed of robot vision system using this intelligent card can be boost greatly comparing to the traditional robot vision system.

The rest of this paper is organized as follows. Section 2 describes the design of the intelligent image card. The high speed camera interface –GigE Vision and its implementation on FPGA will be discussed in section 3. Section 4 presents algorithms realized on FPGA. In section 5, experiments and discussion is described. Finally, conclusions and future work are given in section 6.

II. DESIGN OF THE INTELLIGENT IMAGE CARD

For developing a high speed robot vision system, we have developed an intelligent image card. The main functions of this intelligent image card are as follows: (i) image capturing; (ii) implementation of image processing algorithms according to the command from host computer; (iii) communication with host computer.

The block diagram of the intelligent image card is shown in Fig. 1.

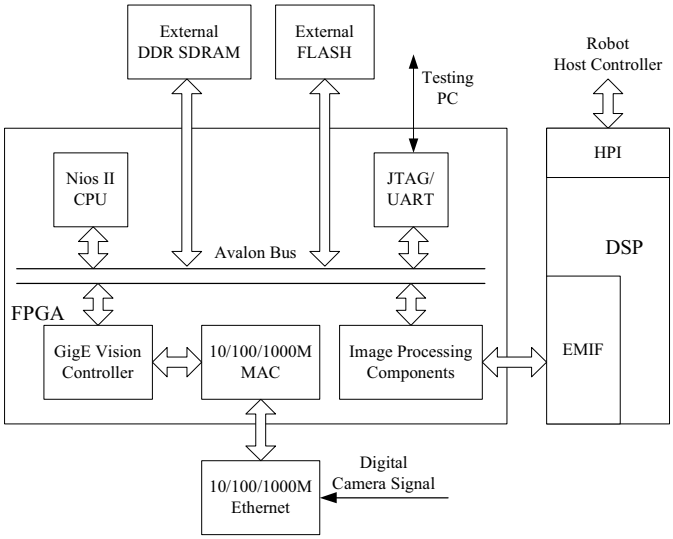

Fig. 1 Block diagram of hardware implementation

The image capturing module consists of a 10/100/1000M Triple Speed Ethernet (TSE) and a GigE Vision Controller. Digital camera data packets is inputted via TSE interface and be decoded by GigE Vision Controller. The main task of FPGA is to implement image capturing module and the image data pre-processing algorithm which need large mount of computation according to the command from the host computer. The Altera EP3C40F484 chip is chosen as FPGA processor for its low cost, high performance application, which has a high quality-to-price ratio. There are 40K Logic Elements, 1Mbit Embedded Memory, and 18Bit x 18Bit Embedded Multipliers in this FPGA chip [5]. With these hardware resources, a SOPC (System on Programmable Chip)

system is designed. The SOPC system is composed of a Nios II processor, a 10/100/1000M trip speed Ethernet MAC, a GigE Vision Controller, a JTAG/UART serial port and User Customized Image Processing Components. All these SOPC modules are integrated by Avalon Bus. A DSP chip TMS320DM642 (from TI) is also used to deal with high-level image processing algorithm and to control the timing of this intelligent image card. The external DDR SDRAM module provides program running space for Nios II CPU and the buffer to save image data temporally for Image Processing Components module. The program of Nios II and DSP is stored in External FLASH module. The interface ports module includes a serial UART port, which can download the configuration data into FPGA or return the testing result to PC via JTAG/UART unit, and a HPI port, which can communicate with Robot Host Controller.

The working procedure of the intelligent image card is as follows:

(i)The Nios II CPU initializes all SOPC units, starts a communication task to wait DSP command.

(ii)The DSP sends out image data acquisition command to FPGA via EMIF port.

(iii)The image capturing module, which is managed by Nios II CPU, receives net packets via triple speed Ethernet port, decodes the data packets and extracts out the image data by GigE Vision Controller, and then sends the image data to Image Processing Components module.

(iv)The Image Processing Components module receive the raw image data, process the data with image processing algorithm (such as filter, edge detect, etc.) which implemented by hardware, and then send the processing result back to communication task.

(v)The communication task sends the results to Robot Host Controller through the HPI port and restarts the data acquisition and processing again.

III. GIGE VISION AND ITS IMPLEMENTATION ON FPGA

The bandwidth of image capturing channel is the premise of a high speed vision system. To get high bandwidth of image capturing, a high speed GigE vision interface is also extended in this card. In this section we will discuss this vision extension and its implementation on FPGA.

A. Introduction of GigE Vision

The GigE Vision is a new standard developed by a committee of the Automated Imaging Association (AIA), for high performance machine vision system [6,7,8]. The GigE Vision standard is based on the User Datagram Protocol (UDP). Instead of establishing a host-to-host connection as with TCP, UDP uses ports to allow application-to-application connections. While this makes UDP less reliable than TCP, it increases high speed image transfer which is really required for high speed vision applications. To overcome the unreliability of UDP, some extra protocols have been added to GigE Vision. These two protocols introduced by the GigE Vision standards committee are GigE the Vision Control Protocol (GVCP) and the GigE Vision Streaming protocol (GVSP). The GVCP defines how to control and configure compliant devices (such as cameras), specifies stream channels, and provides mechanisms for cameras to send image and control data to the central processing units. The main task of GVCP is to add some mechanisms to UDP to guarantee the reliability of image transmission. The GVSP is another application layer protocol that allows an application to receive image data, image information, and other information from a device. GVSP provides mechanisms to guarantee the reliability of packet transmission (through GVCP) and to minimize the flow control required due to the unreliability of UDP [6].

B. GigE Vision for high speed robot vision system

GigE Vision offers many features which make it quite suitable for an image capturing interface in high speed robot vision systems. The main feature is its high bandwidth of data transmission. The actual bandwidth of GigE Vision interface can speed up to 800Mbps, and has the potential for even higher bandwidths with 10 Gigabit Ethernet[9]. Therefore GigE Vision provides enough bandwidth to transmit video in fast frame way which is an important issue for high speed vision systems. The reduced costs and simplified installation is another excellent feature of GigE Vision. Using Power over Ethernet, the connection of camera and the vision system only needs one network cable without extra data grabber hardware. This means the GigE vision system occupies less space which is important for robot system. Due to the reasons outlined in this section, it is the authors' believe that GigE vision is a suitable interface for high speed robot vision systems.

C. Implementation of GigE Vision

As mentioned above, the GigE Vision standard consists of two protocols: GVCP and GVSP. If we implement these two protocols wholly by hardware, the entire project will require more logic resources and lack of flexibility. If we implement them with software, which run by Nios II, the speed of data transmission will be cut down greatly, and the image capturing rate will be lower than 2 frames per second. Therefore a hardware and software co-processing architecture implementation is proposed as follows.

For each UDP data packet consists of source and destination port numbers, the GigE Vision Controller module judges which protocol the incoming data packet belonged to by its packet port number. If the incoming data packet is a GVCP packet, a NicheStack net service software task, which launched by Nios II softcore CPU, will be used to decode the packet. If the incoming data packet is a GVSP packet, the packet will be decoded by hardware for that GVSP packet comprises much more data to be decoded than GVCP packet. This architecture, using the hardware to process mass data GVSP packet and software task to process GVCP control packet, can not only guarantee the speed of whole data transmission but also improve system flexibility. The data flow of GigE Vision Controller module is shown as Fig.2.

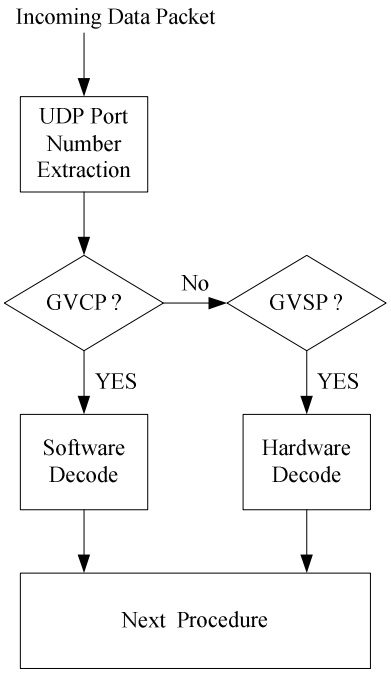

Fig. 2 Data flow of GigE Vision Controller

IV. IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS BY FPGA

Image processing algorithms, such as noise filtering, edge detection, Hough transformation, are computationally expensive and can not be executed easily in real-time by a single processor using software. Recent advances in semiconductor technology have now made it possible to implement a complex algorithm on a FPGA chip. Therefore, using FPGA to realize the image processing algorithm is a good choice for robot vision system since it has the characteristics of concurrency and completes the computing work by hardware instead of software. In this section, two examples will be given to show how an image processing algorithm is implemented by FPGA.

A. Implementation of Gaussian filter

1) Algorithm design

Filters based on Gaussian functions are of particular important because their shapes are easily specified, and suppress the noise in the image effectively[10]. In 2-D space, a circularly symmetric Gaussian has the form of

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

Where σ is the standard deviation of the distribution.

Since the image data are stored as a collection of discrete pixels, a discrete approximation to the Gaussian function is required to perform the convolution. Theoretically, the Gaussian distribution is nonzero everywhere, which would require an infinitely large convolution kernel, but in practice it is nearly zero in the position where it is more than about three standard deviations away from the mean, and so convolution kernel is truncated. Fig. 3 shows a 5×5 digital approximated convolution kernel.

Multiplication requires large hardware resource and longer execution time. So the coefficients in the convolution kernel are selected to be in the form of power of 2. Thus the operations can be performed in hardware by bit shifting, saving hardware resource and reduce delay time.

$$\frac{1}{16}\begin{bmatrix} 2^{-2} & 2^{-1} & 2^{-1} & 2^{-1} & 2^{-2} \\ 2^{-1} & 2^{-1}+2^{-2} & 2^{0} & 2^{-1}+2^{-2} & 2^{-1} \\ 2^{-1} & 2^{0} & 2^{1} & 2^{0} & 2^{-1} \\ 2^{-1} & 2^{-1}+2^{-2} & 2^{0} & 2^{-1}+2^{-2} & 2^{-1} \\ 2^{-2} & 2^{-1} & 2^{-1} & 2^{-1} & 2^{-2} \end{bmatrix}$$

Fig. 3 Gaussian operator for smoothing

2) Hardware implementation

Fig. 4 shows the implementation of Gaussian filter in FPGA. Four FIFO row buffers and a 5x5 moving window Gaussian operator is used. Four FIFO buffers with the depth of one row pixels of image are employed to access all the pixels in the 5x5 window at the same time. Since the design is pipelined, the Gaussian filter starts once the 4 FIFO buffers are full. That is, the output is produced after a latency of fourth width of image plus four cycles. Since the Gaussian filter operation is the sum of product of each pixel and corresponding mask coefficient, an architecture of 5x5 moving window operator is adopted. Using this architecture, multiplication and accumulation of convolution operation can be completed within one clock cycle. Although multiplication usually requires large hardware resource and long execution time, it can be realized easily in the FPGA just by a bit shifting operation with the coefficients, as shown in Fig. 3.

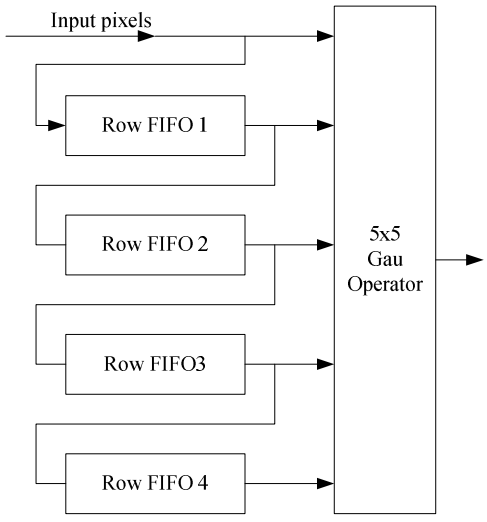

Fig. 4 Hardware implementation of Gaussian filter

B. Implementation of Canny edge detector

1) Algorithm design

Edge detection is another fundamental operation in image

processing applications [11,12]. Canny proposed a new approach to edge detection that is optimal for step edges contaminated by white nose. The algorithm can be shown as following computation [13]:

Suppose G(x,y) is a 2D Gaussian and I(x,y) is the image, we get smoothed image H(x,y) as

$$H(x,y) = G(x,y) * I(x,y) \tag{2}$$

Define direction n as perpendicular to the edge direction, a robust estimate of it can be obtained based on the smoothed gradient direction is available. The normal to the edge n is estimated as

$$n = \frac{\partial H / \partial n}{|\partial H / \partial n|} \tag{3}$$

The edge location is then at the local maximum of first derivative of H(x,y) in the direction n, that is the zero-crossing point of second derivative of H(x,y).

$$\partial^2 H / \partial n^2 = 0 \tag{4}$$

This equation illustrates how to find local maxima in the direction perpendicular to the edge. This operation is often referred to as non-maximal suppression (NMS).

After NMS operating, the data obtained usually contains some spurious responses. This is called the 'streaking' problem and is quite common in the edge detection problem. These streaking can be eliminated by using a threshold with hysteresis.

2) Hardware implementation

As mentioned above, the Canny edge detector can be carried out in the following three steps [14]:

(a) Image smoothing;

(b) Gradient calculation and Directional Non Maximum Suppression;

(c) Thresholding with hysteresis.

The implementation of Gaussian smoothing has been introduced in the previous section. We will introduce the hardware implementation of step b and step c.

Hardware implementation of step b:

This stage implements the calculation of gradient and then directional non-maximum Suppression operation. Since the actual images are always discrete, we define the direction n in (3) as vertical, horizontal, left-diagonal and right-diagonal of the 3x3 adjacent window of current pixel. The first-derivative of each direction is then calculated by (5).

$$E = \left(\{-1,+1\}\right)_{3X3} \times \left(H(i,j)\right)_{3X3} \tag{5}$$

Using a {-1,+1} operator to the adjacent pixels along each direction, we get EV, EH, EDL and EDR, the results of (5) in vertical, horizontal, left-diagonal and right-diagonal directions. The magnitude of gradient of current pixel is the maximum of |EH|, |EV|, |EDR|, |EDL|, and the direction of gradient is one of the four directions corresponding to the maximum of |EH|, |EV|, |EDR|, |EDL|.

$$\left|grads\left(H(i,j)\right)\right| = \max\left\{\left|E_H\right|,\left|E_V\right|,\left|E_{DR}\right|,\left|E_{DL}\right|\right\} \tag{6}$$

$$\theta = Arg\left(\max\left\{\left|E_H\right|,\left|E_V\right|,\left|E_{DR}\right|,\left|E_{DL}\right|\right\}\right) \tag{7}$$

Since 3x3 convolutions are used to calculate the gradients, neighboring 8 pixels are required. In order to access 8 neighboring pixels in a single clock cycle, two FIFO buffers are employed to store the output pixels of the previous stage.

Once the direction of the gradient is known, the pixel that has no local maximum gradient magnitude is eliminated. The comparison is made between the current pixel and its neighbors, along the direction of the gradient. For a 3x3 window is needed during the comparison, 2 FIFO buffers of width of the image are also employed before the comparison. This operating is referred to as directional non-maximal suppression. The pipelined design of this stage is shown in Fig. 5.

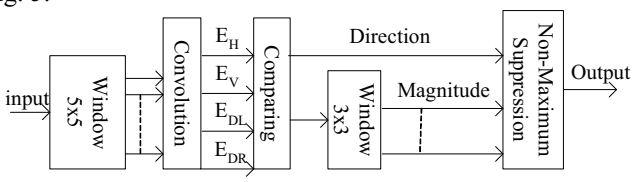

Fig. 5 Pipelined Directional NMS

Hardware implementation of step c:

Since the output of the non maximum suppression stage contains some spurious edges resulted from noises, the method of thresholding with hysteresis is used.

Two thresholds (high threshold Th_h and low threshold Th_l) are employed. Suppose f is the image obtained from the non maximum suppression stage. If the gradient magnitude of f(i,j) is above Th_h,, set $f_1(i,j)$ to 1, else set $f_1(i,j)$ to 0. The image f_1 thus represents the strong edge image. If the gradient magnitude of f(i,j) is between Th_h and Th_l, set $f_2(i,j)$ to 1, else set $f_2(i,j)$ to 0. The image f_2 thus represents the weak edge image.

To get the connection between the weak edge pixel and the strong edge pixel, a 3x3 window is used. If any of the neighbors is a strong edge pixel, the center weak edge pixel is then considered as a strong edge pixel, else it is considered as background pixel. The resultant image is an image with optimal edges. The pipelined design of this stage is shown in Fig. 6.

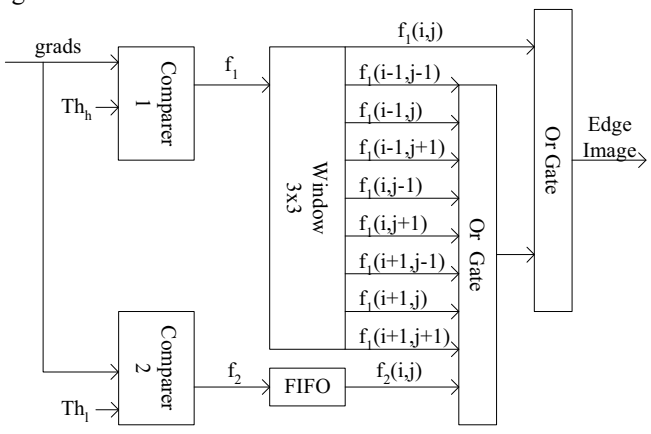

Fig. 6 Pipelined thresholding with hysteresis

IV. PERFORMANCE COMPARISON WITH PC BASED ROBOT VISION SYSTEM

To confirm the performance of the newly developed robot vision system, we have compared some of the main performances of a PC based robot system with the newly developed robot vision system. The configuration of the two systems is as follows:

The PC based robot vision system consists of an IPC board with a Pentium-M 1.6GHz CPU and 256MB memory, a PCI bus image grabber, and an analog CCD camera. The implementation of the image processing algorithms uses the optimized function of Open CV.

Our newly developed robot vision system consists of an intelligent image card with GigE vision extension described in section 2 and section 3, and a GigE vision digital camera. The system picture is shown in Fig. 7. The implementation of the image processing algorithms is done by the FPGA on the intelligent image card.

Fig. 7 Picture of our newly developed robot vision system

Experiments of capturing moving table tennis ball have been carried out to compare the image processing ability of the PC based robot vision system and our newly developed robot vision system. The ball is dropped from a certain altitude, and the two vision systems capture and process continuously. Fig. 8 show the capturing results and processing results of the two vision systems at three different time points during the ball falling. Table 1 gives the capturing speed and the processing time consumption of each vision system.

TABLE I
PROCESSING TIME OF TWO ROBOT VISION SYSTEMS

	PC-based Vision system (Image size: 720x572x8bit)	Our vision system (Image size: 800x600x8bit)
Capturing Speed	30 fps	200 fps
5X5 Gaussian	33 ms	1.3 ms
Canny edge detect	90 ms	2.5ms

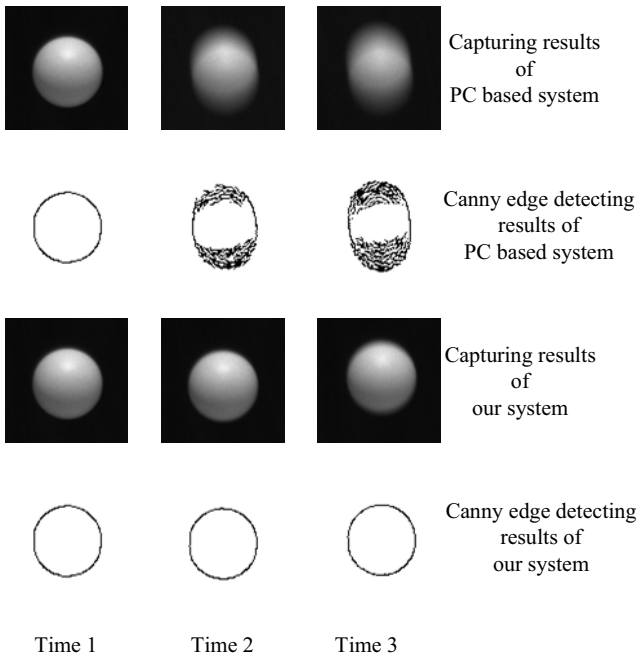

Fig. 8 Capturing and Canny edge detecting results of the two vision systems

It can be seen clearly that our newly developed robot vision system is much faster than the traditional PC based robot vision system and has smaller image trailing when dealing with fast moving objects, which means that it is possible that this kind of FPGA and DSP based vision system can meet the needs of fast capturing and real-time processing for a mobile robot system.

V. CONCLUSION

In this paper, a high speed robot vision system with GigE vision interface is described. Compared with traditional PC based robot vision system, this new robot vision system has the characteristics of high capturing speed, real-time processing and small size. Therefore, this new robot vision system is more suitable for compact high performance mobile robots.

ACKNOWLEDGMENT

This work was supported in part by the National 863 Project (2009AA043902-2) and Key Laboratory of Measurement and Control of CSE(School of Automation, Southeast University).

REFERENCES

[1] Peng Lu, Kui Yuan and Wei Zou. A High Performance Low Power Consumption Robot Vision System. Third International Conference on Natural Computation, 2007, pp. 171–175.

[2] Xing Zhang, M.H. Lee. A Developmental Robot Vision System. Systems. IEEE International Conference on Man and Cybernetics, 2006, pp.2024-2029.

[3] Hamid GholamHosseini and Shuying Hu, A High Speed Vision System for Robots Using FPGA Technology. 15th International conference on Mechatronics and Machine Vision in Practice, 2008, pp. 81–84.

[4] Seunghun Jin, Junguk Cho, Xuan Dai Pham and etc., FPGA Design and

Implementation of a Real-Time Stereo Vision System, IEEE Transactions on Circuits and Systems for Video Technology, Vol.20, 2010, pp. 15-26.
[5] Altera Corporation, "Cyclone III Device Handbook, Volume 1 (Book style)," Available: http://www.altera.com.
[6] Basler Vision Technologies , "The Elements of GigE Vision", whitepaper, http://www.baslerweb.com/.
[7] Basler Vision Technologies, "GigE Vision – CPU load and latency", whitepaper, http://www.baslerweb.com/.
[8] "Can GigE Vision deliver on its promise", Technical white paper, January 2008, http://www.sony-vision.com/
[9] E. Norouznezhad, A. Bigdeli, A. Postula and B. C. Lovell. A high resolution smart camera with GigE vision extension for surveillance applications. IEEE International Conference on Distributed Smart Cameras, 2008, pp.1-8.
[10] V. Muthukumar and V. R. Daggu, "Image Processing Algorithms on Reconfigurable Architecture using Handel-C," in Proceedings of the EUROMICRO Systems on Digital System Design 2004, pp. 218-226.
[11] F. M. Alzahrani and T. Chen, ""A Real-Time Edge Detector: Algorithm and VLSI Architecture," Real Time Image, 1997, pp. 363-378.
[12] Song Yang, Siew-Kei Lam and Srikanthan. An efficient edge and corner detector. International Conference on Control Automation Robotics & Vision, 2010, pp.1628-1631.
[13] Wenhao He and Kui Yuan, An Improved Canny Edge Detector and its Realization on FPGA, 7th World Congress on Intelligent Control and Automation, 2008, pp. 6561-6564.
[14] Daggu Venkateshwar Rao, Muthukumar Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using Handle-C," Information Technology: Coding and Computing, International Conference, vol.2, 2004, pp. 843-847.