# A High Speed Vision System for Robots Using FPGA Technology

Hamid GholamHosseini, Shuying Hu
School of Engineering, Auckland University of Technology,
Private Bag 92006, Auckland 1142, New Zealand
Email: hgholamh@aut.ac.nz

***Abstract*-High speed image and video processing is becoming increasingly important in many applications, especially in robotics. A high speed vision system involves grabbing image frames from a single or multiple sensors and processing of the data in a limited time. Therefore, the requirement of real-time processing of the images is the key problem in dealing with such applications. Moreover, to obtain the best performance, it is vital that algorithms and hardware of a vision system be reconfigurable and flexible. A FPGA-based system was devoted to the design of vision system for robots. It offers the required features for robots' vision system such as; parallel processing of the video images, good computational power, low cost and reconfigurable hardware. The image processing component of the proposed vision system consists of one core processor for implementing the software part and up to eight functional units for the hardware implementation part. The core processor controls and communicates with the functional units. The mixed hardware/software method utilises the advantages of both hardware and software techniques. The implementation of a normal filter with a coefficient matrix of 3 x 3 as applied to a 64 x 64 image is chosen as an example. It can be verified that the multiplication operations for implementing such filter are independent and therefore can be run simultaneously. The addition operations can be done when the multiplication results are available. Therefore, the core processor can perform all parallel operations in order to minimise the total number of implementation stages. The Nios II Cyclone EP1C20 development board is used for implementing the vision system. It was found that the proposed system processes the selected grey scale images at a speed of about 80 times faster than the traditional software based methods.**

## I. INTRODUCTION

To perform embedded face detection and facial feature extraction algorithms, electronic vision systems employ a wide variety of devices as microcontrollers, general processors, Digital Signal Processors (DSPs) and reconfigurable structures as FPGAs (Field Programmable Gate Array). With advances in the VLSI technology FPGA implementation has become an attractive alternative. Assigning complex computation tasks to hardware and exploiting the parallelism and pipelining in algorithms yield significant speedup in running times.

Traditional robot vision systems transmit original image data from robot to computer through a video cable or RF antenna. After processing this data, the result is sent to the robot again through data bus, then controlling the robot to move to the appointed position. Due to the great amount of data, the speed of the vision system is limited, and the whole function is affected. Using a single processor hardware and sequential software are not recommended for real-time vision and other perception applications.

High speed and reliable processing units as well as sophisticated software development environment are needed to satisfy the real-time processing requirements of a robot vision system [1]. On the other hand, using multiple CPU for these vision systems is costly and inefficient [2].

Digital Signal Processor is a special microprocessor with multi functional units which can process complex mathematics as well as images [3]. Recently, Texas instrument TMS320C67xx been used as a low power processor with floating point feature and dedicated image processing library for an embedded vision system for mobile robots [4].

On the other hand, FPGA is a high performance embedded processor which is a low cost, high speed and reliable platform. DSPs are more flexible than FPGAs for performing complex calculations; however, FPGAs are much faster than DSPs.

In one example, a TI's TMS320C80 programmable DSP and a Xilinx FPGA were compared by implementing a typical image processing task. It was found that for processing the same image with the same task it takes 6.9ms by FPGA as opposed to 252ms using DSP [5]. Therefore, an FPGA-based system can be devoted to the design of high speed vision system for robots.

## II. HARDWARE/SOFTWARE CO-DESIGN METHOD

With the development of FPGAs, their superior features become more evident. However, it has been noticed that FPGA designs are still in the traditional hardware design domain with limited available software. We have employed a FPGA design method which considers advances in design tools as well as supporting software-oriented techniques for programmable hardware platforms [6].

The image processing component of the proposed vision system consists of a core processor for the software part and up to eight functional units for the hardware implementation part. The core processor controls and communicates with the functional units. By the new design method, users should be able to compile algorithms for both core processor and the functional hardware units. A test environment was created to allow both software and hardware versions to be evaluated and measured by identical inputs. As software designs are more flexible than hardware designs, there is no need to write low-level hardware descriptions when implementing a hardware/software (H/S) co-design application.

The mixed H/S method utilises the advantages of both hardware and software techniques. Moreover, the ability to compile software algorithms directly for the FPGA platforms makes them more efficient for less cost, as well as low risk applications.

Filtering is the most common method in processing of the input images in robot vision systems. Each pixel of the filtered image is computed as a function of one or several pixels (within its neighbourhood) of the input image. The performance of filters depends on the filter transfer function. For example, choosing an interpolation function for a filter may produce a smoother image than the original. It is required that the output pixel values are computed from minimum number of neighbourhood pixels, otherwise the resulting image may get blurred [7]. The filtering process creates a new image as a result of processing the neighbourhood pixels of an existing image. This is the base of the robot vision image processing algorithms.

## III. Design Structure

### A. *Parallel method*

In general, the expression of a normal filtering operation of an image, *img*, with the size of $n \times m$ by a filter function, $f$, with the size of $x$ *by* $y$ is given by [8]:

$$R(i,j) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} f(s,t)\, img(i+s, j+t) \qquad (1)$$

where $a$ and $b$ are assumed to be $(x-1)/2$ and $(y-1)/2$, for $i = 0, 1, \ldots, n-1$ and $j = 0,1,\ldots, m-1$.

In order to explain the methods of implementing such filters, a normal filter with a coefficient matrix of 3 x 3 as applied to a 64 x 64 image is chosen as an example. According to equation (1), there are 9 multiplication and 8 addition operations for a 3 x 3 filter to process each pixel. It can be verified that all multiplication operations are independent of each other and therefore can be run simultaneously [8]. Moreover, the addition operations can be done after the multiplication results are available. Therefore, the core processor can perform all 17 operations individually within 5 steps as shown in Figure 1.

Each operational step can only be run when the previous step has been completed. However, all operations of the same step can be run simultaneously. Due to the complexity of multiplication operation, it takes much longer to be performed than the addition, so step 1 (Figure 1) is the longest step of all the steps.

It is observed that for performing this filtering example, only 5 steps are needed in comparison with the 17 steps required for the same operation using direct method as expressed by equation (1). Therefore, the parallel implementation of this filter is faster than implementing this filter using direct method.

The number of steps is distinctly related to the filter size as well as implementation method. In general, the number of operational steps to implement a filter of size I x I using direct and parallel methods can be calculated as follows:

*1) Filtering, using direct method requires:*

The number of steps for multiplication operations is:

$M = I*I$,

The number of steps for addition operations is:

$A = I*I - 1$,

The total number of steps is:

$S = M + A = 2*I*I - 1$.

*2) Filtering, using parallel operation method requires:*

The number of steps for multiplication operations is:

$M = 1$,

The number of steps for addition operations is:

$A = 2*(lgI/lg2)$,

The total number of steps is:

$S = M + A = 1 + A = 1 + 2*(lgI/lg2)$.

Since the multiplication time is much longer than that the addition, the step time is different.

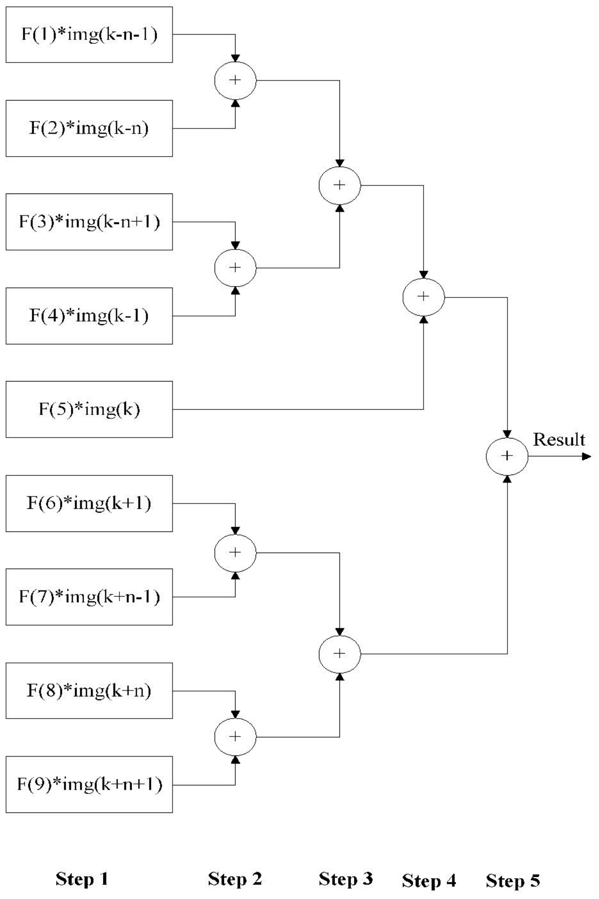

Figure 1: The flow chart of parallel implementation of a filter using 9 multiplications (synchronized) and 8 additions.

## IV. Hardware Filter Design Using Functional Units

To implement the filter in hardware based on equation (1), two First-In First-Out (FIFO) buffers are used. Image FIFO is for storing the image data coming through input A and Filter FIFO is for storing the filter coefficients collecting from input B (Figure 2).

The Image FIFO stores the image pixels from k–n–1 to k+n+1. As the filter coefficients are altered, another memory (the Filter FIFO) is needed to store the coefficients of f(1) to f(9). Therefore, the entire hardware structure consists of 3 parts; the Image FIFO, the Filter FIFO and a Logic Circuit, which performs the calculations. This structure is shown in Figure 2.

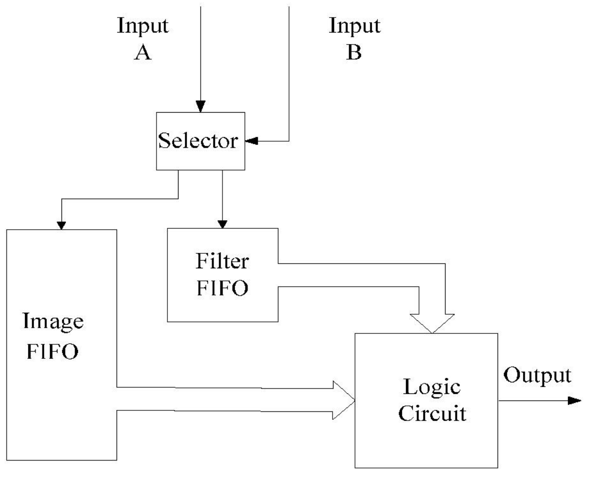

Figure 2: The hardware structure of a filter functional unit.

## V. Implementation and Results

Quartus II as a design software can provide a multiple-platform design environment to meet all design requirements. It includes a synthesis environment and basic design tool for System on Programmable Chip (SOPC). The system is also composed of a Nios II processor, which has easy-to-use development kit, and a Nios II Integrated Development Environment, which provides a full-featured source editor, and a C/C++ compiler. Considering the good features of the Nios II, Altera Cyclone EP1C20F400C7 development board is selected for this project.

A hardware filter functional unit is attached to the Nios II processor. The interface of this functional unit is a standard ALU interface. This hardware filter can be executed by simply using the relevant software code. There are two steps for the hardware implementation of a filter:

i. Initialisation of the hardware.
ii. Sending image data to the filter.

The test image is stored in the onboard flash memory and a personal computer. The board communicates with the PC via the serial "com" port and the "JTAG".

When the system configuration program finishes compiling, a JTAG port is used to download the program from the PC to the development board. During the test process, the "com" port is used to download the test program from the PC to the development board. After the image is processed, the processed image data will be sent from the main memory of the test system (generated together with the Nios II processor) to the PC via the "com" port. Furthermore, the JTAG port can be used to send the test results to the PC. When the testing system is running, a Nios II processor and hardware filters (such as normal, dilation, erosion and smoothing filters) will be generated in the development board. The C code will run in the Nios II processor (Figure 3).

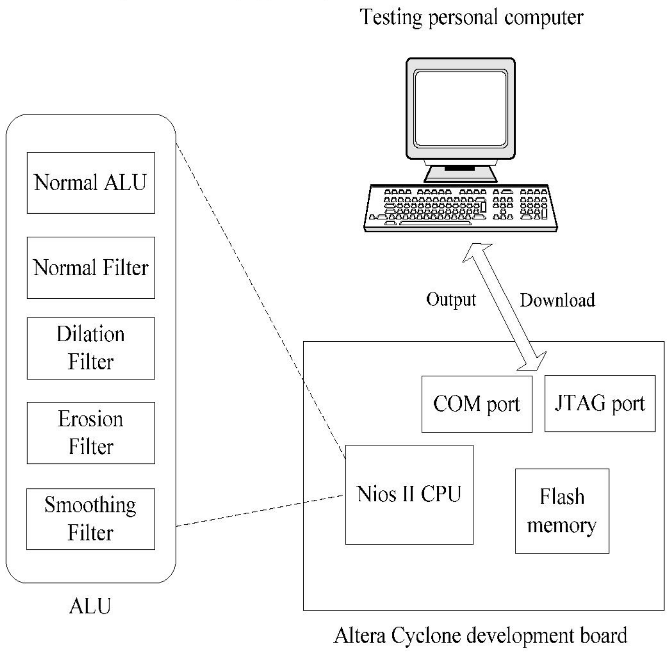

Figure 3: The testing system block diagram.

The project can be extended to implement other filters for images of any size. Other filter structures including Laplacian, vertical edge detection, dilation and erosion were tested both by employing direct method and mixed H/S co-design method. The hardware filter with parallel structure can provide a better speed performance than a filter implemented by software. On average, the processing speed of the proposed structure is about 80 times faster than the software method.

Table 1 indicates the computational time of the tested filters using software only and the H/S co-design methods.

In this table the input image for the first four tested filters (Laplacian, vertical edge detection, dilation, and erosion) is a grey scale image of 64x64 pixels and the input image for the last two filters (basic average and Gaussian) is a grey scale image of 256x256 pixels.

TABLE I
COMPUTATIONAL TIME OF THE TESTED FILTERING OPERATIONS USING TWO DIFFERENT METHODS.

| Type of filter | Computational time (ms) | |
|---|---|---|
| | Software only method | Hardware/Software Co-design |
| 3 x 3 Laplacian | 1135.22 | 13.9 |
| 3 x 3 Vertical edge detection | 1132.7 | 13.9 |
| 5 x 5 Dilation | 416.09 | 13.9 |
| 5 x 5 Erosion | 372.5 | 13.9 |
| 3 x 3 Basic averaging | 20162.52 | 208.2 |
| 3 x 3 Gaussian | 20162.53 | 208.2 |

## VI. DISCUSSION AND CONCLUSION

There are many ways to realise the image processing tasks of robot vision systems, i.e. general processor, DSP chips, and application specific integrated circuit (ASIC). The general processor can not satisfy the demand of high-speed processing as it is designed for general purpose applications. DSP has pipeline architecture and can accelerate image processing; however, it is slower than FPGA. ASICs are single chip implemntations, however they are expensive and not adaptable. Moreover, a FPGA/DSP mixed platform can be used for robotic vision system in which the DSP works, from a functional point of view, as a supplementary FPGA operator [10, 11].

The FPGA based system with embedded processor or DSP kernel becomes the hardware platform of SOPC and supports co-operating of H/S. This paper is aimed to utilise H/S co-design to improve the processing speed and increase application flexibility of the vision system in robotics using FPGA. Parallel processing not only increase the processing speed but it can also give robot vision system a cognitive behavior by simplifying some complicated high-level visual tasks [10].

Filtering operation which is the most common technique in image and video processing was chosen to test the performance of the proposed technique.

The important features of a FPGA-based system for realising the proposed H/S co-design method were investigated. Comparing with the direct software methods, the proposed FPGA-based system achieves about 80 times faster processing time compare with the traditional hardware methods.

Image FIFO is used in the proposed system to store the image data coming through input A. To achieve a higher speed, the system needs to read the image data directly from a hardware device (such as a digital camera) or a memory. After processing the image by FPGA, it needs to store the processed data in a memory or send it directly to another hardware device

As this structure combines the advantages of traditional software and hardware methods, it can be used in further stages of the development of this research.

Future work will be involving FPGA implementation of face detection and facial feature analysis algorithms for robot vision.

## REFERENCES

[1] K. M. Hou, A. Belloum, E. Yao, X. W. Tu, M. Shawky, M. Meribout, J. L. Mayorquim, A. Trihandoyo, and B. Jardin, "A reconfigurable and flexible parallel 3D vision system for a mobile robot", *Proceedings of Computer Architectures for Machine Perception,* pp. 215-221, 1993.

[2] A. A. Jerraya and W.Wolf, *Multiprocessor Systems-on-Chips*, Morgan Kaufmann, 2004.

[3] R. Oshana, *DSP software development techniques for embedded and real-time systems*, Burlington, MA, Newnes, 2006.

[4] N. Sawasaki, M. Nakao, Y. Yamamoto, K. Okabayashi, "Embedded vision system for mobile robot navigation", *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006*, pp. 2693-2698, 2006.

[5] D. C. M. Bilsby, R. L. Walke, and R. W. M. Smith, "Comparison of a programmable DSP and a FPGA for real-time multiscale convolution", *High Performance Architectures for Real-Time Image Processing (Ref. No. 1998/197), IEE Colloquium on,* pp. 1/6-4/6, (1998).

[6] A. A. Jerraya and W. Wolf, "Hardware/software interface codesign for embedded systems", *Computer*, vol. 38, pp. 63-69, 2005.

[7] G. Mischler, "ImageFiltering", http://www.schorsch.com/kbase/glossary/image_filtering.html, retrieved 1/9/2008.

[8] R. E. W. Rafael C. Gonzalez, *Digital image processing*, 2nd Edition, Upper Saddle River,N.J., Prentice Hall, 2002.

[9] M. A. Barry Wilkinson, *Parallel programming: techniques and applications using networked workstations and parallel computers*, Upper Saddle River, NJ, Pearson Prentice Hall. 2004.

[10] F. Dias Real, P. Chalimbaud, F. Berry, F. Marmoiton and J. Serot, "Embedded Early Vision systems: implementation proposal and Hardware Architecture", *Cognitive System for interactive sensor* (*COGIS 2006*), Paris, France, March 2006.

[11] K. Shimizu and S. Hirai, "Implementing Planar Motion Tracking Algorithms on CMOS+FPGA Vision System", *Proceedings of the 2006 IEEE/RSJ, International Conference on Intelligent Robots and Systems*, pp. 1366-1371, Beijing, China, October 9-15, 2006.