Reconfigurable Frame-Grabber for Real-Time Automated Visual Inspection (RT-AVI) Systems

Cuenca Asensi, S[1].; Ibarra Picó, F[2]. ;Alvarez, R[3].

Universidad de Alicante, Departamento de Tecnología Informática y Computación,
Campus de San Vicente, Alicante, Spain
[1]sergio@dtic.ua.es, [2]ibarra@dtic.ua.es, [3]rias@alu.ua.es

Abstract. In most of the automated systems for visual inspection tasks, real time requirements constitute an important aspect to have in to account in the design of them. Often, a frame-grabber attached to a MMX-optimised software libraries are not enough to satisfy the above requirements and it is necessary to use expensive specialised hardware and architectures. Reconfigurable hardware gives us the best of both worlds: the flexibility of software and the high performance of customised hardware. In this paper we present a reconfigurable frame-grabber concept to integrate complex real-time processing functions needed for high-speed line inspection applications directly on-board. This allows the efficient hardware-software co-design to achieve high-performance low-cost solutions .

1 Introduction

In order to implant RT-AVI systems on industrial environments it is necessary, on one hand, to satisfy requirements of real time, robustness and reliability, and on the other, requirements of economic profitability. A strong commitment exists between the first ones and the last one.

The final price of an application and therefore its economic feasibility will depend on the development costs, and the cost of the image acquisition and processing hardware. The different nature of the working environments and activities make very useful the use of commercial software supporting real time capable hardware during the development tasks. Nowadays, most image processing systems are implemented using MMX-optimised software libraries [1] [2], running on personal workstations. These kind of systems offer a good performance/cost ratio covering a wide spectrum of applications. However, for more demanding time requirements, e.g. web inspection, document imaging or quality control in high-speed line inspection, it is necessary to use specialised hardware and architectures [3],[4],[5],[6]. This customised hardware usually increases the cost, reduces the flexibility and limits the applications of the system.

Some approaches based on FPL has been proposed. In the intelligent camera concept [7] [8] an FPGA is directly connected to the video-data stream and outputs data to a low bandwidth output bus. This eliminates the need for external frame-grabber, but limits the processing to 1D operations followed by a data compression algorithm. On-board hardware processing concept has been applied to several frame-grabbers [2] [9], these include FPL parts to perform a limited number of basic pixel transformation like thresholding, gain&lighting correction, gray scale, or 1D filter. These boards are attached to the PCI bus eliminating the need of data compression and leaves to the host the more sophisticated operations.

Our proposal, reconfigurable frame-grabber for texture analysis (RCFG), enlarges the possibilities of this concept to support complex algorithms for textured surfaces inspection and takes advantage of the FPL reconfigurability to suit the algorithm particularities. The on-board processing allows supplying the main processor with elaborated data rather than the unprocessed frame, and in the most of the cases, reducing the data bandwidth required between the acquisition system and the main processor. The RCFG integrates the processing functions on data-stream thus providing higher parallelism possibilities to reduce the frame latency. On the other hand, using reconfigurable hardware (FPGAs), we can make this frame grabber more flexible while retaining high speed performance. In this way, we only have to reprogram the hardware to perform a different pre-processing operation to each frame or application; and if we consider the frame grabber attached to the main processor with a bi-directional bus, the functionality of the frame grabber could be changed on demand in milliseconds.

2 Algorithms

Texture is an important characteristic when considering automatic inspection for quality control. A wide variety of measures have been proposed related to texture properties. Among them, statistics measures are widely used in the classification and inspection of textured surfaces [10],[11],[12],[13] but although the performance of such algorithms is usually very good [14], their structure is complex and the data flow process is large. Consequently, the computation cost is high and the implementation in high speed production lines is difficult. In this work we propose a reconfigurable frame-grabber concept that permits the efficient hardware implementation of first and second order histograms extraction for statistical-based texture analysis.

The statistics used are generally based on the distributions of pixel features like pixel intensity, edginess magnitude, edginess direction, or sum and difference of intensity between neighbour pixels. When characterizing textures, both the individual elements and the statistical features derived from them may be used. Some of the most common statistics used are: *Maximum Probability (Mp) , K moments (Mk), K Inv. moments(Imk), Energy(En), Entropy(Et), Skew (Sk), Kurtosis (Ku), Cluster Shade(Cs), Clust. Prominence(Cp), Haralick's Correlation (Hc), etc…*

On the other hand, the histograms may be based on the probabilities of single occurrences (first order histograms) or joint occurrences (second order histograms).

Some examples of first order histograms are:

Grey level histogram (GLH) computes the grey level probabilities *P(i)* of the image, where; *i*=0, 1, 2…G, and G is the number of grey levels.

Edginess histogram (EH), in this case the gradient with displacement *d* is calculated for every pixel, and then the histogram of gradient magnitude or direction probabilities are computed.

Some second order histograms are:

Grey level coocurrence histogram (GLCH) is based on the coocurrence matrix [1], this is constructed from the image by estimating the pair wise statistics of pixel features. Each element *(i,j)* of the matrix represents an estimate of the probability that two pixels with a specified separation have levels of the feature *i* and *j*. This probability can be estimated as $P_{q,d}(i,j)=P(i,j)= C_{q,d}(i,j)/N$. Where $C_{q,d}(i,j)$ is the number of cooccurrences, the separation is specified by a displacement *d* and an angle *q*, and *N* is the total number of cooccurrences. The coocurrence histogram has GxG bins, where G is normally reduced to 32 or 16 grey levels.

Grey level sum and difference histograms (GLSH, GLDH) are similar to coocurrence, these are the histograms of the sum and difference of all pixels d_x and d_y apart. Similar features to coocurrence can be extracted combining sum and difference histograms. The probability distribution of GLDH can also be used for texture classification [3]. In this way, *DIFFX* and *DIFFY* are the histograms of absolute grey level differences between neighbour pixels computed in horizontal and vertical directions, respectively, while *DIFF2* accumulates absolute differences in vertical and horizontal directions and *DIFF4* in all four principal directions respectively, in a single histogram.

For texture classification, all these algorithms are usually applied on square image sub windows, mainly with 32x32 or 64x64 pixels and G=256, 32, 16 grey levels.

3 Design and Implementation of the Proposed Architecture

3.1 Logical Design

The general task that the RT-AVI system has to perform could be described as follows: every frame of RxC pixels is divided in R/SxC/S sub-windows of SxS pixels, then N features are extracted from every pixel and H histograms of B bins are calculated for every sub window. When the frame has been processed, NxR/SxC/S histograms are transferred to the host for statistics calculation and classification. If we analyse the actual calculations made from image acquisition to statistics classification we find four clearly separated stages shown in figure 1: image pre-processing, histogram calculation, statistics calculation and finally texture classification. The first two stages involve intensive computation on integer data so these tasks can be easily carried out by the reconfigurable parts of the frame grabber. The sophisticated floating point calculations required in the third and fourth stages are left to the main processor because of its superior performance/cost ratio in these tasks.

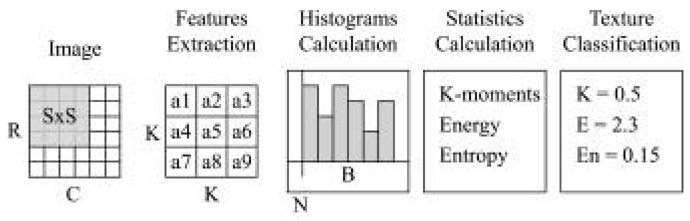

Fig. 1. Basic processing stages

A very simple and primitive solution would be as shown in figure 2. In this version we first read a frame from the camera and store it into the frame grabber's RAM; then we process that frame and calculate the histograms for each sub window, storing that histograms also on the on-board RAM; finally the host computer reads the histograms from the frame-grabber RAM and calculates statistics from them. This is far from optimal because it does not use any parallelism.

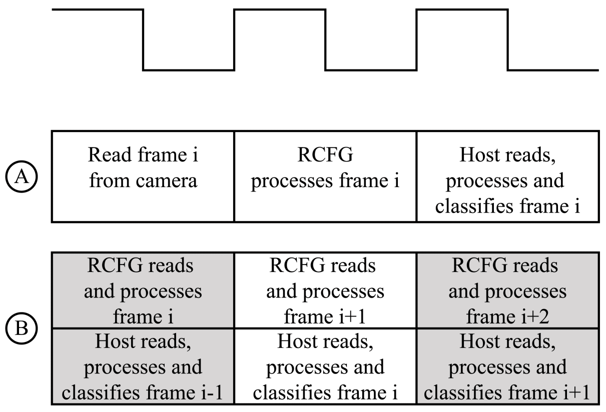
A
Read frame i from camera
RCFG processes frame i
Host reads, processes and classifies frame i
B
RCFG reads and processes frame i
RCFG reads and processes frame i+1
RCFG reads and processes frame i+2
Host reads, processes and classifies frame i-1
Host reads, processes and classifies frame i
Host reads, processes and classifies frame i+1

Fig. 2. Primitive solution with no parallelism (a) vs. parallel solution (b)

A better solution would try to improve performance by introducing parallelism at two different levels: camera - RCFG and RCFG - host. If we could process the frame line by line instead of the full frame at once, we would be processing the image at the same time we are reading it rather than after it has been read. Looking at the host, the RCFG does not need to wait until the host finishes the statistical calculations, instead it can be processing the next frame while the host processes the previous one.

To implement the parallelism between the camera and the RCFG we need to read the first K-1 lines before we start any processing since we need the lines above and below the one we are processing to implement the KxK convolution. Once we have read those lines we should pipeline this operation and keep processing a line while a new one is read. This requires some FIFOs to store these lines and maintain the pipelined execution. To parallelise the execution of the RCFG and the host, we must have, at least, two banks of onboard RAM so while the host reads the histograms for the previous frame the RCFG is writing the histograms for the next frame on a separate bank. Any single bank can be accessed by the host or the RCFG but not at the same time.

Figure 3 shows an overview of the architecture. Due to the simplicity of the pre-processing operations and taking advantage of the parallelism of the logical blocks of the FPGAs, the pixel data stream can be processed by fixed-point arithmetic units in a pipelined fashion. To provide greater flexibility we can assume the extraction of pixel features as a generic KxK convolution of the image pixels allowing simple additions, subtractions or even more complex pixel combinations involving more than just two pixels. K-1 FIFOs are required to pipeline these pre-processing operations. Using the pre-processing module several features can be extracted from every pixel allowing simultaneous calculation of different histograms for every sub-window.

Histogram computation is similar for both first and second order statistics. The histograms values (bins) are stored in external memories (Histogram Buffers); the pixel features previously extracted are used by the Address Generators to create the addresses of the bins that have to be incremented. An incrementer is used to carry out this operation and return the new bin value to the corresponding histogram buffer location. The Address Generators are critical components because they have to take into account the sub-window where the current pixel is included, hence all the histograms (N per sub-window) are calculated at the same time.

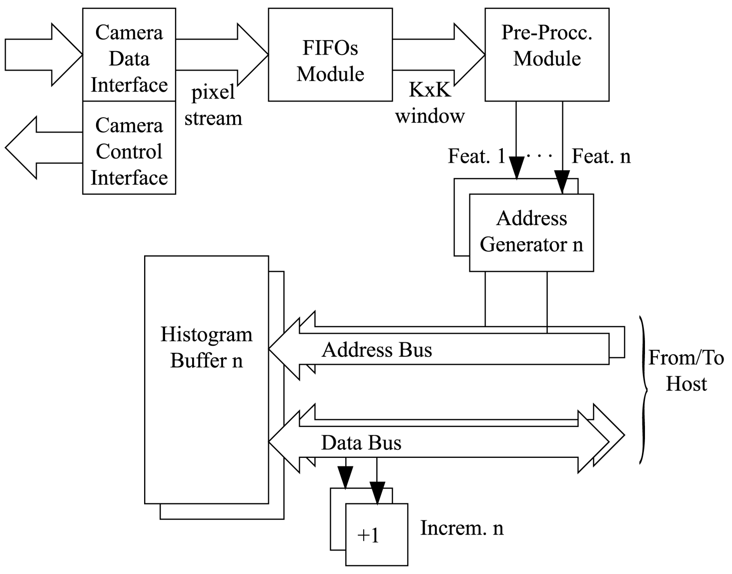

Fig. 3. Overview of the RCFG architecture

Several parameters of the architecture have to be set to perform the different measures, table 1 gives some of the possibilities. E.g. to implement GLCH with rotation invariant and taking windows of 32x32, the selected parameters are: S=32, N=4 (d=1 and q=0º, 45º, 90º, 135º), *Pre_proc* = Concatenation.

Table 1. Setting up the RCFG to perform different measures

Algorithm	S (window size)	N (nº of Hist).	Pre_proc
GLH	64, 32, 16	1	IDENT
EH	64, 32, 16	1	SUB, CONV
GLCH	64, 32, 16	1 to 4	CONCAT
GLSDH	64, 32, 16	2 to 8	SUB, ADD
DIFF	64, 32, 16	1 to 4	SUB, ABS

3.2 Prototype Implementation

In order to validate the proposed architecture we are currently using high performance prototyping board, Celoxida RC-1000PP [15], and a line scan CCD camera, DALSA Spark [8]. Spark camera provides 2048 pixels resolution with 8-bit data @30MHz with a maximum line rate of 12KHz. The output is EIA-644 (LVDS) format thus it can be directly attached to the RCFG by means of LVDS CMOS line drivers and receivers.

The RC-1000PP is a PCI board which carries a Xilinx Virtex V1000 FPGA device. This card is PCI compliant and is plugged into a PC computer (in our case a Pentium III) allowing simply interfacing between host and frame-grabber and bi-directional high rate data transfers. The on-board memory consists in four banks of asynchronous static RAM having 2Mbytes each and they can be accessed either in 8 or 32 bit mode. The card comes with a library of C functions that can be used by the host software to interface with the card (reading and writing to onboard memory, programming the FPGA, etc.). A fairly revolutionary language to specify the FPGA design is also included; this language is called Handel C and is extremely similar to conventional C though modified a little to allow hardware particularities. This language makes prototyping and design implementation extremely fast because it is mostly software oriented.

In the actual prototype we define the frame as a 1024x1024 256 levels grey scale image and is divided in 64x64 windows making a total of 256 windows. The histograms are stored as 32 bits values even though 16 bit values are more than enough but the RC-1000PP does not support 16 bits memory accesses. Thus 1Mb is required to store the processed frame and 256Kb are required to store the first order histograms of a single frame. The full process is divided into two stages; in the first stage (stage A) the FPGA writes the processed image in bank 0 and simultaneously the histograms in bank 1 while the host reads banks 2 and 3 corresponding to the previous frame; in stage B the bank assignment is reversed and the FPGA writes to banks 2 and 3 and the host reads from banks 0 and 1. This division provides the required parallelism between host and FPGA. To achieve the desired parallelism between the FPGA and the camera a set of 6 on chip RAMs have been implemented, each consisting of 1024 positions of 8 bits. These RAMs are used alternatively on odd and even lines so the required pipelining is possible. The actual prototype implements the parallel version explained before, and is capable of performing a 3x3 convolution and storing the processed image and the histograms for the sub windows in 2 cycles per pixel (if we ignore pipelining start-up).

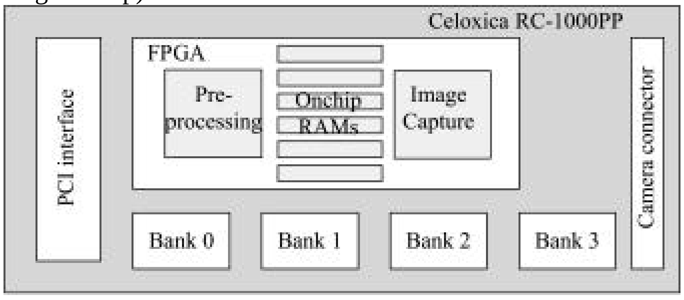

Fig. 4. The prototype architecture

4 Conclusion

The main proposal of this paper is the fact that by enhancing the frame-grabber to perform first and second order histogram computations, a higher degree of parallelism is achievable. Because of the parallelism introduced between the frame grabber and host, image acquisition, pre-processing and histogram calculation take virtually no time because they are simultaneous to the statistics computed by the host.

Nevertheless, the usage of reconfigurable hardware in the frame grabber provides extremely high speed computations with excellent flexibility allowing to change completely the functionality of the frame grabber simply by reconfiguring the FPGA and adjusting the image acquisition system to any new algorithms that may be developed in the main processor.

The degree of parallelism of this solutions depends greatly on the memory architecture of the system. We could greatly increase performance if we had more memory banks and therefore the possibility to calculate two or more histograms in parallel. When increasing the parallelism of the frame grabber the performance of the camera is critical since you may reach the point where you have to wait for the camera to provide pixels to continue computations.

References

1. Matrox, inc. http://www.matrox.com.
2. Imaging Technology, inc. http://www.imaging.com
3. Baykut A. et al. Real-Time Defect Inspection of Textured Surfaces. Real-Time Imaging 6, 17-27, 2000.
4. Wei-Bin Chen and Gëtan Libert. Real-Time Automatic Visual Inspection of High-speed Plane Products by Means of Parallelism. Real-Time Imaging 4, 379-388, 1998.
5. http://vetech.com
6. http://www.datacube.com

7. S.Hossain Hajimowlana, et. al. An In-CameraDataStreamProcessing SystemforDefectDetection in Web InspectionTasks. _ . Real-Time Imaging 5, 23-34, 1999.
8. Dalsa inc. http:// www.dalsa.com/
9. I2S, inc. http://www.i2s-linescan.com/
10. R.M. Haralick. Computer and Robot Vision. Vol I. Addison-Wesley, New York, 1992.
11. P.C. Chen and T. Pavlidis, Segmentation by texture using a co-occurrence matrix and a split-and-merge algo-rithm, Computer Graphics and Image Processing **1**0, pp. 172-182 (1979).
12. D. Harwood, T. Ojala, M. Pietikäinen, S. Kelman and L.S. Davis, Texture classification by center-symmetric auto-correlation, using Kullback discrimination of distributions, Pattern Recognition Letters **1**6, pp. 1-10, (1995).
13. K. Shiranita, T. Miyajima and R. Takiyama. Determination of meat quality by texture análisis. Pattern Recognition Letters, 19: 1319-1324, 1998
14. T. Ojala, M. Pietikäinen and D. Harwood. A comparative study of texture measures with classification based on feature distributions. Pattern Recognition 29(1):51-59, 1996.
15. Celoxica, inc http://www.celoxica.com