Image Registration of Real-time Video Data using the SONIC Reconfigurable Computer Platform

Wim J. C. Melis[1], Peter Y. K. Cheung[1], Wayne Luk[2]

[1]Department of Electrical & Electronic Engineering
Imperial College, Exhibition Road
London SW7 2BT, England

[2]Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, England

Abstract

This paper is concerned with the image registration problem as applied to video sequences that have been subjected to geometric distortions. This work involves the development of a computationally efficient algorithm to restore the video sequence using image registration techniques. An approach based on motion vectors is proposed and is found to be successful in restoring the video sequence for any affine transform based distortion. The algorithm is implemented in FPGA hardware targeted for a reconfigurable computing platform called SONIC. It is shown that the algorithm can efficiently restore the video data in real-time.

1. Introduction

The computational demand and the high data throughput of real-time video image processing have long been recognised as a niche area for reconfigurable computing. Many successful image and video applications have been implemented on various general purpose reconfigurable computing platforms [1] [2] [3]. Unfortunately most of them never go beyond the few basic common applications such as convolution, correlation, edge detection and linear transformations. Together with Sony Broadcast, Imperial College has developed a dedicate reconfigurable computing architecture, known as SONIC, which was designed specifically for broadcast video image editing, manipulation and processing [4] [5]. Many interesting video effects and processing problems have been successfully implemented on this experimental research platform. This paper is concerned with one such application known as the 'image registration' problem.

Image registration in this work is defined as the restoration of an image or a video sequence that is subjected to some form of geometric distortion. One example of the need for image registration is that of improving the robustness of watermarks in image or video data. It is well known that one way of destroying watermark information in an image or a video frame is to apply geometric distortion such as rotation, shear or scaling [6]. A number of approaches have been proposed to restore the image to the pre-distorted form [7] [8]. However, most of the proposed methods employ algorithms that are suited to a general-purpose computer with a floating-point unit. In this paper, a computational efficient algorithm for image registration is proposed. Our algorithm is based on localised motion vectors and is particularly suitable for reconfigurable computers. Its implementation on the SONIC architecture is described, and the effectiveness of the algorithm is examined.

This paper is organised as follows: after a brief description of the SONIC architecture and its latest implementation known as UltraSONIC in Section 2, the image registration problem for watermarking is explained in Section 3. Section 4 describes the proposed algorithm followed by its implementation on UltraSONIC in Section 5. The results of the algorithm are described in Section 6 and Section 7 concludes this paper.

2. The SONIC Architecture

The SONIC platform [4] [5] is a reconfigurable computing system capable of handling the high data throughput and delivering the computational power needed for real-time video applications. Figure 1 depicts a simplified block diagram of the SONIC system. The design consists of Plug-

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 IEEE

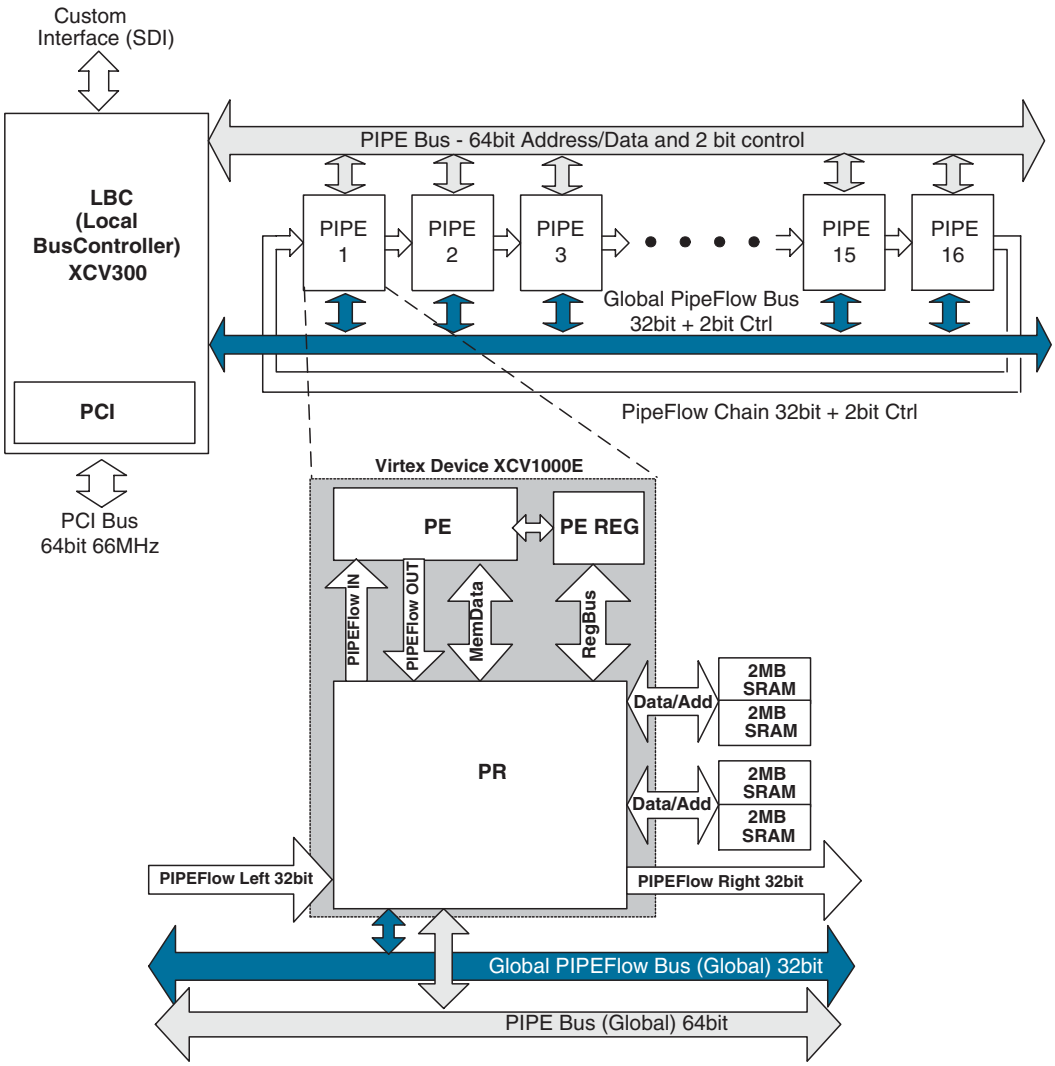

Figure 1. Simplified Block Diagram of the SONIC architecture

In Processing Elements (PIPEs) connected by a PIPE bus and two PIPEflow buses. SONIC's architecture exploits the spatial and temporal parallelism in video image processing algorithms. It also enables design reuse and supports the software plug-in methodology.

SONIC's bus architecture consists of two shared global buses combined with a flexible pipeline bus. The global PIPE bus is used for transferring information between the host system and the PIPEs such as image data, configuration data to the PIPEs' FPGA resources and control of the routing of data on the PIPEflow buses.

Two PIPEflow buses are used to transfer information between PIPEs. The global PIPEflow bus allows broadcast data while the PIPEflow Chain connects adjacent PIPEs to support pipelined operations. The system uses a predefined raster-scan protocol to send data over these buses.

The PIPE consists of three parts: PIPE Engine (PE), PIPE Router (PR) and PIPE Memory (PM). The PIPE engine handles computation while the PIPE router handles image data movement and formatting. The PIPE memory provides local buffers for video data which reduce bus traffic.

The principle of separating the computation engine PE from the routing engine PR has many advantages. It allows computational functions to be implemented independently from dataflow, thus facilitating design reuse. A particular user-application configures only the PIPE engine and not the PIPE router. For example, the same median filter 'hardware routine' on the PIPE engine could be used to process

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 IEEE

video data from the host via the global PIPE bus, from adjacent PIPEs via either PIPEflow buses, or from PIPE memory.

The unique design of the PIPE router provides a flexible and scalable solution to routing and formatting video data. User's application running on the host system controls the PIPE router via a set of well-defined SONIC Application Programming Interface (API) routines. Under software control, the PIPE router can perform three different type of dataflow functions: 1) *data formatting* – such as YC_rC_b to RGB colour space conversion, 2) *data routing* – allows flexible dataflow between the various buses, 3) *data accessing* – provides various data scanning functions such as raster scanning or "stripped access" which is particularly useful for 2D filters and block processing.

An earlier implementation of SONIC was based around a Flex10K100 device for the PIPE engine and a Flex10K50 device for the PIPE router [5]. The latest implementation, UltraSONIC, is designed with a Virtex XCV1000E device implementing both the PIPE engine and the PIPE router. The PIPE memory is implemented with synchronous SRAM, therefore easing interfacing timing problems. Sufficient memory is included to buffer two video frames at HDTV resolution. Interface to the host system is via a 64-bit PCI bus running at 66MHz.

Although the SONIC architecture is designed mainly to use FPGA to implement the PIPEs, the same architecture can be extended to substitute the FPGA with a custom ASIC or a DSP processor. For example, video I/O PIPE or MPEG-2 codec PIPE can easily be designed and incorporated into the system.

3. The Image Registration Problem in Watermarking

Watermarking is the insertion of information (watermark data) into host data without perceptible corruption. This embedded data should only be accessible by authorised parties and a degree of robustness is necessary to withstand possible attacks. Watermarking systems are used for a broad range of applications including copyright protection and metadata embedding.

One method of attack to watermarked image and video is through geometric distortion. For example, most watermark extraction and detection algorithms cannot tolerate rotations, shifts or shear. To overcome such attack, the corrupted image can be passed through an image registration step before watermark detection is performed as shown in Figure 2. Furthermore, the watermarking algorithm can either be blind or non-blind. In blind watermarking, the original image is not required to extract or detect the watermark. In contrast, non-blind watermarking assumes that the original image is available. Figure 3 shows a possible

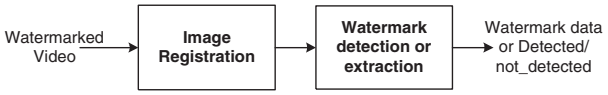

Figure 2. Image Registration in Watermarking

approach to the non-blind image registration problem. The watermarked video sequence W is subjected to a geometric distortion function T to produce a corrupted video sequence C with the watermark information corrupted. Given that the original video sequence O is known, an estimate of the distortion function $\hat{T}$ is made. From this estimate, an inverse distortion function $\hat{T}^{-1}$ is derived. This inverse function is then used to restore the corrupted video to improve the possibility of extracting or detecting the watermark. The restored video R can further be compared with the original video sequence in order to improve the estimated distortion function in a recursive manner.

Since the data throughput of a broadcast quality video sequence is high, it is reasonable to assume that the distortion function T is applied globally over the entire video frame. Furthermore, linear distortions such as rotation, shift, shear and scaling can be formulated as an affine transformation given by (1):

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \qquad (1)$$

where (x, y) represent the original pixel position, (x', y') the position to which the pixel is mapped, and a, b, c, d, e and f are constants for a given distortion function. To perform image registration, these six constants need to be estimated and an inverse function constructed to restore the image.

4. The Solution

Our approach in solving the image registration problem is based on localised motion vectors [9]. A frame from the original video sequence is divided into square blocks. For each reference block a motion vector is computed by finding the best match in the correspondent frame from the corrupted video (see Figure 4). Since the distortion cannot be large otherwise it would be noticeable, only a relatively small region in the vicinity of the reference block needs to be searched. Typically the reference window would be 16×16 pixels while the search window would be 32×32. Figure 5 shows the motion vector map for a global rotational distortion with the centre of rotation at the centre of the frame.

In order to find the relationship between a motion vector (u, v), the coordinate of the centre of an image

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 IEEE

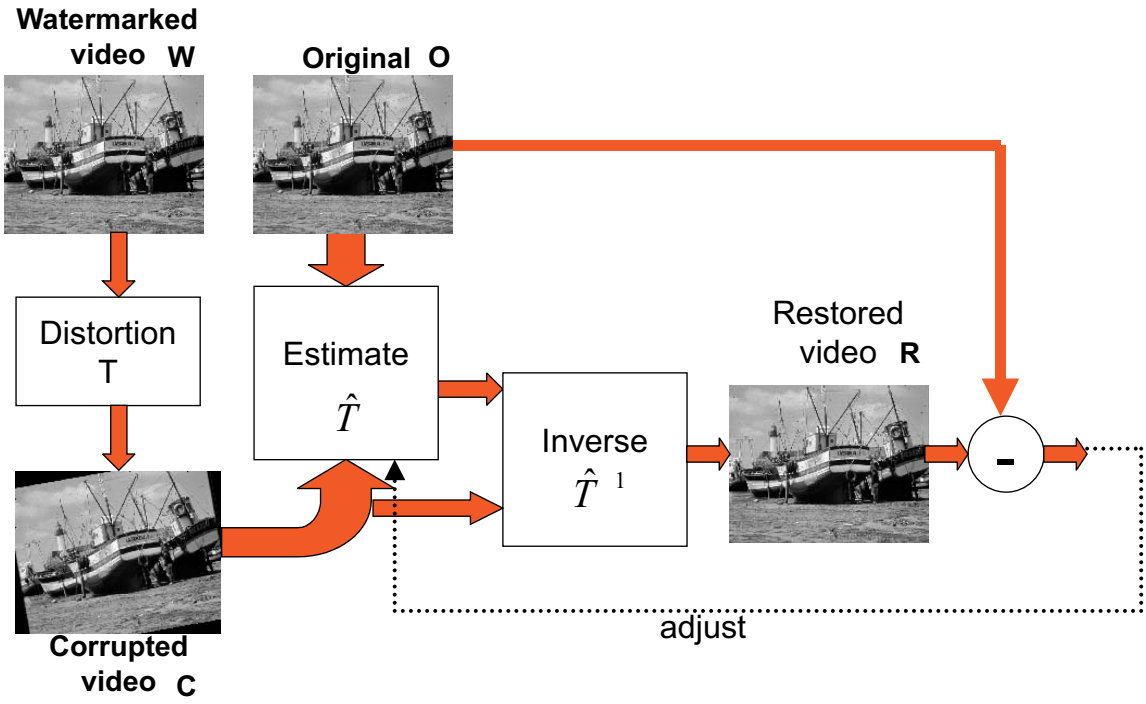

Figure 3. An Approach to Image Registration

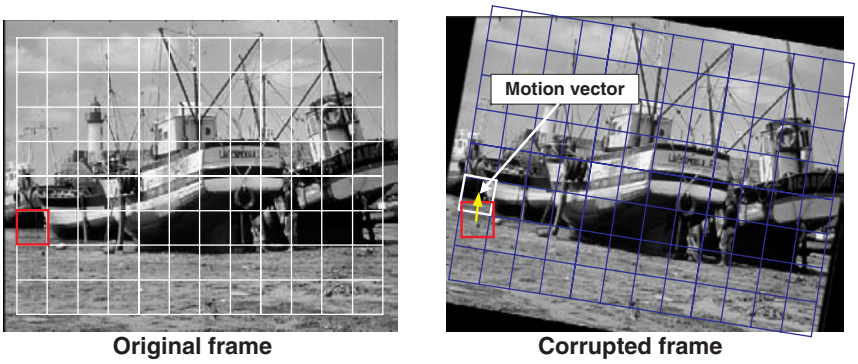

Figure 4. Motion Vector Estimation

block (x, y) and the affine transform distortion parameters (a, b, c, d, e, f), we can arrange (1) as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x' - x \\ y' - y \end{pmatrix} = \begin{pmatrix} a-1 & b \\ c & d-1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \quad (2)$$

(2) can be separated into two orthogonal components for x and y directions:

$$u = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} a-1 \\ b \\ e \end{pmatrix} \quad (3)$$

$$v = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} c \\ d-1 \\ f \end{pmatrix} \quad (4)$$

In theory only three motion vectors are necessary to compute the six distortion parameters. However (3) and (4) are exact only for the case where the distortion is a transla-

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 IEEE

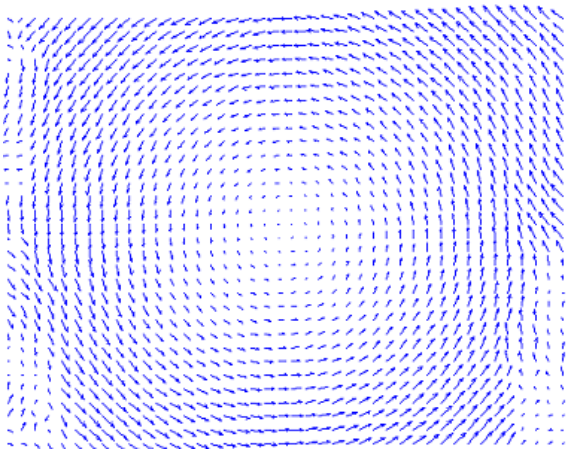

Figure 5. Motion Vector Map with anticlockwise rotation

tion by an integer number of pixels in the x and y directions. It is an approximation for all other cases of affine transform distortion. For example, some of the motion vectors shown in Figure 5 at the corners of the frame where the distance from the centre of rotation is largest are wrong. To obtain the best estimates for (a, b, c, d, e, f), all motion vector information can be combined as:

$$\mathbf{u} = \mathbf{A}\mathbf{p} \tag{5}$$

$$\mathbf{v} = \mathbf{A}\mathbf{q} \tag{6}$$

where,

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{pmatrix}, \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} \tag{7}$$

$$\mathbf{A} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_M & y_M & 1 \end{pmatrix} \tag{8}$$

$$\mathbf{p} = \begin{pmatrix} a-1 \\ b \\ e \end{pmatrix}, \mathbf{q} = \begin{pmatrix} c \\ d-1 \\ f \end{pmatrix} \tag{9}$$

(5) and (6) represent an over-determined system with unknowns $\mathbf{p}$ and $\mathbf{q}$. The least square solutions for $\mathbf{p}$ and $\mathbf{q}$ can be obtained by:

$$\hat{\mathbf{p}} = \begin{pmatrix} \hat{a}-1 \\ \hat{b} \\ \hat{e} \end{pmatrix} = \mathbf{A}^{-1}\mathbf{u} \tag{10}$$

$$\hat{\mathbf{q}} = \begin{pmatrix} \hat{c} \\ \hat{d}-1 \\ \hat{f} \end{pmatrix} = \mathbf{A}^{-1}\mathbf{v} \tag{11}$$

where $\mathbf{A}^{-1}$ is the pseudo-inverse of $\mathbf{A}$ and $(\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f})$ are the least square solutions for the six parameters of the distortion function. Note that $\mathbf{A}$ is a constant matrix with its values $(x_1, x_2, \ldots, x_M)$ and $(y_1, y_2, \ldots, y_M)$ given by the centre coordinates of the reference windows used to compute the motion vectors. Therefore the pseudo-inverse $\mathbf{A}^{-1}$ only needs to be computed once for each reference window size.

Once $(\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f})$ are found, image registration is accomplished by applying the inverse distortion function to the corrupted video according to:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \hat{a} & \hat{b} \\ \hat{c} & \hat{d} \end{pmatrix}^{-1} \begin{pmatrix} x' - \hat{e} \\ y' - \hat{f} \end{pmatrix} \tag{12}$$

where $(\hat{x}, \hat{y})$ and (x', y') are the pixel coordinates in the restored and corrupted video frames respectively. In practical applications, (12) is not applied directly since mapping each pixel in the corrupted frame to a restored frame will in general leave many pixels unfilled. Therefore the restoration process is conducted in a reverse manner: for each pixel in the restored frame, the pixel coordinate in the corrupted frame is computed in order to reverse the distortion process. Linear interpolation is also employed to produce a smooth image.

5. Implementation

The implementation of the image registration system on UltraSONIC consists of three main blocks: 1) motion vectors calculation, 2) distortion parameter estimation and, 3) image registration. The hardware/software partitioning of these three blocks is obvious. Both motion vector computation and image registration are computationally expensive and involve vast amount of dataflow. They are implemented in VHDL for the Virtex resource on the UltraSONIC PIPE. Distortion parameter estimation is only done at most once per video frame. In fact, it is very unlikely that the distortion function changes frequently because this would produce noticeable distortion in the video sequence. This block is therefore implemented on the host system in C++.

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 **IEEE**

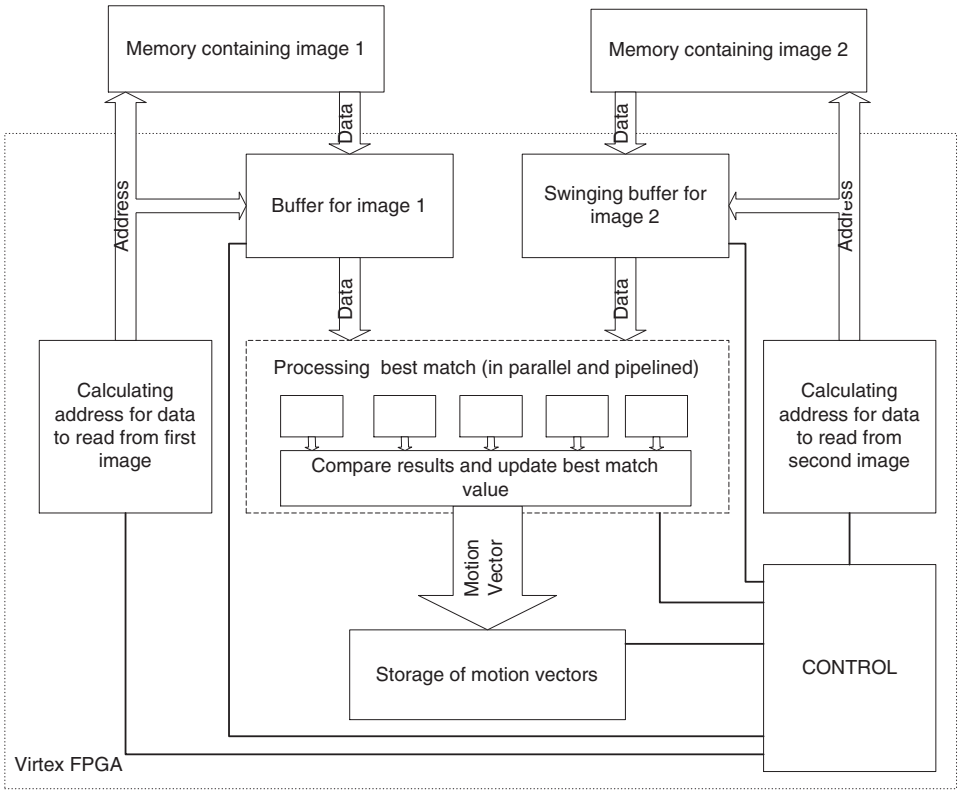

Figure 6. Simplified Block Diagram of the Motion Vector Estimation Engine

5.1 Motion vector estimation

Many full-search motion vector estimation architectures have been proposed. Our implementation is adapted from a design proposed by [11]. This architecture has the advantage of minimizing data flow in such a way that all data are read only once from the external memory that stores the original and the corrupted video frames. This is achieved with extensive data buffering using the large amount of embedded RAM in the Virtex devices. By storing one entire row of blocks from the search frame and prefetching new data into buffer locations as soon as its content is no longer needed, the motion vector estimation engine is implemented with complete overlapping of matching error calculation and data fetch.

Figure 6 shows a block diagram of the motion vector estimation engine. The entire motion vector estimation fits onto one UltraSONIC PIPE.

For a 720×576 video frame, assuming that a reference block size of 16 is used and the search window range is ± 4 to ± 15 pixels, calculating the motion vector map for each frame takes around 7ms with the UltraSONIC PIPE running at 66MHz clock. This is well within the video frame rate of 25Hz for PAL and 29.97Hz for NTSC.

5.2 Distortion Parameter Estimation

Implementation of the solution of (10) and (11) is relatively straight forward. As discussed in the last section, for a given motion vector estimation grid, the value of $\mathbf{A}^{-1}$ can be precalculated. Most of the time taken by distortion parameter estimation is in transferring the motion vector data from the UltraSONIC PIPE to the host computer over the PCI bus. Computation of $(\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}, \hat{f})$ only requires two matrix multiplications given in (10) and (11).

5.3 Image Registration

Due to the use of the reverse mapping combined with the linear interpolation (which uses 4 pixels) and the fact that there are only two memory ports available, there would be at least two clock cycles needed before one pixel is calculated. As the image registration certainly has to run in real-time, only one clock cycle can be allowed for each pixel. Heavy parallelism and pipelining combined with buffering of data are used to achieve this.

Buffering the data for this algorithm is more difficult than for the motion vector estimation. This is because the data to be buffered depend on the nature of the distortion

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 **IEEE**

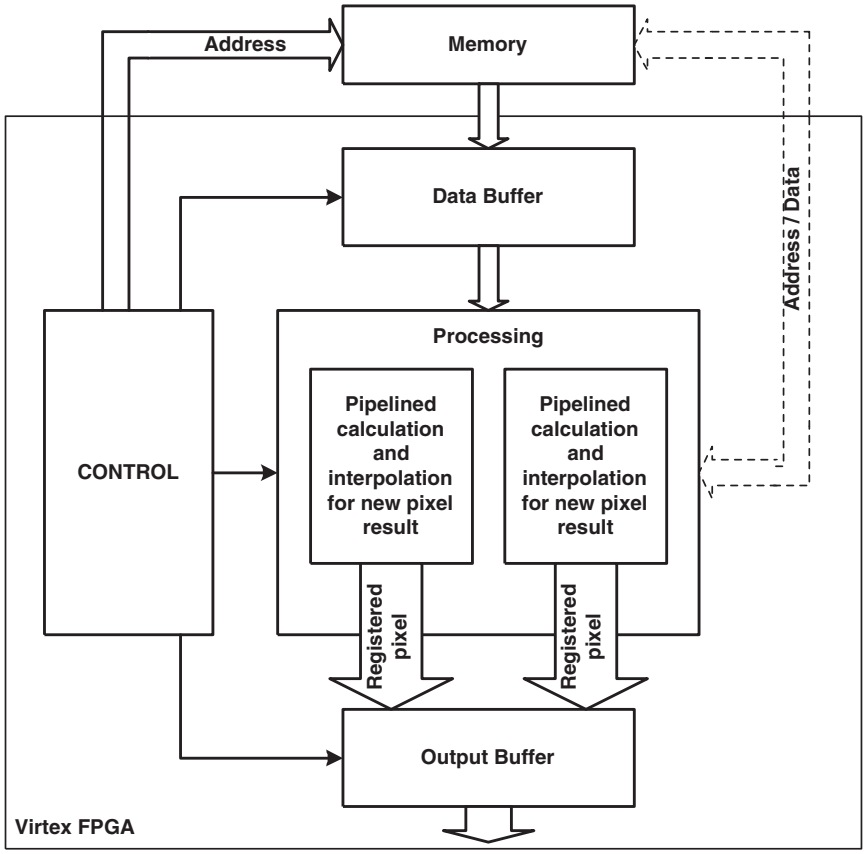

Figure 7. Simplified Block Diagram of the Image Restoration Engine

and the coordinate of the pixel to be computed. Consider the case of a rotation, the area to be buffered when working on the edges is considerably larger than working around the centre. In order to ensure that real-time performance is achieved, the implementation is designed to cope with the following three cases: 1) all required data are in the buffer, 2) required data are not in the buffer, but the output buffer is still sufficiently filled and, 3) the data are not in the buffer and the output buffer is almost empty. The first case is normal operation. In the second case the calculating units are stalled until the data are read from memory. As this introduces some delay, real-time performance is only possible by inserting an output buffer which maintains a continuous video output. In the third case where the output buffer is almost empty, the four-pixel linear interpolator is replaced with a two-pixel interpolator. This reduces the amount of data read and the processing load.

A block diagram of the image registration engine is shown in Figure 7.

6. Results and Evaluations

In order to determine the accuracy of the image registration algorithm, the entire algorithm was also implemented in *matlab*. This implementation was then applied to distorted watermarked images. The watermarked images were distorted either by a watermark attack program Stirmark [12] [13] [14] or by an affine transform program written in *matlab*. Figures 8 and 9 show results of applying the algorithm to a typical video frame with rotational distortion.

Figure 8 is a plot of the estimated angle of rotation against the actual distortion angle used. Perfect restoration occurs if the solution falls on the 45° line. The dotted line shows the results of the algorithm when a search size of ± 20 pixels is used. It falls almost exactly on the 45° line for rotational angles up to 3.5°. Beyond this a larger search size of ± 40 pixels provides much better results (shown as solid line).

Figure 9 is a plot of the distance of the estimated centre of rotation from the mid-point of the image against the actual distance for two angles of rotation. Again the estimated centre lies almost on the perfect solution line at 45°.

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 IEEE

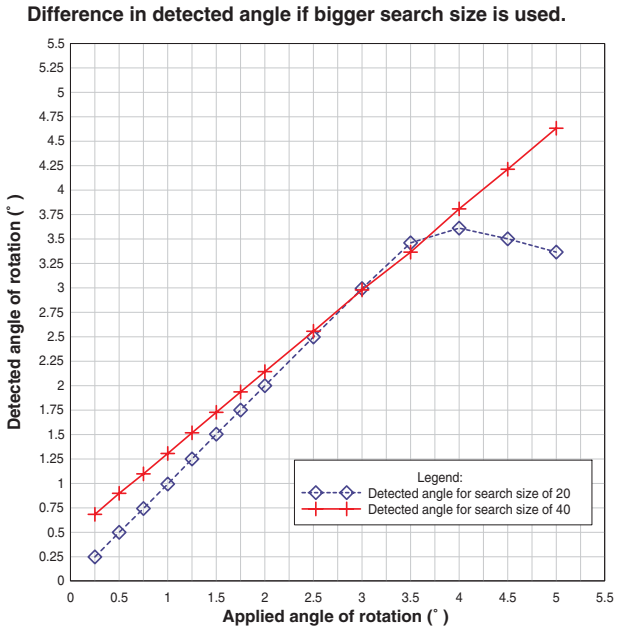

Figure 8. Angular Accuracy after Registration for Rotation Distortion

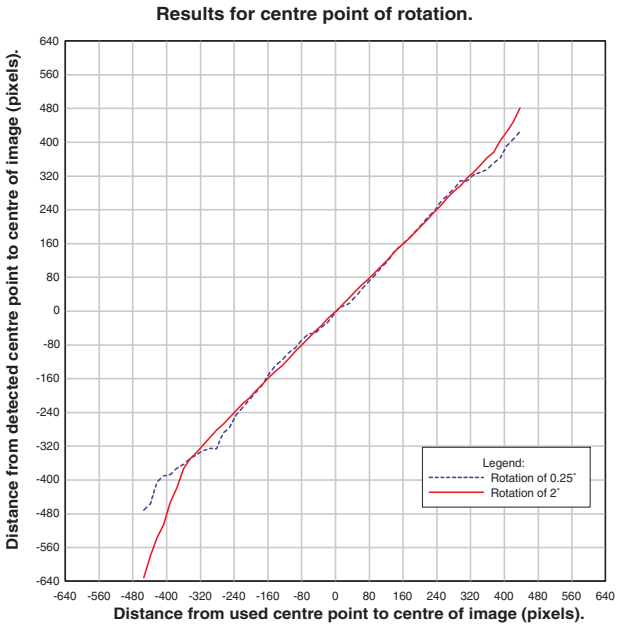

Figure 9. Centre Coordinate Accuracy after Registration for Rotation Distortion

Figure 10 shows the accuracy of the algorithm on shift, shear and rotation distortions applied to two images. The only significant inaccuracy is that of 10% shear. This is due to the small search size used.

Figure 11 shows the effectiveness of the image registration algorithm in watermark detection on corrupt images. Whether a watermark is detected or not is based on a figure of merit known as *sim value* (or similarity value). A *sim value* of 10 or above indicates a watermark has been detected. A *sim value* below 10 indicates that the registration algorithm has not managed to restore the corrupted image in order for the watermark to be detected. As can be seen here, the *sim values* after applying our registration algorithm on all the images tested are well above the threshold of 10.

7. Conclusion and Future Work

An original algorithm, based around motion vectors, for solving the problem of image registration for watermarked video is presented. It is shown to be sufficiently general to handle all global distortions based on affine transformations. The method of calculating the motion vectors and performing the registration can also be applied recursively for improved results.

The hardware implementation of the algorithm allows real-time restoration of the watermarked video data and enables subsequent watermark detection. The hardware implementation minimizes the dataflow by careful on-chip data buffering. Real-time performance is achieved through extensive pipelining and parallelism.

Currently the algorithm only supports global linear affine distortions. The next step is to extend the algorithm to handle distortions that are spatial-variant. This should be possible by dividing the video frame into smaller regions. Provided that the distortion within each region can be approximated by an affine transformation, the algorithm described here should be effective.

Finally the possibility of using this algorithm for applications other than watermarking will be investigated. There are many real-time applications, such as video compression, that involve the calculation of motion vectors, and they can benefit from this algorithm and its implementation on SONIC.

Acknowledgements

The authors would like to thank Jason Pelly, John Stone, Simon Haynes and Henry Epsom from Sony Broadcast and Professional Research Labs for their help and support. We are particularly grateful for Jason's help in verifying our results and the many suggestions he made to improve the paper. We would also like to thank Mike Brookes for his suggestions.

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 **IEEE**

		Shift				Shear				Rotation			
		Applied X-shift (pixels)	Applied Y-shift (pixels)	Detected X-shift (pixels)	Detected Y-shift (pixels)	Applied X-shear (%)	Applied Y-shear (%)	Detected X-shear (%)	Detected Y-shear (%)	Applied rotation angle (°)	Used centre point (Xc,Yc) in pixels	Detected rotation angle (°)	Detected centre point (Xc,Yc) in pixels
Source	Boat	3	0	2.997	-0.011	0.01	0.01	0.0097	0.0100	-0.25	(300, 288)	-0.2593	(295, 284)
		-7	0	-7.026	0.003	0.05	0.05	0.0449	0.0481	-0.25	(100, 150)	-0.2366	(95, 135)
		-5	4	-4.907	4.010	0.10	0.10	0.0534	0.0485	-2	(620, 300)	-1.9915	(619, 277)
	Rail	2	0	2.010	-0.011	0.01	0.01	0.0098	0.0099	-0.25	(300, 288)	-0.2610	(312, 290)
		-6	0	-6.056	0.029	0.05	0.05	0.0450	0.0482	-2	(100, 150)	-1.9856	(98, 147)
		-1	2	-1.001	2.015	0.10	0.10	0.0445	0.0395	-2	(620, 300)	-1.9915	(619, 277)

Figure 10. Accuracy for Estimating Different Types of Distortions

Source	Sim value after registration	Details of applied distortion
Football	28.2	Diagonal shift and 1° rotation clockwise
Icehockey	31.5	Rotation of 1.3° anti-clockwise
Doll	67.3	Scaling of 104 %
Pink	39.8	Diagonal shift and 0.4° rotation clockwise
Yatch	23.6	Rotation of 0.6° anti-clockwise, followed by crop and diagonal shift, followed by 0.3° rotation anti-clockwise and crop.
Mobile	29.3	Enlarging the image by 4%

Figure 11. Results of Watermark Detection after Registration

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 **IEEE**

References

[1] K. Henriss, P. Ruffer, R. Ernst and S. Hasenzahl, "A Reconfigurable Hardware Platform for Digital Real-time Signal Processing in Television Studios," in Proc. 2000 IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), 2000, pp. 285-286.

[2] S. Ludwig, R. Slous and S. Singh, "Implementing PhotoShop filters in Virtex," in: Field-Programmable Logic and Applications (FPL), 1999, pp. 233-242.

[3] A. Simpson, J. Hunter, M. Wylie and D. Mann, "Demonstrating Real-time JPEG Image Compression-Decompression using Standard Component IP Cores on a Programmable Logic based Platform for DSP and Image Processing," in: Field-Programmable Logic and Applications (FPL), 2001, pp. 441-450.

[4] S. D. Haynes and others, "SONIC - A Plug-in Architecture for Video Processing," in Field-Programmable Logic and Applications (FPL), 1999, pp. 21-30.

[5] S.D. Haynes and others, "Video Image Processing with the Sonic Architecture," IEEE Computer, April 2000, pp. 50-57.

[6] Hartung F. and Kutter M., "Multimedia Watermarking Techniques," in: Proceedings of the IEEE, vol. 87, No. 7, pp. 1079-1107, July 1999.

[7] P. Loo and N. Kingsbury, "Motion Estimation based Registration of Geometrically Distorted Images for Watermark Recovery," in Security and Watermarking of Multimedia Contents, part of SPIE Electronic Imaging, Vol. 4314, San Jose, Jan 2001.

[8] N. Kasewkamnerd and Rao K. R., "Wavelet Based Watermarking Detection using Multiresolution Image Registration," in: Proceedings TENCON 2000, vol. 2, pp. 171-175, 2000.

[9] B. G. Haskell, "Digital Video: An Introduction to MPEG-2," New York: Chapman and Hall, 1996, pp. 118-121.

[10] G. L. Brown, "A Survey of Image Registration Techniques," ACM Computing Surveys, Vol. 24 , No. 4, 1992, pp. 325-376.

[11] S. H. Nam, J. S. Baek and M. K. Lee, "Flexible VLSI Architecture of Full Search Motion Estimation for Video Applications," IEEE Trans. Consumer Electron., vol. 40, No. 2, pp. 176-184, May 1994.

[12] F. A. P. Petitcolas and R. J. Anderson, "Evaluation of Copyright Marking Systems," in Proc. ICMCS '99, vol. 1, 1999, pp. 574-579.

[13] F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, "Attacks on Copyright Marking Systems," in Proc. Information Hiding, Second International Workshop, Portland, Oregon, USA, LNCS 1525, Springer-Verlag, 1998, pp. 219-239.

[14] F. A. P. Petitcolas. (2001, June). Stirmark. [Online]. http://www.cl.cam.ac.uk/ fapp2/watermarking/stirmark/

Proceedings of the 10 th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02)
1082-3409/02 $17.00 © 2002 **IEEE**