Hardware Efficient Algorithm for Image Registration of Real-time Video and Image Data

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 image registration as applied to video sequences that have been subjected to geometric distortions. It describes the development of two computationally efficient algorithms to restore broadcast quality video sequences using image registration techniques. The image registration technique is based on motion vectors and found to be successful in restoring the video sequence for either affine transformations or the more general perspective transformations. It is shown that both algorithms can accurately restore the video data. The algorithms can efficiently be implemented on a reconfigurable computer system for real-time video restoration.

1 Introduction

A large number of watermarking algorithms have been presented [20, 21]. However in developing these algorithms there is often the trade off between improving the algorithm so it allows including more data in a secure way and making the algorithm more robust against distortions. Some algorithms present a combination of registration and specially designed watermarking algorithms to achieve higher robustness [17]. This paper is concerned with detecting the applied distortion and registering the image to improve the watermark detection, independent of the used algorithm.

Reconfigurable computing platforms are well known to cope with the high data throughput and computational demand related to real-time video image processing. Therefore there are a large number of successful implementations of image and video applications [9, 12, 18] on this type of platforms. The target platform for this application was developed in cooperation between Imperial College and Sony Broadcast and is called SONIC [6, 7]. Its architecture has been specialised for broadcast video editing, manipulation and processing. The presented algorithm is developed for efficient implementation on this type of reconfigurable systems to allow for real-time performance.

Image registration is in this work defined as the restoration of an image or a video sequence that is subjected to some form of geometric distortion. Although this techniques does know a large amount of applications, in this paper it is used to improve the ability to detect watermarks in image and video data. It is well known that most watermarking techniques are not robust to 2-Dimensional (2D) or 3-Dimensional (3D) geometric distortions such as rotation, shear, shift or perspective transformation [4]. A number of approaches to restore the image have already been proposed [11, 10]. However they employ algorithms that are suited to a general-purpose computer with floating point unit. Other approaches focus on the synchronisation between the distorted video and original video [2] or estimate the distortion making use of feature points [3]. The authors of this paper present two computational efficient algorithm which can be used for either affine or the more general perspective transformation based distortions.

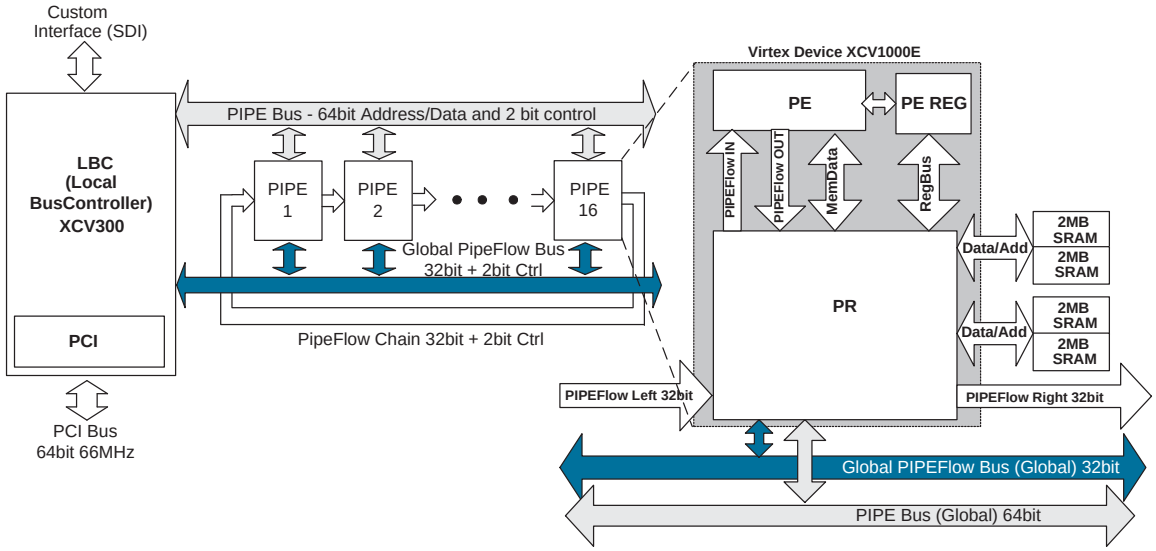


Figure 1. Simplified Block Diagram of the UltraSONIC architecture

The organisation of the paper is as follows: after a brief description of the target architecture, the image registration problem for watermarked data is explained in Section 3. The algorithms to solve this problem are presented in the next sections, after which their implementation is shortly described in Section 5. The results of the algorithms are presented in Section 6, which is followed by the Conclusion (Section 7).

2 The SONIC Architecture

The SONIC reconfigurable computing platform [6, 7] is designed to cope with the high data throughput and deliver the computational power needed for real-time video applications. A simplified block diagram of the latest implementation is shown in Figure 1. The design consists of Plug-In Processing Elements (PIPEs) which are interconnected by a set of buses. The platform exploits as well temporal as spatial parallelism in video processing algorithms. It also facilitates design reuse and supports the software plug-in methodology.

The PIPEs consist of three parts: PIPE Engine (PE), PIPE Router (PR) and PIPE Memory (PM). The PE handles the computation, while the PM is meant to buffer video data as a way to reduce the bus traffic. The PR handles image data movement and formatting. The separation of PR and PE allows for computation functions to be implemented independently from the dataflow. The settings for the PR are controlled by the users application through a set of well-defined SONIC Application Programming Interface (API) routines.

The SONIC architecture knows already two implementations, of which the first one was based around a Flex10K100 for the PE and a Flex10K50 device for the PR. The latest implementation, UltraSONIC [8], is based around Virtex XCV 1000E devices implementing both the PE and PR.

3 The Image Registration Problem in Watermarking

Watermarking is the insertion of information (watermark data) into host data without perceptible corruption. This embedding of data can be done for several reasons, including copyright protection and metadata embedding. Especially in the first case, the embedded data should only be accessible by authorised parties which implies a requirement for a certain level of robustness against attacks.

Applying geometric distortions to the watermarked image and video data is only one way of attacking the watermark data. For example, most watermark extraction and detection algorithms do not tolerate 2D and 3D geometric transformations. To overcome such attack, the corrupted image can be passed through an image registration step before watermark detection is performed. Furthermore, the watermark algorithm can either be blind or non-blind. In blind watermarking, the original image

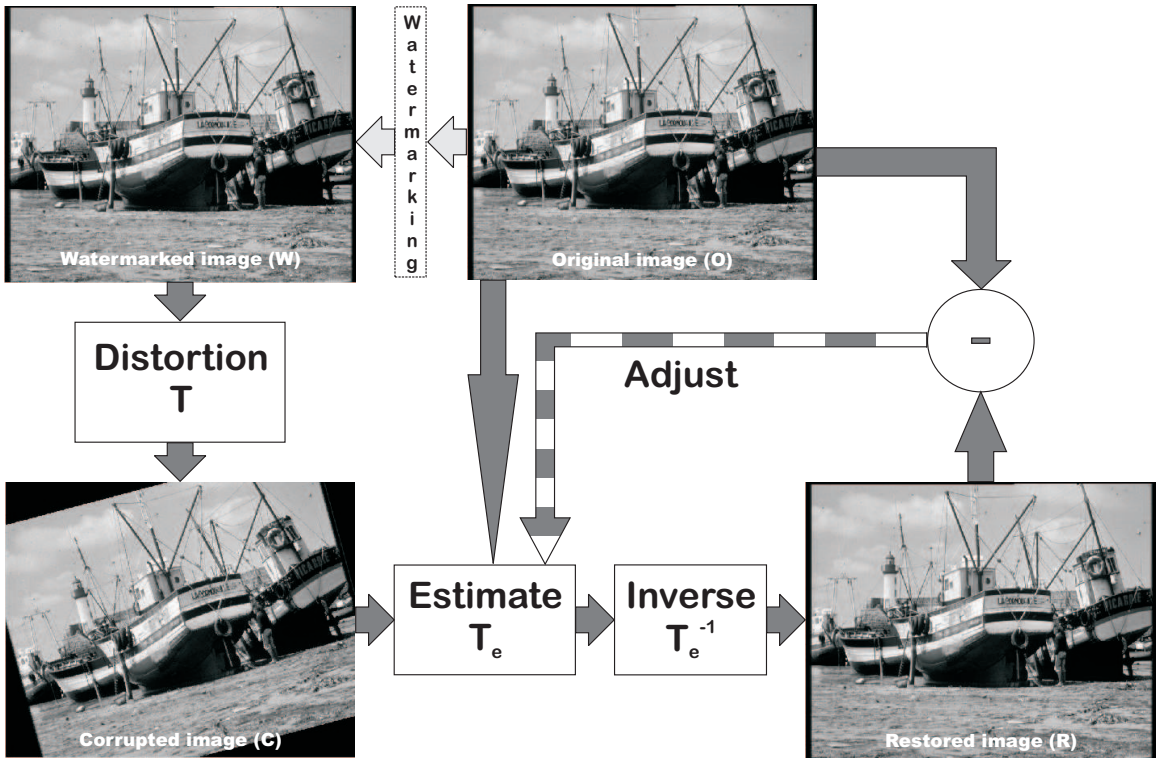


Figure 2. An Approach to Image Registration

is not required to extract or detect the watermark. In contrast, non-blind watermarking assumes that the original image is available.

A possible image registration approach to non-blind watermarking is shown in 2. The watermarked video sequence W is subjected to a geometric distortion function T which produces a corrupted video sequence C with the watermark information corrupted. Making use of the original video sequence O, an estimate of the distortion T_e is made. From this estimate, an inverse distortion function T_e^{-1} is derived and used to restore the corrupted video to improve the possibility of extracting or detecting the watermark. This restored video sequence R can then be further compared with the original sequence O to improve the estimated distortion function in a recursive manner.

Due to the high data throughput of a broadcast quality video sequence, it is reasonable to assume that the distortion is globally applied. Furthermore, any type of linear distortion such as: rotation, shear, shift and scaling can be formulated as an affine transformation using Equation (1). An affine transformation is actually a special case of the more general perspective transformation as presented in Equation (2) (based on [19]).

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

$$\begin{pmatrix} x' & y' & z' \end{pmatrix} = \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} a & c & g \\ b & d & h \\ e & f & i \end{pmatrix} \tag{2}$$

In each of the Equations (1, 2), (x, y)/(x, y, z) represent the position in the original 2D image (where $z = 1$). The new 2D position of the pixel is represented by (x'', y''), whereas (x', y', z') is the position in the 3D space which will need to be converted. The parameters sets $(a, b, \ldots, f)$ or $(a, b, \ldots, i)$ specify the distortion function. These parameters have to be estimated and an inverse function T_e^{-1} can then be used to restore the image. The affine transformation is a special case of the perspective transformation where $z' = z = i = 1$ and $g = h = 0$ in Equation (2).

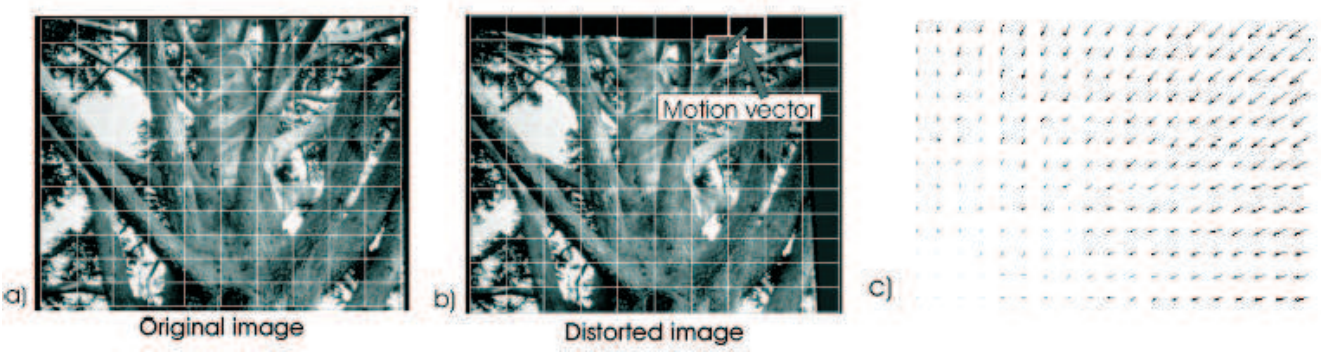


Figure 3. Motion Vector Estimation a) & b) Division into blocks and determined motion vector; c) Motion vector map for perspective distortion of similar type as shown in b).

4 The Solution

The approach in solving the image registration problem is based on localised motion vectors [5]. 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 3 a) & b). Since motion vectors are themselves linear displacements, their usefulness in the current application depends on two factors: 1) the distortion is translational; 2) the distortion is not translational but is sufficiently small that it can be approximated as translational at the block level. The assumption that the distortion is small is not unreasonable, otherwise its effect would become noticeable. In our application, a typical value used for the block size (in pixels) is $16x16$, and a search area of ± 20 pixels is normally sufficient for the expected distortions. This results in a motion vector map, of which an example for a perspective distortion of similar type to the one shown in Figure 3 b) is shown in Figure 3 c).

4.1 Affine Transform

In order to find the relationship between a motion vector (u, v), the coordinate of the centre of an image block (x, y) and the affine transform distortion parameters (a, b, c, d, e, f), Equation (1) can be rearranged to:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a-1 & b \\ c & d-1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{3}$$

which is based on the definition of the motion vectors being:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x'' - x \\ y'' - y \end{pmatrix} \tag{4}$$

This Equation (3) can then be separated into two orthogonal components one for x- and one for the y-direction:

$$u = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} a-1 \\ b \\ e \end{pmatrix} \tag{5}$$

$$v = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} c \\ d-1 \\ f \end{pmatrix} \tag{6}$$

In theory only three motion vectors are necessary to compute the six distortion parameters. However, Equation (5) and (6) are exact only for the case where the distortion is a translation by an integer number of pixels in the x and y directions. It is

an approximation for all other cases of affine transform distortion. Therefore, to obtain the best estimates for (a, b, c, d, e, f), all motion vector information can be combined as:

$$\mathbf{u_{aff}} = \mathbf{A_{aff}}\mathbf{p_{aff}} \tag{7}$$

$$\mathbf{v_{aff}} = \mathbf{A_{aff}}\mathbf{q_{aff}} \tag{8}$$

where,

$$\mathbf{u_{aff}} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_M \end{pmatrix}, \mathbf{v_{aff}} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} \tag{9}$$

$$\mathbf{A_{aff}} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_M & y_M & 1 \end{pmatrix} \tag{10}$$

$$\mathbf{p_{aff}} = \begin{pmatrix} a-1 \\ b \\ e \end{pmatrix}, \mathbf{q_{aff}} = \begin{pmatrix} c \\ d-1 \\ f \end{pmatrix} \tag{11}$$

Equation (7) and (8) represent an over-determined system with unknown $\mathbf{p_{aff}}$ and $\mathbf{q_{aff}}$, for which the least square solutions can be obtained by:

$$\hat{\mathbf{p}}_{\mathbf{aff}} = \begin{pmatrix} \hat{a}-1 \\ \hat{b} \\ \hat{e} \end{pmatrix} = {\mathbf{A_{aff}}}^{-1}\mathbf{u_{aff}} \tag{12}$$

$$\hat{\mathbf{q}}_{\mathbf{aff}} = \begin{pmatrix} \hat{c} \\ \hat{d}-1 \\ \hat{f} \end{pmatrix} = {\mathbf{A_{aff}}}^{-1}\mathbf{v_{aff}} \tag{13}$$

where ${\mathbf{A_{aff}}}^{-1}$ is the pseudo-inverse of $\mathbf{A_{aff}}$ 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 $\mathbf{A_{aff}}$ 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 ${\mathbf{A_{aff}}}^{-1}$ 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:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \hat{a} & \hat{b} \\ \hat{c} & \hat{d} \end{pmatrix}^{-1} \begin{pmatrix} x'' - \hat{e} \\ y'' - \hat{f} \end{pmatrix} \tag{14}$$

where $(\hat{x}, \hat{y})$ and (x'', y'') are the pixel coordinates in the restored and corrupted video frames respectively. In practical applications, Equation (14) 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 further employed to produce a smooth image.

Although this method can efficiently handle affine transformations, it is also possible that a small perspective transformation is applied, which requires a slightly different way to calculate the parameters.

4.2 Perspective Transform

Since perspective transforms are actually 3-Dimensional transformations (see Equation (2)), but only 2-Dimensional images are considered, a conversion is necessary. Combined with the fact that i and z can be set to one (as described in [19]) this leads to:

$$x'' = \frac{x'}{z'} = \frac{ax + by + e}{gx + hy + 1}, \quad y'' = \frac{y'}{z'} = \frac{cx + dy + f}{gx + hy + 1} \tag{15}$$

in which x'' and y'' are the coordinates in the new 2D image.

Rearranging these equations to the following format will make further usage easier:

$$x'' = ax + by + e - gxx'' - hyx'', \quad y'' = cx + dy + f - gxy'' - hyy'' \tag{16}$$

These equations give a relation between the values x and y, for which again the centre of the block is taken, the parameters and the new positions (x'', y''). The new positions are again not known, but can be derived from the motion vectors (u, v) according to the first line of Equation 4. Making use of these equations, the values for x'' and y'' can be calculated and substituted in Equation (16), which allows the construction of a matrix representation with only the parameters (a, b, c, d, e, f, g, h) as unknown. Although in theory four sets of motion vectors and their corresponding block coordinates are sufficient to determine the parameters, it is again better to use all the motion vectors and block coordinates that are available. This results in Equation (17) similarly exploiting the least square method to reduce the effect of the less reliable motion vector values on the overall parameter value estimation.

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1'' & -y_1x_1'' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx_n'' & -y_nx_n'' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1'' & -y_1y_1'' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny_n'' & -y_ny_n'' \end{pmatrix} \begin{pmatrix} a \\ b \\ e \\ c \\ d \\ f \\ g \\ h \end{pmatrix} = A_{persp}p_{persp} = \begin{pmatrix} x_1'' \\ \vdots \\ x_n'' \\ y_1'' \\ \vdots \\ y_n'' \end{pmatrix} \tag{17}$$

The least square solution can be found using:

$$\hat{p}_{persp} = A_{persp}^{-1} \begin{pmatrix} x_1'' & \dots & x_n'' & y_1'' & \dots & y_n'' \end{pmatrix}^T \tag{18}$$

where A_{persp}^{-1} is the pseudo inverse of A_{persp} and $\hat{p}_{persp}$ is the matrix containing the least square solutions of the distortion function parameters. As matrix A_{persp} is in this case dependent on (x_i'', y_i''), which in turn depends on the motion vectors (u_i, v_i), the pseudo inverse has to be calculated for each new set of motion vectors before the parameters can be determined.

Once the parameter values are found, they can be used to perform the image registration making use of the reverse mapping technique, by applying the following equation:

$$\begin{pmatrix} \hat{x} & \hat{y} & \hat{z} \end{pmatrix} = \begin{pmatrix} x' & y' & z' \end{pmatrix} \begin{pmatrix} \hat{a} & \hat{c} & \hat{g} \\ \hat{b} & \hat{d} & \hat{h} \\ \hat{e} & \hat{f} & 1 \end{pmatrix}^{-1} \tag{19}$$

Since $\hat{z}$ is known to be 1, z' can be calculated by substituting Equation (15) into Equation (19) at a known coordinate (x'', y''). Once z' is found, $(\hat{x}, \hat{y})$ can be solved using Equation (15) and Equation (19). Similar to the registration for the affine case, linear interpolation can further be used to produce a smooth image.

5 Implementation

The implementation of the image registration system requires two parts to be implemented on HW to achieve real time performance, these are: 1) motion vectors calculation, and 2) image registration. The parameter calculation can easily

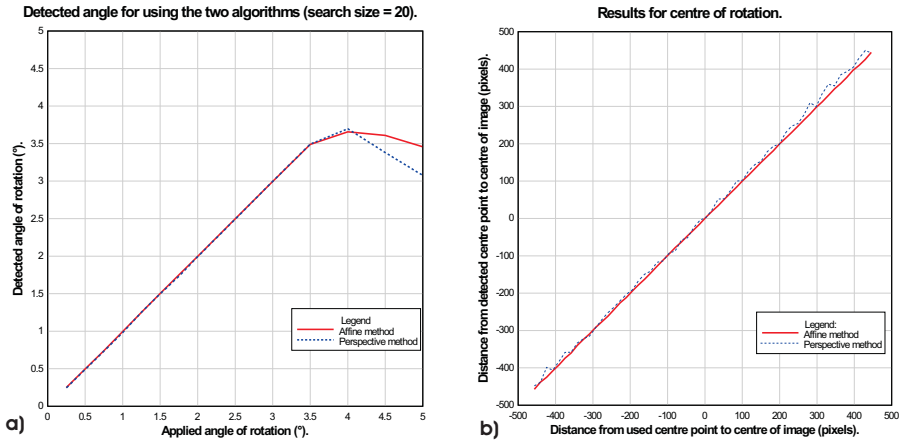


Figure 4. Accuracy after Registration for Rotation Distortion comparing both algorithms

be performed in software since this is only required once per frame. It is also one of the main differences between both algorithms, the other change required is the actual image registration.

Many full-search motion vector estimation architectures have been proposed. Our implementation is adapted from a design proposed by [13]. Generally this algorithm has high computational and data flow demands. Therefore it comes to minimizing the data flow in such a way that all data are read only once from the external memory. This can be achieved with extensive data buffering.

Implementation of determining the parameters according to Equation (12) and (18) is relatively straightforward. When only an affine transformation is applied then $\mathbf{A}_{aff}^{-1}$ can be precalculated. In case of a perspective distortion, this cannot be done since $\mathbf{A}_{persp}^{-1}$ is dependent on the motion vectors themselves. Once the parameters $(a, b, \ldots, f)$ or $(a, b, \ldots, h)$ are determined they can be passed on to the image registration component.

Due to the use of the reverse mapping combined with the linear interpolation (which uses 4 pixels) and the required real time performance the data flow demands are again large. Therefore it is again essential to exploit on-chip data buffering. The registration algorithm is again slightly different depending on whether the full perspective or only affine transformation is performed. However since the latter is a special case of the former, only one implementation is essentially required, and can be adapted according to the requirements.

6 Results and Evaluation

In order to determine the accuracy of the image registration algorithms, the entire algorithm is implemented in *Matlab*. This implementation is then applied to distorted watermarked images. The watermarked images are distorted either by a watermark attack program Stirmark [14, 15, 16] or by a transform program written in *Matlab*. The first part of this result section investigates the accuracy of both algorithms for a rotational transformation. Further accuracy results are then presented for other transformations, including perspective. After which the registration capabilities for watermarked data are presented.

6.1 Accuracy Measurement

Figure 4 indicates the accuracy of detecting the angle and centre point for a rotational distortion. Perfect restoration occurs if the solution falls on the $45°$ line. It can be noticed that both methods have approximately the same accuracy close to this $45°$ line. Beyond rotational angles of $3.5°$ a larger search size of ± 40 pixels provides much better results. The reason for the inaccurate detection especially for larger angles of rotation is related to the fact that for larger rotations the detection of a correct motion vector becomes harder since the block to match has rotated in the distorted image, therefore the motion vector estimation is not highly accurate. If the search area increases then it also becomes more likely that a false positive is detected, resulting in inaccurate motion vectors and parameters.

Table 1. Accuracy for Estimating Parameters of Different Types of Distortions

			a	b	c	d	e	f	g	h
Shear	Rail	Applied	1	0.01	0.01	1	16.00	8.00	-	-
		Detected	1.00	0.01	0.01	1.00	16.05	7.98	-	-
		Applied (1)	1.00	0.05	0.05	1.00	-18.00	-14.40	2.30E-06	6.00E-07
		Detected	1.00	0.05	0.05	1.00	-17.29	-13.56	2.09E-06	1.47E-07
	Boat	Applied	1.00	0.05	0.05	1.00	0.00	0.00	-	-
		Detected	1.00	0.05	0.05	1.00	0.03	-0.05	-	-
		Applied	1.00	0.01	0.01	1.00	2.88	-3.60	3.30E-08	1.60E-08
		Detected	1.00	0.01	0.01	1.00	2.96	-3.59	2.14E-08	-3.06E-07
Shift	Rail	Applied	1.00	0.00	0.00	1.00	2.00	0.00	-	-
		Detected	1.00	0.00	0.00	1.00	2.01	-0.01	-	-
		Applied (2)	1.00	0.00	0.00	1.00	-15.00	3.00	5.80E-09	1.46E-09
		Detected	1.00	0.05	0.05	1.00	-15.00	3.00	4.78E-16	-7.55E-16
	Boat	Applied	1.00	0.00	0.00	1.00	3.00	0.00	-	-
		Detected	1.00	0.00	0.00	1.00	3.00	-0.01	-	-
		Applied	1.00	0.00	0.00	1.00	-6.00	9.00	8.00E-08	4.60E-08
		Detected	1.00	0.00	0.00	1.00	-6.00	9.00	9.65E-16	-4.32E-16
Rotation	Rail	Applied	1.00	-0.01	0.01	1.00	-2.87	3.07	-	-
		Detected	1.00	-0.01	0.01	1.00	-2.90	3.03	-	-
		Applied (3)	1.00	0.01	-0.01	1.00	3.97	-5.86	5.50E-09	8.90E-10
		Detected	1.00	0.01	-0.01	1.00	3.84	-5.93	2.33E-08	-1.86E-07
	Boat	Applied	0.99	0.01	-0.01	0.99	3.97	-5.86	-	-
		Detected	0.99	0.01	-0.01	0.99	4.03	-5.87	-	-
		Applied	1.00	0.01	-0.01	1.00	1.76	-4.36	4.50E-08	3.90E-07
		Detected	1.00	0.01	-0.01	1.00	1.67	-4.33	-1.09E-07	2.19E-07

Figure 4 b) is a plot of the distance of the estimated centre of rotation from the mid-point of the image against the actual distance for a particular angle of rotation using the two different methods. Again the estimated centre lies almost on the perfect solution line at $45°$.

Table 1 shows the accuracy of both algorithms on shift, shear and rotation distortions applied to two images. The results of the affine method can be noticed by the fact that there is no entry for the g and h parameter. In case a perspective transform is applied, then the parameters specific to this perspective transformation are chosen rather small, since any large value gives a very significant modification to the image. These type of transforms would never be applied since they would be too noticeable to the person watching the distorted image. It can be noticed that in general the applied distortion is detected with a rather high accuracy. The small values, related to the perspective distortion (g and h) are often detected with less accuracy which is partly related to the values being very small.

6.2 Registration of Watermarked Images

Although the inaccuracy of some of these detected parameters is not optimal, the main goal is to effectively register the image to allow watermark detection. 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 from Tables 2 which contains data for the affine registration algorithm and Table 3 containing data for the perspective registration algorithm, the *sim values*of all tested images are well above the threshold of 10.

7 Conclusions

Two algorithms, based around motion vectors, to solve the problem of image registration for watermarked video are presented. It is shown they are sufficiently general to handle all global distortions based on affine and/or perspective transformations. The method of calculating the motion vectors and performing the registration can also be applied recursively for

Table 2. Results of Watermark Detection after Registration using the affine method

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

Table 3. Results of Watermark Detection after Registration using the perspective method

Source	Applied Transformation	Sim-value after registration
Tree	Pure perspective transformation (only g and h)	37.9
Basket	Pure affine transformation (g and h = 0)	82.7
Rail	Transformation with parameters of Table 1(1)	30.6
Boat	Transformation with parameters of Table 1(2)	126.1
Susie	Transformation with parameters of Table 1(3)	100.2

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/perspective 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/perspective 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.

References

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

[2] H. Cheng and M.A. Isnardi, "Spatial temporal and histogram video registration for digital watermark detection," in: Proc. International Conference on Image Processing, ICIP 2003, 14-17 Sept. 2003, vol. 3, pp. II-735-738.

[3] P.J.O. Doets, I. Setyawan and R.L. Lagendijk, "Complexity-scalable compensation of geometrical distortions in image watermarking," in: Proc. International Conference on Image Processing, ICIP 2003, 14-17 Sept. 2003, vol. 1, pp. I-513-516.

[4] F. Hartung and M. Kutter, “Multimedia Watermarking Techniques,” in: Proc. of the IEEE, vol. 87, No. 7, pp. 1079-1107, July 1999.

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

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

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

[8] S. D. Haynes, H. G. Epsom, R. J. Cooper and P. l. McAlpine, ” UltraSONIC: A Reconfigurable Architecture for Video Image Processing,” Proc. Field Programmable Logic (FPL) (2002)

[9] 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.

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

[11] 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.

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

[13] 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.

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

[15] 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.

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

[17] C.V. Serdean, M.A. Ambroze, M. Tomlinson and J.G. Wade, ”Adding robustness to geometrical attacks to a wavelet based, blind video watermarking system,” in: Proc. IEEE International Conference on Multimedia and Expo, 26-29 Aug. 2002, vol. 1, pp. 557-560.

[18] 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.

[19] G. Wolberg: Digital Image Warping. Los Alamitos: IEEE Computer Society Press **3rd ed.** (1994) 52-56

[20] M.S. Yasein and P. Agathoklis, ”A wavelet-based blind and readable image watermarking algorithm,” in: Proc. of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 3-6 Nov. 2002, vol.2, pp. 1215-1219.

[21] G. Zhang, S. Wang and Q. Wen, ”An adaptive block-based blind watermarking algorithm,” in: Proc. International Conference on Signal Processing 2004 (ICSP 2004), 31 Aug.-4 Sept. 2004, vol. 3, pp. 2294-2297.