Acceleration of Affine Transform for Multiplane Image Stabilization in Digital Camera

Pulak Mondal, Pradyut Kumar Biswal, and Swapna Banerjee

***Abstract*—Image stabilization is a technology that prevents digital photos becoming blurred. It reduces the effect of camera shake caused by hand movement, slow shutter speeds or when using a long telephoto lens without a tripod. The multiplane image stabilization provided by this high-speed imaging system captures several frames that do not have blur and creates a single image synthesized from the reference image by matching the positions. One of the key steps in this process is multiplane image synthesis which uses affine transform. The transform is applied on multiple images to generate a single image. This paper discusses an algorithm for parallel implementation of the affine transform applicable for an image and also presents the performance results of the implementation of affine transform in FPGA. The result shows that the parallel algorithm runs 4 times faster than the conventional affine transform algorithm and also an image of size approx. 2.07M pixels can be transformed with a frame rate of 65 frames per second.**

***Index Terms*—Affine transform, digital camera, FPGA, multiplane image stabilization, parallel processing.**

I. Introduction

Image stabilization is a technology that prevents digital photos becoming blurred. It reduces the effect of camera shake caused by hand movement, slow shutter speeds or when using a long telephoto lens without a tripod. Although use of a tripod is common in photography with longer exposure times, the multiplane image stabilization method by Sony makes handholding possible in such situations [1].

The multiplane image stabilization provided by this high-speed imaging system captures several frames that do not have blur (for e.g. 8 frames) and creates a single image synthesized from the reference image by matching the positions. To implement this function, the system includes a motion vector detection engine and multiplane synthesis engine. The motion vector detector engine calculates the amount of blur and the multiplane synthesis engine combines multiple images while matching their positions. The affine transformation model is adopted for position matching and can perform rotation correction, which is impossible with optical image stabilization systems such as lens shift and sensor shift methods.

A key step in the multiplane image stabilization is multiplane synthesis which uses affine transform for multiple numbers of planes. The affine transform consists of various operations such as rotation, shearing, scaling and translation [2]. The affine transformation is one of the computationally intensive steps of this process, which involves many number of matrix operations such as matrix multiplication. Therefore, acceleration of this transform is most sought for.

The affine transformation algorithm, in general, is computationally expensive. So, for real time processing, a dedicated hardware involving parallelism may be a good alternative. Image rotation is the most time consuming out of all the operations viz. shearing, scaling and translation. So, most of the previous researchers have put more emphasis on affine rotation and for implementing it, they have adopted non-separable and separable rotation methods. A separable rotation method decomposes the rotation into two or more 1D transformation along the x- and y- directions. There are several decompositions such as two-pass [3],[4] and three-pass algorithms [5]. Though the separable method reduces the use of memory, it introduces errors at each pass and increases complexity. In non-separable implementation, people have designed application specific chip for image rotation using Co-Ordinate Rotation Digital Computer (CORDIC) algorithm [6]-[8]. Though this algorithm is multiplication free, but hardware requirement is more and initial latency for CORDIC is also high. Affine Transform by Removing Multiplications (ATRM) which is also a multiplication free algorithm uses the relationship between two neighboring pixels [9], [10]. However, multipliers are required to calculate the first pixel location. Thereafter, adders are used instead of multipliers to find the transformation. In [11], it has been proposed to implement the affine transform using cellular architecture with a complex system.

The complete architecture design of the affine transform has been described in [6], [11] without detail implementation result. The implementation of only the image rotation has been reported by various authors, which are either CORDIC-based [7], [8], [12] or LUT-based [13], [14]. Two separate LUTs are used in [13] to store sine and cosine values for computation of the pixel position in an image. In [14], an algorithm named Affine transform by pixel replication (ATPR) was proposed which calculates affine transform of two pixel locations of an image simultaneously. In this paper we have proposed a modified ATPR algorithm, which calculates affine transform of four pixel locations of an image simultaneously resulting in faster implementation of the transform.

The rest of the paper is organized as follows. Section II describes the mathematical background of the 2D affine

Manuscript received September 15, 2012; revised October 12, 2012.

P. Mondal is at Indian Institute of Technology, Kharagpur, India (e-mail: mondalpulak.mondal@gmail.com).

P. K. Biswal was in Institute of Technology, Kharagpur, India. He is now in International Institute of Information Technology, Bhubaneswar, India (e-mail: pradyut.biswal@gmail.com).

S. Banerjee is with the at Indian Institute of Technology, Kharagpur, India (e-mail: swapna@ece.iitkgp.ernet.in)

transform. Section III explains the proposed algorithm. The implementation of the proposed algorithm is explained in Section IV. The MATLAB simulation results and the implementation results have been discussed in Section V and Section VI concludes the paper.

II. 2D Affine Transformation

The affine transform consists of four operations i.e. rotation, scaling, shearing and translation as shown in Fig. 1. The generalized 2D affine transform is represented as,

$$\begin{bmatrix} p' \\ q' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \quad (1)$$

where, (p, q) is pixel location the input image and (p', q') is pixel location of the transformed image. The affine transform parameters are represented by, a, b, c, d, t_x and t_y. For implementation purpose, the (1) is modified as,

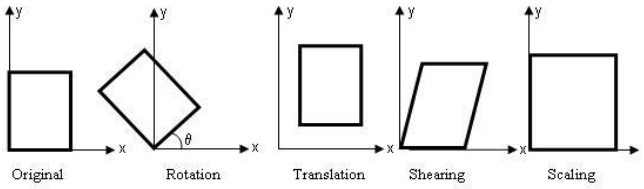
$$\begin{bmatrix} p' \\ q' \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \end{bmatrix} \times \begin{bmatrix} p - x_{cent} \\ q - y_{cent} \\ 1 \end{bmatrix} + \begin{bmatrix} x_{cent} \\ y_{cent} \end{bmatrix} \quad (2)$$

where, x_{cent} and y_{cent} are center coordinates of the image.

Due to shifting of the center of the image, each pixel location will have replicate positions in the image but with the change of sign. In [15], by exploiting the symmetrical nature of the pixel locations, ATPR algorithm is able to calculate transform of 2 pixel locations at atime. Inn this paper, we have proposed a lgorithm named Modified ATPR (MATPR) which can calulate transform of 4 pixel locations from a single transformation operation. The proposed algorithm is explained with mathmatical calculation in the following section.

From an implementation point of view, the affine transform can be implemented either as forward mapping or as backward mapping. In forward mapping, the destination location point is obtained from a particular source pixel location i.e. determining where a particular source pixel will be mapped to. In backward mapping, the source location point is obtained from a particular destination pixel location i.e. determining where a particular destination pixel will come from.

In our implementation, backward mapping is used. In backward mapping, the destination pixel coordinates are integers where as the calculated source pixel coordinates are non-integers. So, the source location point falls between a set of four pixels. Generally, two common interpolation methods are applied to calculate the source pixel value i.e. nearest neighborhood interpolation (NNI) and Bilinear interpolation (BLI). In NNI, the pixel closest to the calculated source pixel is selected. In BLI, linear interpolation is used in both horizontal and vertical directions to select the pixel value. Though BLI gives better visual result, it is computationally complex compared to NNI. So, for less complex implementation, NNI is used in our work



III. Proposed MATPR Algorithm

The proposed algorithm considers symmetry of 4 pixel locations. For the affine transform of pixel location *(x,y)*, (2) can be rewritten as,

$$\begin{aligned} p_1(x, y) &= x_1(x, y) + t_x + offset(1) \\ q_1(x, y) &= y_1(x, y) + t_y + offset(2) \end{aligned} \quad (3)$$

where, $offset(1) = x_{cent} = M/2$, $offset(2) = y_{cent} = N/2$ and

$$\begin{aligned} x_1(x, y) &= a * x + b * y \\ y_1(x, y) &= c * x + d * y \end{aligned} \quad (4)$$

The affine transform of pixel location *(x,y+offset(2))* is:

$$\begin{aligned} p_2(x, y) &= x_1(x, y + offset(2)) + t_x + offset(1) \\ q_2(x, y) &= y_1(x, y + offset(2)) + t_y + offset(2) \end{aligned} \quad (5)$$

where,

$$\begin{aligned} &x_1(x, y + offset(2)) = a * x + b * (y + offset(2)) = \\ &x_1(x, y) + b * offset(2) \\ &y_1(x, y + offset(2)) = \\ &c * x + d * (y + offset(2)) = y_1(x, y) + d * offset(2) \end{aligned}$$

The affine transform of pixel location *(-x,-y)* is:

$$\begin{aligned} p_3(x, y) &= x_1(-x,-y) + t_x + offset(1) = -x_1(x, y) + t_x + offset(1) \\ q_3(x, y) &= y_1(-x,-y) + t_y + offset(2) = -y_1(x, y) + t_y + offset(2) \end{aligned} \quad (6)$$

The affine transform of pixel location *(-x,-y-offset(2))* is:

$$\begin{aligned} p_4(x, y) &= x_1(-x,-y - offset(2)) + t_x + offset(1) \\ q_4(x, y) &= y_1(-x,-y - offset(2)) + t_y + offset(2) \end{aligned} \quad (7)$$

where,

$$\begin{aligned} x_1(-x,-y - offset(2)) &= -x_1(x, y) - b * offset(2) \\ y_1(-x,-y - offset(2)) &= -y_1(x, y) - d * offset(2) \end{aligned}$$

Hence, the affine transform of 4 pixel locations are obtained from the affine transform of only one pixel location but with some extra addition operations. By this way, not only

the computation is reduces by four to get the affine transform of the entire image also for the parallelism and regularity,this algorithm will be a good proposition for VLSI implementation. The algorithm of the proposed MATPR is explined below.

A. Algorithm:

1) Input the image size and transform parameters.

/*M=No. of rows and N=No. of columns. *a*, *b*, *c*, *d*, t_x and t_y are transform parameters*/

2) Calculate offset: $(offset(1), offset(2)) = (M/2, N/2)$

3) Calculate the constant parameters.

$k_1 = offset(2) * b, k2 = offset(2) * d$

$(e_1, e_2) = (Offset(1), Offset(2)) + (t_x, t_y)$

$(d_1, d_2) = (e_1, e_2) + (k_1, k_2)$

$(g_1, g_2) = (e_1, e_2) - (k_1, k_2)$

4. For $x = -\frac{(M-1)}{2}$ to 0 do.

4) For $y = -\frac{(N-1)}{2}$ to 0 do.

5) $x_1 = a * x + b * y, y_1 = c * x + d * y$

6) $(p_1, q_1) = (x_1, y_1) + (e_1, e_2)$

7) $(p_2, q_2) = (x_1, y_1) + (d_1, d_2)$

(p_3, q_3) = 2's complement of $(x_1, y_1) + (e_1, e_2)$,

(p_4, q_4) = 2's complement of $(x_1, y_1) + (g_1, g_2)$, /* (p_1, q_1), (p_2, q_2), (p_3, q_3) and (p_4, q_4) are affine transform of pixel locations (x, y), $(-x, -y)$, $(x, (y + offset(2)))$ and $(-x, -(y + offset(2)))$ respectively.*/

8) If (p_u, q_u) is within the source image then apply interpolation on pixels close to (p_u, q_u), where, u (1,4)

pixvalu = new pixel value after interpolation

$Q[x + offset(2)][y + offset(2)]$ = Pixval1

$Q[x + offset(2)][y + offset(2) + \frac{N}{2}]$ = Pixval2.

$Q[-x + offset(2)][-y + offset(2)]$ = Pixval3.

$Q[-x + offset(2)][-y + offset(2) - \frac{N}{2}]$ = Pixval4

End if.

9) End for

10) End for

IV. Implementation of the Algorithm

The MATPR algorithm explained in Section III has been implemented in FPGA. The architectural block diagram is shown in Fig. 2. According to the size of the input image, the coordinate generator unit generates value of *x* and *y* which is connected as input to the coordinate transform unit. The coordinate generator unit consists of a cascaded counter. The input control unit generates the transform parameters according to the '*opn*' signal. When the '*opn*' signal is high, then rotation operation is performed and the parameters are obtained from the LUT. The cosine and sine values for all angles are stored in the LUT in same address location. When the '*opn*' signal is low, all other operations such as shearing, scaling and translation is performed. In this case, the input control unit passes the parameters which are given in the input. The affine transformation as mentioned in (4) is performed by the coordinate transformation unit. The transformed coordinates may fall outside the image. The coordinate checker unit checks the coordinate value and if it falls outside the range of the image, that particular coordinate is ignored. When the transformed coordinate is a valid one, the address unit generates the address of the memory considering both the *x*- and *y*- coordinates. Data from corresponding memory locations are interpolated using NNI to get the final pixel values for the coordinates generated by the coordinate transformation unit.

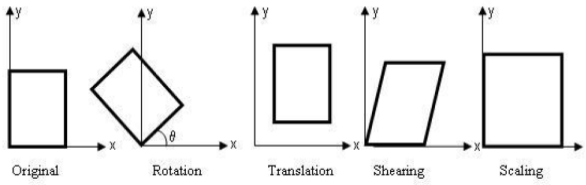


Fig. 1. Basic affine transform operations

V. Results

The conventional affine transformation, the ATPR algorithm [15] and the proposed MATPR algorithm has been implemented using MATLAB. The size of the test images with 256 grayscale values are 128×128, 256×256 and 512×512 and 1920×1080. For hardware implementation, the algorithms are coded in Verilog HDL language and mapped into FPGA.

A. Simulation Result

The Fig. 3(a) shows the original image and Fig. 3(b-c) shows the affine rotation operation using conventional and proposed MATPR algorithm respectively. The Fig. 3(d-f) shows shearing, scaling and scaling with translation operations respectively. The sample image is a cameraman image of size 256x256. From the result it can be observed that the image quality remains same using the proposed algorithm.

The Table I compares the number of matrix multiplication required to perform any affine operation using different algorithms for various image sizes. It is evident from Table I that the MATPR algorithm requires 75% less no. of matrix multiplication compared to the conventional algorithm and computation is reduced by 50% compared to ATPR algorithm.

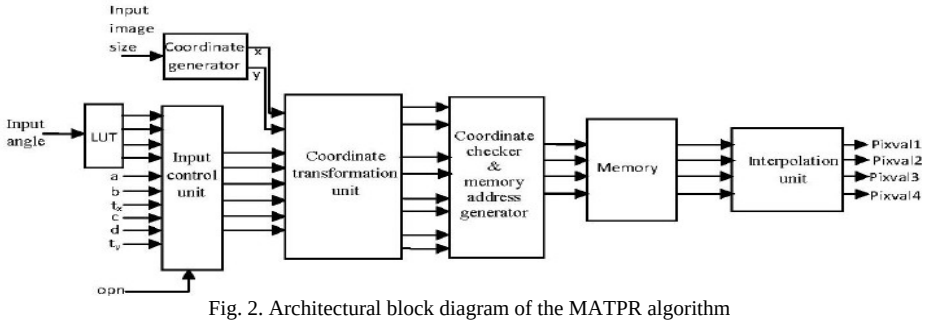
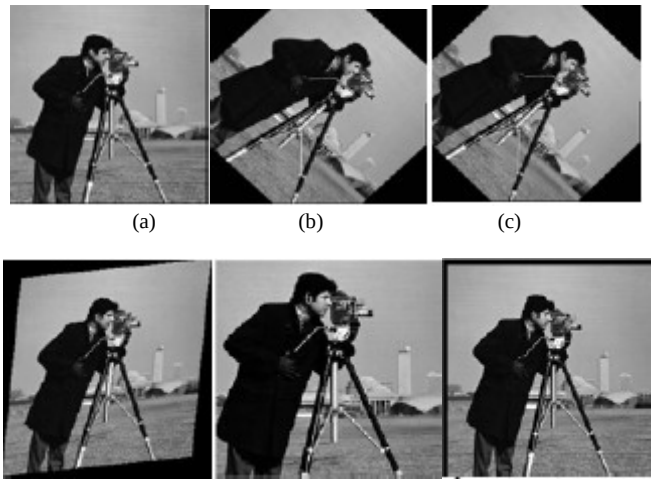


Fig. 2. Architectural block diagram of the MATPR algorithm

TABLE I: NO. OF MATRIX MULTIPLICATION FOR ANY OPERATION

Image Size	Conventional Algorithm	ATPR algorithm	MATPR Algorithm
128×128	16384	8192	4096
256×256	65536	32768	16384
512×512	262144	131072	65536
1920×1080	2073600	1036800	518400



(a) (b) (c)

(d) (e) (f)

Fig. 3. (a). Original Image, (b) Rotation by 45^0 using conventional algorithm, (c) Rotation by 45^0 using MATPR algorithm, (d) Shearing, (e) Scaling, (f) Translation.

A. Implementation Results

The algorithm is synthesized using Xilinx ISE Simulator and is mapped into FPGA. The Table II compares the total execution time of the affine transform. It can be seen from the table that the execution time using MATPR algorithm is much faster compared to other implementations due to exploiting the inherent parallelism and pipeline implementation. The hardware resources for the algorithm has been obtained after post, place and routing of the design in the Xilinx FPGA**.**

TABLE II: COMPARISON OF THE TIMING FOR THE AFFINE TRANSFORMATION

References	Algorithm	Image size	Time (ms)
[3]	Two-pass	512×512	10
[7]	CORDIC	512×512	25
[9]	Conventional	600×600	15
[12]	CORDIC	256×256	14.7
[14]	ATPR	256×256	0.138
Proposed	MATPR	256×256	0.058

TABLE III: HARDWARE COMPARISON

Logic Utilization	ATPR	MATPR
4 input LUT	200	1714
Occupied slices	245	1093
Block RAM	32	32
Maximum Frequency(MHz)	321.590	280.796
Total time(in ms)	3.23	1.80
Device: XC2VP30FG676-6		

Table III gives the comparison of hardware resources used considering NNI during the implementation of ATPR algorithm and MATPR algorithm. It can be found from the Table that though the hardware resources are more in proposed algorithm compared to ATPR algorithm but for an image size of 1920×1080, total execution time is almost halved.

VI. CONCLUSION

In this paper, an algorithm named MATPR is proposed for implementation of affine transform of an image. In advanced digital camera, multiplane image stabilization process has to be performed on a still image of size approx 2.07 MPixels with 40 Frames per second. Affine transform is one of the important operation for implementation of the process. In this paper, we have presented a fast algorithm for implementation of affine transform. The affine transformation model is adopted for position matching of 8 consecutive frames and rotation operation is performed on those images to create a single image. Using the proposed algorithm, this operation is performed within 15ms with a speed of 66 frames per second which is well within the targeted speed.

ACKNOWLEDGMENT

The authors would like to thank Ministry of HRD, Govt. of India for supporting this research work.

REFERENCES

[1] URL. [Online]. Available: www.sony.net/Products/SC-HP/cx_news/vol55/pdf/ featuring55

[2] S. Ghali, "Introduction to Geometric Computing," Springer Verlag, London, 2008.

[3] N. Tsuchida, Y. Yamda, and M. Ueda, "Hardware for Image Rotationby Twice Skew Transformations," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. Assp-35, no. 4, April, 1987.

[4] D. Fraser, "Comparison at high spatial frequencies of two-pass and one-pass geometric transformation algorithms," *Computer Vision, Graphics and Image Processing*, vol. 46, pp. 267–283, 1989.

[5] M. Unser, P. Thevenaz, and L. Yaroslavsky, "Convolution based Interpolation for Fast, High quality Rotation of Images," I*EEE Trans. on Image Proc*, vol. 10, no. 4, pp. 1371-1381, Oct. 1995.

[6] K. Maharatna and S. Banerjee, "Cordic based array architecture for affine transformation of images," in *International Conference on Communications, Computers and Devices*, 2000.

[7] I. Ghosh and B. Majumdar, "VLSI implementation of an efficient ASIC architecture for real time rotation of digital images," *International Journal of Pattern Recognition Artificial Intelligence*, vol. 9, pp. 449-462, 1995.

[8] S. Suchitra, S. K. Lam, C. T. Clarke, and T. Srikanthan, "Accelerating rotation of high-resolution images," *IEE Proc.-Vis. Image Signal Process*, vol. 1953, no. 6, pp. 815-824, December 2006.

[9] W. Badawy and M. Bayoumi, "A multiplication –free algorithm and a parellel architecture for affine transformation," *Journal of VLSI Signal Processing*, vol. 31, pp. 173–184, 2002.

[10] S. Lee, G. Lee, E. S. Jang, and W. Y. Kim, "Fast affine transform for real-time machine vision applications," in *ICIC 2006, Berlin*, 2006, Lecture Notes in Computer Science 4113, pp. 1180–1190, Springer.

[11] G. P. Biswas, I. Dutta, P. Krishna, and I. Sengupta, "Cellular architecture for affine transforms on Raster images," *IEE Proc. Computer Digit. Tech*., vol. 143, no. 2, pp.103-110, March 1996.

[12] X. G. Jiang, J. Y. Zhou, J. H. Shi, and H. H. Chen, "FPGA Implementation of Image Rotation Using Modified Compensated CORDI,'' *6th International Conference On ASIC*, ASICON 2005.

[13] S. M. Bhandarkar and H. Yu, "A VLSI implementation of real time image rotation," in *Proc. Int. Conf. on Image Processing*, vol.2, pp. 1015-1018, 1996.

[14] P. K. Biswal and S. Banerjee, "An Embedded Solution of 2D Fast Affine Transform for Biomedical Imaging Systems," in *Proceedings of VLSI Design and Test Symposium (VDAT 2009)*, Bangalore, India, July, 2009, pp. 259 – 270.

Pulak Mondal was born on 1987 April 5 at west Bengal, India. He completed his B. Tech degree (2009) from the Department of Electronics and Communication Engineering in Jalpaiguri Government Engineering College, India and M.Tech (2011) from the Department of Electronics & Electrical Communication Engineering at Indian Institute of Technology, Kharagpur, India. His research interest includes development of parallel algorithm for signal and image processing and and hardware implementations. Mr. Mondal is Presently persuing his Ph.D degree at Indian Institute of Technology, Kharagpur, India.

Pradyut Kumar Biswal obtained his B.E. and M. Tech from Institution of Engineer's (India) and Visvesvaraya National Institute of Technology, Nagpur, India in 2000 and 2002 respectively. In 2012, he has obtained his Ph.D. degree from the Department of Electronics & Electrical Communication Engineering at Indian Institute of Technology, Kharagpur, India. He has teaching experience of around 5 years. His research interest includes Digital Image processing, Architecture design for image processing algorithms and FPGA implementations. Mr. Biswal is presently positioned as an Assistant Professor in International Institute of Information Technology, Bhubaneswar, India. He teaches Microprocessor, Digital Signal Processing, VLSI Design

Prof. Swapna Banerjee was born on 1949 December 12 at West Bengal, India. She completed her B.E. degree (1971) and M.E. (1974) from the Department of Electronics and Telecommunication Engineering in Jadavpur University, India. She was awarded Ph.D. degree in 1981 from the Department of Electronics & Electrical Communication Engineering at Indian Institute of Technology, Kharagpur, India. She was awarded Post Doctoral fellowship from Tokyo University, Japan in 1985. In 1985 she joined as the lecturer in the Department of Electronics & Electrical Communication Engineering, at Indian Institute of Technology, Kharagpur. She has conducted a number of courses and seminars. To her credit she has a number of publications in reputed international and national journals and conferences on various fields. Her research interests spans around analog and digital VLSI design, biomedical instrumentation, device modeling and digital signal / image processing. Prof. Banerjee is Presently professor in Department of Electronics & Electrical Communication Engineering at Indian Institute of Technology, Kharagpur, India. She is is a Member of IEEE, USA.