MODIFIED FULL SEARCH BLOCK MATCHING ALGORITHM

Madhuri Bamankar
bamankarmadhuri@gmail.com

P.Muralidhar
pmurali@nitw.ac.in

C.B.Ramarao
cbrr@nitw.ac.in

National Institute of Technology, Warangal-506004, India

***Abstract*: Full search block matching algorithm (FSBMA) is most popular motion estimation algorithm. But high computational complexity is the major challenge of FSBM. This makes FSBM to be very difficult to use for real time video processing with the low power batteries. The proposed algorithm i.e. modified full search block matching algorithm (MFSBMA) reduces the computational complexity by keeping the PSNR same as of FSBMA. MFSBMA skips the SAD calculations for a current background microblock and it does SAD calculations for foreground current microblock. This method reduces SAD calculations drastically. The proposed architecture of MFSBMA is pipelined architecture which can work on real time video processing. The proposed algorithm reduces computational complexity by 50% by keeping PSNR same with the tolerance of +3% to -3%.**

***Keywords*: Full Search Block Matching algorithm, Block matching algorithm and Motion Estimation, H.264/AVC**

I. INTRODUCTION

In H.264 video compression standard motion estimation (ME) is major block. Block Matching algorithms (BMA) are used for ME [11]. BMA works on temporal and spatial redundancies present in a video. In block matching algorithm each frame of video is divided into NxN blocks. The motion of each block from current frame to reference frame is represented by Motion Vectors (MV). Motion vector gives the movement of NxN block from reference frame to current frame.

II. MOTION ESTIMATION ALGORITHMS

In Video compression standards Motion Estimation (ES) block uses block matching algorithms such as full search block matching (FSBM), Three step search (TSS), four step search (FSS),etc. FSBM gives best PSNR among all mentioned algorithms.

FSBM is an exhaustive search of NxN current frame block in previous frame search region. Apart from advantage of good quality video compression in FSBM it requires large number of computations. This makes FSBM very difficult for hardware implementation.

Another block matching algorithm is Three step search (TSS) algorithm [1]. It selects centre of search region and calculates sum of absolute difference (SAD) for locations which are 4 positions apart from centre. Thus, it calculates SAD at 8 locations in first step. In next step, it repeats the procedure by reducing the step size by 2. In third step, it reduces step size by 1. This reduces number of computations but it is prone to missing small motions.

Four step search (FSS) is same as TSS. Instead of 3 steps 4 steps are required in FSS. First stage of FSS requires 5x5 window instead of 9x9 window in TSS [2].

Diamond search method selects center and calculates SAD at 9 locations in first stage. The 8 points around center makes shape of diamond. In next stage 5 checkpoints around center makes diamond shape. At every stage it converges to minimum SAD checkpoint and next step proceeds from SADmin checkpoint.

These algorithms suffer from irregular data flow and does not guarantee about the optimum solution.

The proposed algorithm separates the background pixels from foreground pixels. It calculates SAD for foreground blocks only. As in video frame minimum 50% area is background area, so number of SAD calculations required reduces drastically. SAD is block matching criteria to find best match for current microblock in search region.

There are many block matching criteria such as, Mean squared error (MSE), Sum of absolute difference (SAD), etc. SAD is computationally more simpler than MSE.

$$\text{SAD}(S, C(m)) = \sum_{x=1}^{M} \sum_{y=1}^{N} |C(x,y) - R(x - mx, y - my)| \quad (1)$$

Where, C(x,y) current block pixel

$R(x\text{-}m_x, y\text{-}m_y)$ reference block pixel.

For every reference block in search region SAD is calculated with same current block. Minimum SAD block defines the best match of current block.

The rest of the paper is arranged as follows. Section III explains about background elimination algorithms. Section IV defines proposed algorithm and section VI explains architecture design of proposed algorithm. Simulation results are given in section V and Conclusion in section VIII.

III BACKGROUND ELIMINATION ALGORITHMS

In any video movement of object from one frame to next frame is very small and rest of the background remains same. There is no need of calculating motion vectors (MVs) for

background blocks. background elimination algorithm segregates background blocks from foreground blocks.

Background subtraction is used in many emerging video applications, such as video surveillance, traffic monitoring, and gesture recognition for human-machine interfaces. There are many background elimination methods present [6]. First method is frame differencing which has less complexity. Second is approximate median method which has medium complexity and third has high complexity mixture of Gaussian method [6].

In Frame differencing the current frame is subtracted from the previous frame, and if the difference is greater than a threshold T_s, the pixel is considered part of the foreground.

$$|\text{frame}_i - \text{frame}_{i-1}| > T \quad (2)$$

A challenge with this method is determining the threshold value. In median filtering, the previous *N* frames of video are buffered, and the background is calculated as the median of buffered frames. Main disadvantage is storage requirement. It has a little trouble with quickly changing light levels, but handled them better than mixture of Gaussians.

This paper uses frame differencing method as, main aim of this paper is to reduce computational complexity.

IV PROPOSED ALGORITHM: FAST FSBM

We designed the proposed motion estimation algorithm to decrease the computation complexity of the full search algorithm and also to tracking one object. First we read a frame after a frame from the video.

The algorithms works as follow:

1. Read first frame name it as reference frame (RF).
2. Read second frame name it as current frame (CF).
3. Take the difference of CF and RF, it gives background frame (BF).
4. Threshold (T) each background pixel value i.e. if BF pixel value is less than threshold T then replace CF pixel to zero. It means that pixel is background pixel.
5. If BF pixel is greater than T, it means it is foreground pixel.
6. Calculate motion vectors for foreground pixels.
7. Repeat same procedure for all frames.

In a video, frame to frame pixel movement is very less. Background area is than foreground area. This reduces the number of computations required for full search block matching algorithms significantly. The threshold value will be different for different type of video.

V SIMULATION RESULTS

The proposed algorithm is simulated in MATLAB. The experiments were conducted on five different video clips. The video clips used are Coastguard (qcif) 176x 144 300 frames, akiyo (qcif), foreman (qcif), mobile (qcif), Highway (cif) 352x288.

The quality of recovered video clips was verified by calculating PSNR. Computational complexity (CC) was also calculated for every video.

$$\text{PSNR}(I_t, I_{t+1}) = 10 \log(255^2 / \text{MSE}) \quad (3)$$

$$\text{MSE}((I_t, I_{t+1})) = (1/MN) \sum_{m=1}^{M} \sum_{n=1}^{N} I_t(m,n) - I_{t-1}(m,n) \quad (4)$$

The Fast full search block matching algorithm (FFSBMA) results are compared with Full search block matching algorithm (FSBMA) results. Comparison shows that computational complexity of FFSBMA is 50% less than FSBM, while PSNR is almost same.

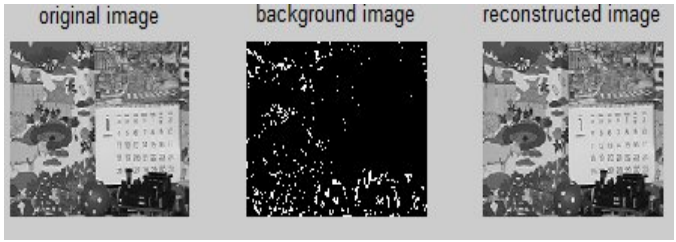

Fig.1 Mobile

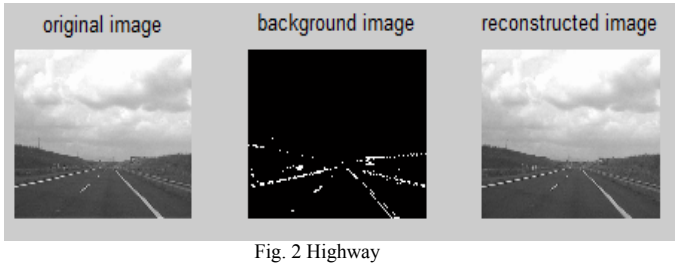

Fig. 2 Highway

Fig. 3 Foreman

For mobile fig(1) video input qcif (172 x 144) threshold was 10. For Foreman fig(3) video input qcif of size (172 x144) threshold was 25. For third input video highway cif fig(2) of size (352x 288) threshold was 25.

TABLE 1: PSNR for FS an MFS method

Method	PSNR				
	Video Input				
	Coastguard	Akiyo	Foreman	Mobile	Highway
FS	30.24	38.04	30.5	24.82	33.82
MFS	29.22	40.63	29.8	26.81	30.7

TABLE 2: CC for FS an MFS methods

Method	Computational Complexity				
	Video Input				
	Coastguard	Akiyo	Foreman	Mobile	Highway
FS	184.5	184.5	184.5	184.5	184.55
MFS	66.58	19.36	73.07	115.38	14.54

VI. ARCHITECTURE

The main blocks of architecture are Background elimination, Processing element array (PEA), Adder tree and motion vector calculation (MV). The block diagram of FFSBMA is given below.

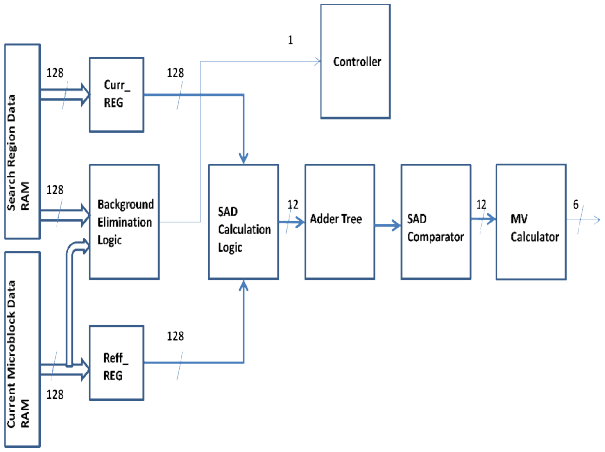

Fig. 4: Functional block diagram of FFSBMA

A. Processing element array (PEA)

PEA is array of 16 PEs, which are processing parallel. Four PEs (PE0, PE4, PE8, PE12) give one 4x4 SAD calculation. All 16 PEs give four 4x4 SADs. Current RAM gives four pixels at a time. Input current RAM data is constant for four clock cycles. Input Reference data is constant for 3 clock cycles.

Current data flows horizontally through PEA. PE0, PE4, PE8, PE12 forward current data to PE1, PE5, PE9, PE13 respectively and so on. Reference data flows vertically i.e from PE1 to PE4, PE2 to PE5, PE3 to PE6 and so on. PE7, PE11, PE15 require extra 3 reference pixels. It achieves 100% PE utilization by employing a preload register and a search data buffer.

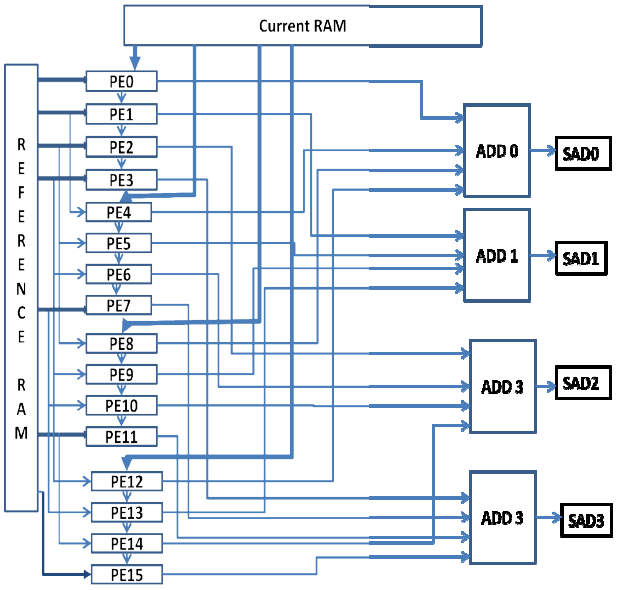

Fig.5 2D systolic array of Processing element array

Such 16 PEAS should be connected in pipelined manner. The vertical data of one PEA comes from horizontal data of above PEA. Top most PEAs get vertical data from reference RAM. 16 PEAs work on 4 microblock SAD calculation in pipelined manner.

B. Processing Element (PE)

A PE plays a vital role in processing three different tasks in this design. PE computes the absolute difference between the CMD and the SRD. It Propagates the CMD and SRD values to the next PE (PCMD and PSRD). Finally the PE sums up the difference values (DV) of the current and the previous data.

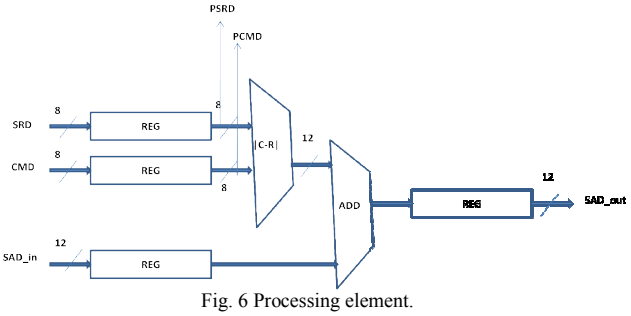

Fig. 6 Processing element.

VII. SYNTHESIS REPORT

This proposed architecture is synthesized on Xilinx ISE 14.2 simulator. Device used for synthesis is (family Vertex 6) xc6vlx75t-3ff484. The source utilization is given below.

Slice Logic Utilization:

Number of Slice Registers:	13265	out of 93120	14%
Number of Slice LUTs:	17354	out of 46560	37%
Number used as Logic:	13130	out of 46560	28%
Number used as Memory:	4224	out of 16720	25%
Number used as SRL:	4224		

Slice Logic Distribution:

Number of LUT Flip Flop pairs used: 23519
Number with an unused Flip Flop: 10254 out of 23519 43%
Number with an unused LUT: 6165 out of 23519 26%
Number of fully used LUT-FF pairs: 7100 out of 23519 30%
Clock period: 7.496ns (frequency: 133.396MHz)

VIII. CONCLUSION

Modified full search block matching algorithm (FFSBMA) reduced the computation complexity significantly compare to full search block matching algorithm. Number of SAD calculations has reduced by applying background elimination method. This gives the moved block and removes the constant block in current frame. FFSBMA had achieved the goal by keeping the quality of video same.

REFERENCES

[1]. C. Thou-Ho, "A cost-effective three-step hierarchical search block matching chip for motion estimation," *Solid-State Circuits, IEEE Journal of,* vol. 33, pp. 1253-1258, 1998.

[2]. Lai-Man Po ; Wing-Chung Ma "A novel four-step search algorithm for fast blockmotion estimation." Circuits and Systems for Video Technology, IEEE Transactions on Volume:6, Digital Object Identifier: 10.1109/76.499840 Publication Year: 1996 , Page(s): 313 – 317.

[3]. Yeong-Kang and C. Lien-Fei, "A performance-driven configurable motion estimator for full-search block-matching algorithm," in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, 2004, pp. II-233-6 Vol.2.

[4]. M. Panovic and A. Demosthenous, "A low power block-matching analog motion estimation processor," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 4827-4830 Vol. 5.

[5]. M. Ghanbari and I. o. E. Engineers, *Standard Codecs: Image Compression to Advanced Video Coding*: Institution of Electrical Engineers, 2003.

[6]. Seth Benton , Background subtraction, part1: MATLABmodels, http://www.eetimes.com/design/militaryerospace design/4017685/Background subtraction-part-1- MATLAB-models.

[7]. N.-n. Sun, C. Fan, and X. Xia, "An effective three-step search algorithm for motion estimation," in *IT in Medicine & Education, 2009. ITIME '09. IEEE International Symposium on*, 2009, pp. 400-403.

[8]. W. Tsung-Yi, C. Kuang-Yao, H. Shi-Yi, L. Tai-Lun, and L. How-Rern, "A VLSI design with built-in SRAM arrays for implementing Full Search Block Matching Algorithm," in *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, 2009, pp. 619-621.

[9]. L. Kexiang, Q. Qingshun, and Z. Zhenzhong, "A novel fast motion estimation algorithm based on block-matching," in *Cross Strait Quad- Regional Radio Science and Wireless Technology -Conference (CSQRWC), 2011*, 2011, pp. 1402-1405.

[10]. "An Improved Full Search Block Matching Algorithm for Imaging Applications". An International Conference on Computer and Communication Engineering (ICCCE) 3-5 july-2012, Malaysia.

[11]. Performance comparison of the emerging H.264video coding standard with the existing standards Kamaci, N. ; Altunbasak, Y. Multimedia and Expo, 2003. ICME '03. Proceedings.2003 International Conference on Volume:1Digital Object Identifier: 10.1109/ICME.2003.1220925 Publication Year: 2003 , Page(s): I - 345-8 vol.1