Optimal Circular 2-D Search Algorithm for Motion Estimation

Siddhartha Ahluwalia, Dr.Anupam Shukla, and Sourabh Rungta.

***Abstract*—An dynamic method for estimating displacement of motion vector within small blocks with minimum mean of absolute difference is presented. An efficient algorithm for searching the direction of displacement has been described. The motion compensation is applied for analysis and design of a hybrid coding scheme and the results show a factor of two gain at low bit rates. The algorithm uses circular 2-D Logarithmic search [CLS] technique for best prediction of minimum distortion block. The search area is continuously reduced by a factor of 2 each time search for best motion vector takes up a new direction with the number of points to be searched for each iteration, decreasing continuously than the first iteration. Experimental results prove the proposed CLS Algorithm has a speed gain of more than 30% over diamond search [DS] algorithm for finding large motion vectors. The approach has been successfully tested under for standard sets of 6 video sequences. An illustrious comparison with parameters such as increase in SIR (%) has been drawn between our algorithm and DS algorithm.**

***Index Terms*— Motion Estimation, MAD.**

I. Introduction

Compressing video efficiently and dynamically has been the foremost target of research in areas of video processing.
The motion estimation block in a video codec computes the displacement between the current frame and a stored past frame that is used as the reference. Usually the immediate past frame is considered to be the reference. More recent video coding standards, such as the H.264 offer flexibility in selecting the references frames and their combinations can be chosen.

We consider a pixel belonging to the current frame, in association with its neighbourhood as the candidates and then determine its best matching position in the references frame. The difference in position between the candidates and its match in the reference frame is defined as the displacement vector or more commonly, the motion vector. It is called a vector since it has both horizontal and vertical components of displacement.

Manuscript received December 12, 2009
Siddhartha Ahluwalia, M.Tech , Department of Information Technology, ABV-IIITM Gwalior, India-474010
Email:siddhartha.iiitm@gmail.com
Dr.Anupam Shukla, Asso. Professor, Department of Information Technology, ABV-IIITM Gwalior, India-474010
Email: dranupamshukla@gmail.com
Sourabh Rungta, Reader, CSE Dept, RCET Bhilai, C.G
Email: sourabh@rungta.org

After determining the motion vectors one can predict the current frame by applying the displacements corresponding to the motion vectors on the reference frame. This is the role of the motion compensation unit. The motion looked if corresponding displacements were applied at different regions of the reference frame.
The basic procedure involves coding the initial frame and then tracking the trajectories traversed by the various objects. Through this ample magnitude of compression is achieved. This can be done only in the case of inter-frame image coding. The first frame of any video sequence is assumed to be intra-coded, rest of the following frames are mostly inter-coded. In inter-coding of frames trajectory information of each pixel is tracked and coded along the initial frame which is intra-coded. The candidates frame is divided into non-overlapping blocks (of size 16 x 16, or 8 x8 or even 4 x 4 pixels in the recent standards) and for each such candidate block, the best motion vector is determined in the reference frame.

Here, a single motion vector is computed for the entire block, whereby we make an inherent assumption that the entire block undergoes translational motion. This assumption is reasonably valid, except for the object boundaries and smaller block size leads to better motion estimation and compression. Here thus the elimination of temporal redundancy eliminates between successive frames improves encoding efficiency greatly.

Block based motion estimation is accepted in all the video coding standards proposed till date. It is easy to implement in hardware and real time motion estimation and prediction is possible. Based on the study of search patterns used in many fast Block Motion algorithms, we propose a fast Block Motion algorithm, which is based on Circular 2-D Logarithmic Search Algorithm [CLSA]. The two parameters used for comparing the CLSA with other previously developed algorithms are two algorithms are MAD (Mean of Absolute Difference) and SIR (Speed Improvement Rate)

A. Block Matching Motion Estimation

Block matching motion estimation is the cardinal process for many motion-compensated video coding standards, in which temporal redundancy between successive frames are effectively removed. To implement the block motion estimation, the candidate video frame is partitioned into a set of non overlapping blocks and the motion vector is to be determined for each such candidate block with respect to the reference.

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

For each of these criteria, square block of size N x N pixels is considered. The intensity value of the pixel at coordinate (n_1, n_2) in the frame–k is given by $S(n_1, n_2, k)$, where $0 \leq n1, n2 \leq N-1$. The frame–k is referred to as the candidate frame and the block of pixels defined above is the candidates block. With these definitions, we now define the matching criteria i.e. MAD.

B. MAD

Considering $(k-l)$ as the past references frame $(l > 0)$ for backward motion estimation, the mean of absolute difference a block of pixels computed at a displacement (i, j) in the reference frame is given by

$$MAD = \frac{1}{N^2}\left(\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} |[s(n_1, n_2, k) - s(n_1, n_2, k-l)]|\right)$$

The mean of absolute difference (MAD) makes the error values as positive as absolute differences are summed up.
The physical significance of the above equation should be well understood. We consider a block of pixels of size N x N in the reference frame, at a displacement of (i, j) , where i and j are integers with respect to the candidate block position. The mean of absolute difference (MAD) too makes the error values as positive as the absolute differences are summed up. The MAD is computed for each displacement position (i, j) within a specified search range in the reference image and the displacement that gives the minimum value of MAD is the displacement vector which is more commonly known as motion vector and is given by

$$[d_1, d_2] = \arg\min_{i,j}[MAD(i, j)]$$

The MAD criterion requires computations of N^2 subtractions with absolute values and N^2additions for each candidate block at each search position. The absence of multiplications makes this criterion computationally more attractive and facilitates easier hardware implementation.

C. Basic Method

In one of the first techniques Full Search Block Motion Estimation we consider a block of N x N pixels from the candidates frame at the coordinate position let it be (r, s) We then consider a search window having a range $\pm p$ in both n_1 and n_2 directions in the references frame .

For each of the $(2p+1)^2$ search position (including the current row and the current column of the reference frame), the candidate block is compared with a block of size N × N pixels, according to one of the matching criteria and the best matching block, along with the motion vector is determined only after all the $(2p+1)^2$ search position are exhaustively explored. Here *p* is the maximum displacement position considered in either direction in integer number of pixels

The FSBM is optimal in the sense that if the search range is correctly defined, it is guaranteed to determine the best matching position. However, it is highly computational intensive. For each matching position, we require $O(N^2)$ computations.

We therefore conclude that FSBM requires large number of computations. Say, we take $p = 7$ pixels, measuring that the best matching position exists within a displacement of ± 7 pixels from the current block position; we require 15 × 15=225 search positions. For real time implementation, quick and efficient search strategies were explored.
It may be noted that such quick search techniques do not make exhaustive search within the search within the search area and can at best be sub-optimal.

II. LITERATURE REVIEW

Many computationally efficient motion estimation algorithms [1–11] have been developed, typically among which are three-step search (TSS) [3], new three-step search(NTSS) [6], four-step search (4SS) [7], block-based gradient descent search (BBGDS) [8] ,diamond search (DS) [9,10] algorithms, octagon based search[2] and hexagon based search[11]. In TSS, NTSS, 4SS and BBGDS algorithms, square-shaped search patterns of different sizes are employed. The search pattern can further be classified into cross"+" pattern and "X" pattern. The earlier is used by TDLs (Two –Dimensional Logarithmic Search)[1] and combination of both is used by CSA(Cross Search Algorithm)[5] and DSWA(Dynamic Search-Window Adjustment).
The DS algorithm adopts a diamond-shaped search pattern, which has demonstrated faster processing with similar distortion in comparison with all square based search patterns. The search pattern has a prominent impact on speed and distortion in block motion estimation. Square shaped search patterns used to be standards but some time back for fast block motion estimation DS became benchmark. But new approaches like Octagon and Hexagon based search pattern proved more efficient than DS.

The proposed reason for disadvantage of above mentioned DS is that diamond shape is not approximate enough to a circle. The advantages of hexagon and octagon based approach over DS is that they are able to generate a search pattern with a uniform distribution of a minimum number of search points and hence tends to achieve faster search speed uniformly. The only common property between DS and Octagon algorithms is search through 9 points in first iteration.

III. ALGORITHM

It has been proposed by *Jain and Jain[1]* that as we move along in any direction and away from the direction of minimum distortion(DMD) the distortion function monotonically increases.

Step 1: Draw a circle with (0, 0) as centre as we assume it to be point of minimum distortion.

Step 2: List all the pixels which lie at the periphery of the circle.

Step 3: List the pixels with following attributes:
Let the pixel be (i, j).
Condition 1: If $(i^2 + j^2) < p^2$

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Condition 2: (i, j) lies within the perimeter of the circle and distance of (i, j) from the nearest neighbouring pixel which has been selected in step 2 is greater than p / 2.
(If step 2 & step 3 has been iterated more than 1 number of times then previous pixels which had already been listed in step 2 and 3 are not considered).

Step 4: If our MAD point (the centre of circle) is still the point of minimum distortion move to step 6 otherwise move to step 5.

Step 5: The point with minimum distortion is taken to be centre of new circle with same radius and go to step 2.

Step 6: If p = 1 goto step 7 otherwise p = p / 2 and go to step 2.

Step 7: Find (i, j) such that D (i , j) is minimum. The (i, j) now computed is DMD

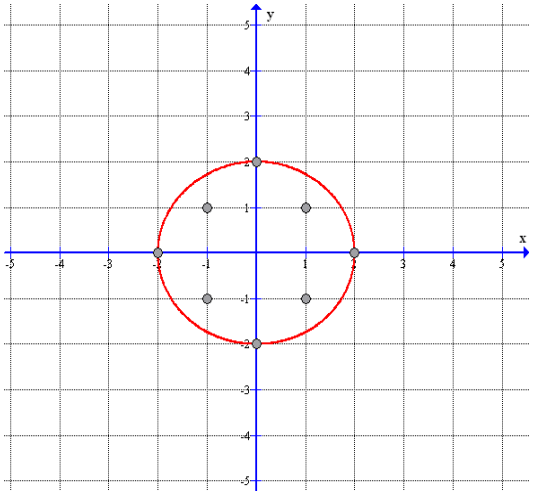

Fig. 1 Dotted pixels to be considered as points of minimum distortion.

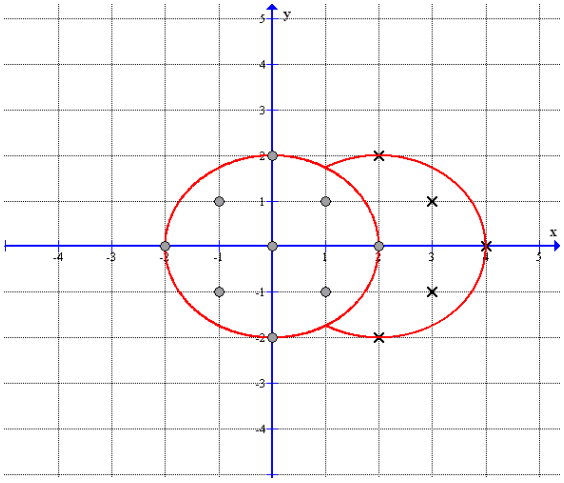

Fig. 2 Crossed pixels indicate pixels to be considered for 2nd iteration

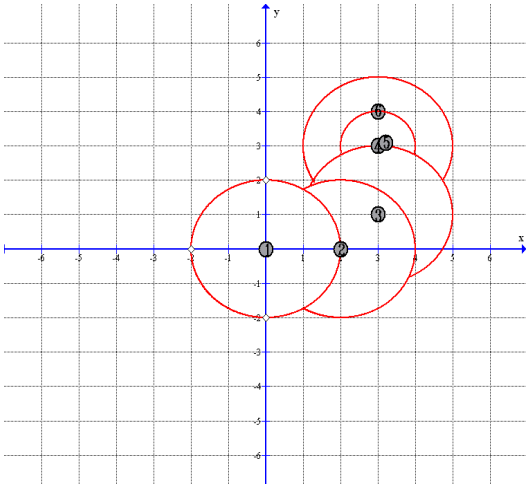

Fig. 3 Circles 1-6 demonstrate movement of point of minimum distortion in each iteration

IV. ANALYSIS OF ALGORITHM

The total number of search points per block will be
N_{CLSA} (fx, fy) = 9 + Qn + 4 (1)
The value of Q can be 3 or 4 depending upon the search pattern recognized.
Here n is the number of times step 2 of the algorithm has been iterated.
Here we are improving our efficiency as well as computational complexity from the Diamond Search (D.S) algorithm as for DS algorithm

$N_{DS} = 9 + Cn_d + 4.$ (1)

Where *C* is either 5 or 3 and n_d is always greater than n in (1). The saving of search points does not affect distortion. The increase in MAD is given by

$$MAD(\%) = \frac{(MAD_{CLSA} - MAD_{DS})}{MAD_{DS}}$$

$$SIR = \frac{(9 + Cn_d + 4) - (9 + Qn + 4)}{(9 + Qn + 4)} \times 100\%$$

V. EXPERIMENTAL RESULTS

The experimental set up is as follows: the distortion measurement of MAD is used. Block size is 16 × 16. Six standard video sequences are used, which varies in motion content as well as frame size. The six video sequence are "Claire", "Dance wolf", "Football", "Suzie", "Salesman" & "Tennis" which cover a wide range of motion contents and various formats(QCIF & CIF). We use SAD as the objective function.
The test condition for simulation is tabulated in Table 4, where each sequence has the first 100 frames in simulation. Each sequence is coded using IPP... structure, that is, the first frame is coded as I frame, and all the remaining frames are coded as P frames. The frame rate is 30 frames per second.
The experiment is conducted with JM86 encoder to evaluate the performance of the proposed CLSA. The search is performed within a square window of size [-16, +16] around

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

the current block position. The number of reference frames is five, and the number of block types is seven.

Average Search points and average MAD values are summarized in tables 3 and 4 respectively for DS and CLSA. Note that only the search region inside the image boundary is considered consistently for fast algorithms tested to make a fair comparison.

The comparison is mainly done between DS and proposed CLSA in terms of number of search points as well as MAD. We can see that our CLSA algorithm consumes lesser number of search points than DS algorithm. There is also a significant reduction in MAD at the cost of visibly increasing number of search points.

According to Tables 3 and 4, Table 5 tabulates the average SIR and average MAD increase in percentage of the proposed CLSA over DS. For "Claire" sequence with motion vectors limited within a small region around (0, 0), our proposed CLSA achieves 3.5% speed improvement over DS. For "Dance Wolf" sequence with medium motion, the average SIR of CLSA over DS is 5.4%. For "Football" and "Suzie", which contain large motion, as proposed in our literature, our CLSA has obtained high speed improvement over DS, more than 13.2% and 18.11% respectively.

As we see for "Salesman" and "Tennis" the improvement in speed is more, which is more than 20%. The larger the motion in a video sequence, the larger the speed improvement rate of CLSA over DS or the other fast algorithms will be. On the other hand, the degradation is trivial, less than 5.1% or smaller of MAD increase for all the video sequences in our experiment.

The search window size of ±16 was also used for the comparison of DS and the CLSA because there is no restriction of window size in the two algorithms. From the observations of Tables 3 and 4 we conclude with the larger window size, the SIR of CLSA over DS increases while the MAD significantly decreases for the large-motion video sequences. For "Salesman" and "Tennis", the speed improvement rates of CLSA over DS are as high as 20.1% and 21.7% respectively. Consequently, all the experimental analysis demonstrates the faster performance of CLSA over DS algorithm.

Table.1 Video Sequences used and their properties

	Resolution	Frames
"Claire"	352 X 240	300
"Dance Wolf"	720 X 480	299
"Football"	720 X 480	59
"Suzie"	720 X 480	70
"Salesman"	352 X 288	448
"Tennis"	720 X 480	39

Table.2 Test Conditions for Simulation

Code version	JM86
Profile	Baseline
RD Optimisation	On
entropy coding	UVLC
GOP structure	IPPP
encoding frames	100
reference frames	5
frame rate	30
block type	All
search range	33 X 33
rate control	Off
CPU, GHz	3.0
RAM, GB	4

Table.3 Average number of search points per block with respect to different video sequences for DS and CLSA.

	DS	CLSA
Claire	12.4	12.29
Dance Wolf	22.56	22.15
Football	17.378	16.98
Suzie	13.120	13.01
Salesman	12.77	12.40
Tennis	18.1	17.70

Table.4 Comparison of Average MAD per pixel for DS and CLSA

	DS	CLSA
Claire	6.101	6.111
Dance Wolf	2.404	2.441
Football	7.409	7.780
Suzie	2.311	2.404
Salesman	2.801	2.805
Tennis	7.546	7.656

Table.5 Average SIR and Average MAD increase in percentage of our CLSA over DS

	Avg. SIR (%)	Avg. MAD increase (%)
Claire	3.5	0.163
DanceWolf	5.4	1.53
Football	13.2	5.01
Suzie	18.11	4.02
Salesman	20.1	0.14
Tennis	21.7	1.45

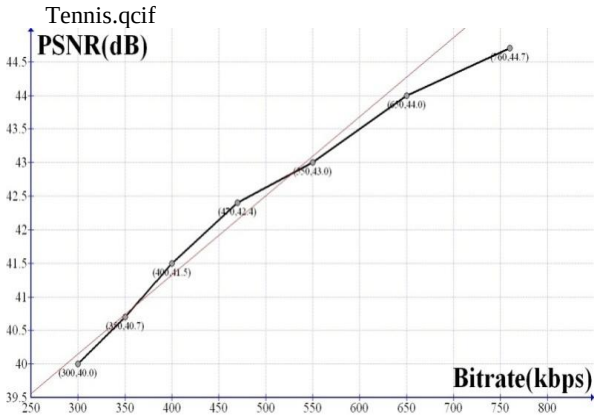

Fig. 4 Rate-distortion performance for CLSA

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

VI. CONCLUSION

The circular 2-D logarithmic motion estimation algorithm is an overall competitive approach over previous approaches such as Diamond search and other efficient block motion estimation search algorithm.

This algorithm is suited for both large and short video sequences and does not depend upon extent of motion. But the larger the motion vector, the more efficient our proposed method would becomes, which is verified by experimental results.

.

REFERENCES

[1] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Commun.*, vol. -29, pp. 1799–1808, Dec. 1981.

[2] Lap-Pui Chau and CeZhu,"A fast octagon-based search algorithm for motion estimation",*Signal Processing* 83 (2003) 671 – 675

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf*., New Orleans, LA, Nov. 29–Dec. 3 1981, pp. G5.3.1–5.3.5.

[4] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun*., vol. COMM-33, pp. 888–896, Aug. 1985

[5] M. Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Commun*., vol. 38, pp. 950–953, July 1990..

[6] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol*., vol. 4, pp. 438–442, Aug. 1994.

[7] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol*.,vol. 6, pp. 313–317, June 1996.

[8] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst.Video Technol*., vol. 6, pp. 419–423, Aug. 1996.

[9] J.Y. Tham, S. Ranganath, M. Ranganath, A.A. Kassim, "A novel unrestricted centerbiased diamond search algorithm for block motion estimation", *IEEE Trans. Circuits Systems VideoTechnol*. 8 (August 1998) 369–377.

[10] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast blockmatching motion estimation," in Proc. *Int. Conf. Inform.*, Commun.,Signal Process., Singapore, Sept. 9–12, 1997, pp. 292–296.

[11] S. Zhu, "Fast motion estimation algorithms for video coding," M.S. thesis, School Electron. Eng., Nanyang Technol. Univ., Singapore, 1998. (supervised by K.-K. Ma).

[12] Ce Zhu, Xiao Lin, and Lap-Lui Chau,"Hexagon based Search Pattern for Fast Block Motion Estimation", *IEEE Trans. on circuits and systems for video technology*, Vol. 12, No.5,May 2002 .

ISBN: 978-988-18210-4-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)