Middle-East Journal of Scientific Research 22 (2): 261-266, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.22.02.21854

# Fibonacci Spiral Search for Block Matching in Video Coding

*[1]L.C. Manikandan and [2]R.K. Selvakumar*

[1]Department of Computer Science Engineering,
Mar Ephraem College of Engineering and Technology, Kanyakumari, Tamilnadu
[2]Department of Information Technology,
AGNI College of Technology, Chennai, Tamilnadu India

**Abstract:** In video coding schemes, block matching algorithm is the key element for motion estimation. Motion estimation can significantly improve video coding efficiency by reducing temporal redundancy existing in a video sequence. In this paper, we propose a new algorithm using Fibonacci Spiral Search (FSS) for fast block motion estimation. The proposed FSS algorithm finds small motion vectors with fewer search points. The experimental results are compared with the various fast motion estimation algorithms that have been employed in H.264 video standard. The reduction in the number of search points significantly reduces the computational complexity and paved the way for speedy compression.

**Key words:** Block-Matching Algorithm · Fibonacci Spiral Search · H.264 · Motion Estimation · Motion Vector · Video Coding.

## INTRODUCTION

Video coding is the process of compacting or shrinking a digital video sequence into a number of smaller bits. Video coding has become an essential part of modern multimedia systems, since it enables significant bit rate reduction of the video signal for transmission or storage. Motion Estimation (ME) is a significant part of any video coding system. A video sequence consists of a series of frames. To achieve video coding, the temporal redundancy between adjacent frames can be exploited. That is, a frame is selected as a reference frame and the subsequent frames are predicted from the reference frame using a technique known as Motion Estimation [1, 2]. The general goals of motion estimation methods are to improve the prediction accuracy, or to reduce the implementation complexity, or a combination of the two. Block Matching Algorithm (BMA) is the method for motion estimation, has been extensively espoused by current video coding standards like H.261, H.263, MPEG-1, MPEG-2, MPEG-4 and H.264 due to its effectiveness and simplicity. Full search (FS) algorithm [3] is the simplest BMA, which provides an optimal solution by fully search all candidates within the search window. But its high computational complexity made it difficult to be implemented in real-time software-based applications. In order to reduce the computational complexity of FS, several fast BMA's have been developedlike 3 Step Search Algorithm [4], New 3-Step Search Algorithm [4], Cross-Search Algorithm [5], 4-Step Search Algorithm [6] Diamond Search Algorithm [7], Spiral Search [8, 9] and Fast-Adaptive Rood Pattern Search FARPS [10,11] etc. These different kinds of BMA's journey different search designs and search approaches for verdict the prime motion vector with significantly compact the amount of searching points as matched with FS algorithm. In this paper, we suggest a simple, competent and fast BMA, called Fibonacci Spiral Search (FSS). In the suggested algorithm, two important aspects are considered to speed up and minimize the number of searching block. The search pattern and search strategy as proposed for performing fast block-matching ME.

**Block Matching Algorithm:** Block Matching Algorithm (BMA) [12-15] is the standard method to estimate the motion vector of a block in the current frame. The BMA eradicates the temporal redundancy, which is originated predominantly in any video sequence. The size of the block affects the performance of motion estimation. Small block sizes afford good approximation to the natural

**Corresponding Author:** R.K. Selvakumar, Department of Information Technology,
AGNI College of Technology, Chennai, Tamilnadu, India.

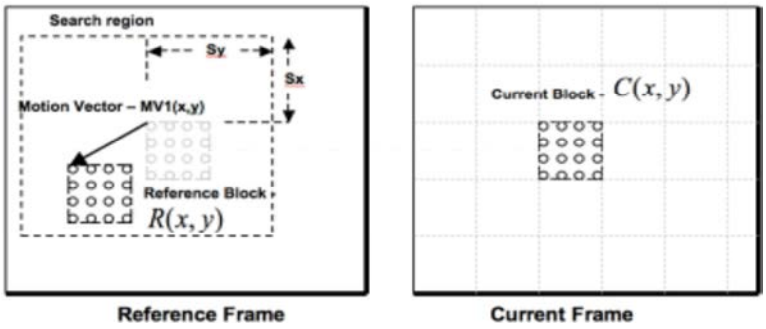


Fig. 1: Block Matching Concept

object boundaries; they also provide a good approximation to real motion. However, small block sizes produce a large amount of raw motion information, which increases the number of transmission bits or the required data compression complexity to condense this information. Small blocks also suffer from object (block) ambiguity problem and the random noise problem. Large block sizes may produce less accurate motion vectors, since a large block may likely contain pels moving at different speeds and directions. The idea overdue block matching is to divide frames into equal sized non-overlapping blocks and compute the displacement of the best-matched block from the prior frame with the motion vector of the block in the current frame within the search window. This basic operation is repeated until all the candidates have gone through and then the best match's candidate is identified. The location of the best matched candidate forms the estimated displacement vector. The computational load could also be reduced by calculating fewer blocks of an image. To increase search efficiency, we could place the initial search point at a location predicted from motion vectors of the spatially or the temporally adjacent blocks. A best matching can often be obtained by searching a smaller region surrounding this initial point. We could first separate the moving image blocks from the stationary ones and then conduct block matching only on the moving objects. This is because a moving or change detector can be implemented with much fewer calculations than a motion estimator. During block matching, each target block of the current frame is matched with a prior frame in order to discover the best matching block. BMA's calculate the best match using Mean Absolute Difference (MAD) [9] or any other distance measurement standards. Figure 1 shows the general block matching concepts.

Here the current frame is divided into a set of pixel blocks and motion estimation is achieved individually for each pixel block. Motion estimation is ended by detecting a pixel block from the reference frame that best matches the current block, whose motion is being projected. The reference pixel block is created by displacement from the current block's location in the reference frame. The displacement is provided by the Motion Vector (MV). MV consists of a pair (x, y) of horizontal and vertical displacement values. There are various criteria available for scheming block matching. The reference pixel blocks are created only from a region known as the search area. Search region describes the boundary for the motion vectors and limits the number of blocks to evaluate. The height and width of the search region are dependent on the motion in video sequences. The available computing power also defines the search range. The larger search region requires more computation due to increase in number of evaluated candidates. Typically the search region is kept wider (i.e. width is more than the height) since many video sequences often exhibit panning motion. The search region can also be changed adaptively depending upon the detected motion. The horizontal and vertical search range, Sx&Sy, define the search area (+/-Sx and +/- Sy) as illustrated in Figure 1.

**Matching Criteria:** The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. For a set of data, the equation for a least square line is determined by Sum of Squares for Error (SSE). The average error of each square is calculated in order to determine the Mean Squared Error (MSE) of a frame. Matching Criteria for video coding are Mean Absolute Error (MAE)[5], Sum of the Squared Error (SSE) and Mean Square Error (MSE). They are defined in equation (1), (2) and (3).

$$MAE(m,n) = \frac{1}{MN}\sum_{i=i_0}^{i_0+M}\sum_{j=j_0}^{j_0+N}[F_t(i,j) - F_{t-n}(i+m,j+n)] \quad (1)$$

$$SSE = \sum_{n=1}^{N}(I_n(x,y) - I'_n(x,y))^2 \quad (2)$$

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}[I(x,y) - I'(x,y)]^2 \quad (3)$$

**Quality Measure Criteria:** Peak Signal to Noise Ratio (PSNR) is measured on a logarithmic scale and depends on the mean square error (MSE) between an original and an impaired image or video frame. PSNR is a very popular quality measure, widely used to compare the quality of compressed and decompressed video images. The formula for calculating PSNR is:

$$PSNR = 10 \times \log_{10}(255^2/MS \quad (4)$$

**Fibonacci Spiral Search (FSS)**

**Fibonacci Sequence:** The Fibonacci numbers are defined recursively, meaning the value of the nth Fibonacci number depends on the value of previous Fibonacci numbers. The nth Fibonacci number is denoted Fn. The values of the Fibonacci numbers are: F1 = 1, F2 = 1 and Fn = Fn-1 + Fn-2; for n = 3; 4; 5; : : : : For example, F3 = F3-1 + F3-2 = F2 + F1 = 2 and F4 = F3 + F2 = 3. From basic definitions of Fibonacci Numbers F[n], the recurrence relation F[n+1] = F[n]+F[n-1] with initial conditions F[0]=0 and F[1]=1. It is noticed that each number is generated just the sum of the two numbers preceding it. The Fibonacci sequence is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc. That is, after the second term, each term is obtained by adding the previous two terms. The1 is found by adding the two numbers before it (0+1). Similarly, the 2 is found by adding the two numbers before it (1+1) and the 3is (1+2) and so on.

**The FSS Method:** Fibonacci Spiral can be constructed from an appropriate set of squares. Each block will fit nicely along the edge of the two previous blocks since F_n=F_(n-1)+F_(n-2). Given a table of records R1, R2,... RN whose keys are in increasing order K1 < K2 <... < KN, the algorithm searches for a given argument K. Assume N+1 = Fk+1

**Step1:** [Initialize] i ← Fk, p ← Fk-1, q ← Fk-2 (throughout the algorithm, p and q will be consecutive Fibonacci numbers)

**Step 2:** [Compare] If K < Ki, go to Step 3; if K > Ki go to Step 4; and if K = Ki, the algorithm terminates successfully.

**Step 3:** [Decrease i] If q=0, the algorithm terminates unsuccessfully. Otherwise, set (i, p, q) ← (i - q, q, p - q) (which moves p and q one position back in the Fibonacci sequence); then return to Step 2

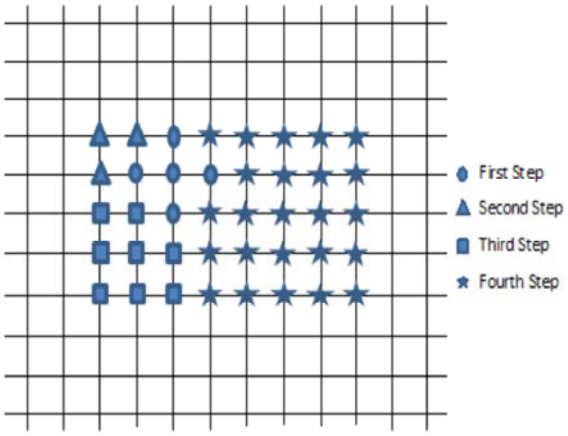


Fig. 2: FSS Anticlockwise directions

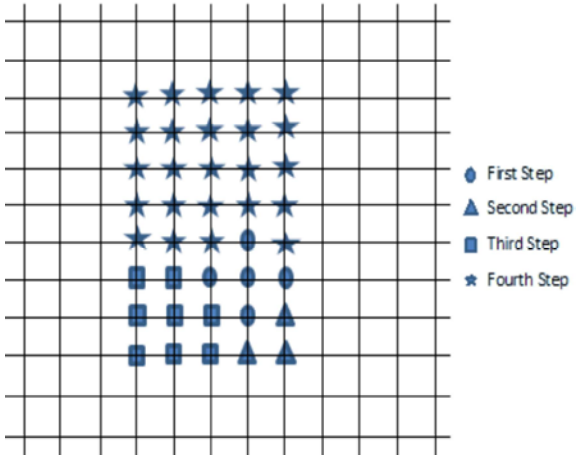


Fig. 3: FSS Clockwise directions

**Step 4:** [Increase i] If p=1, the algorithm terminates unsuccessfully. Otherwise set (i,p,q) ← (i + p, p ← p - q, q ← 2q - p) (which moves p and q two positions back in the Fibonacci sequence); and return to Step 2. It is necessary to keep it rotating around as you add blocks, so the shared edge is at the bottom, right, top and then left edge of the new square. Once the squares are drawn, start spiralling it out from the first square you drew. It should look something like the Figure 2 picture when completed. Figure 3 shows clockwise direction.

**FSS Algorithm:** In the proposed algorithm FSS measure, speed and the number of searching block are considered. The standard search window size is (2 × w) + 1. The maximum searching point of this algorithm is calculated byF_n×F_(n+1). Where Fn is the nth element in Fibonacci sequence. For example n = 4, the maximum searching point is F4 x F5 = 15. Figure 4 shows the Schematic Flow of FSS. In the whole block search process, the first step started

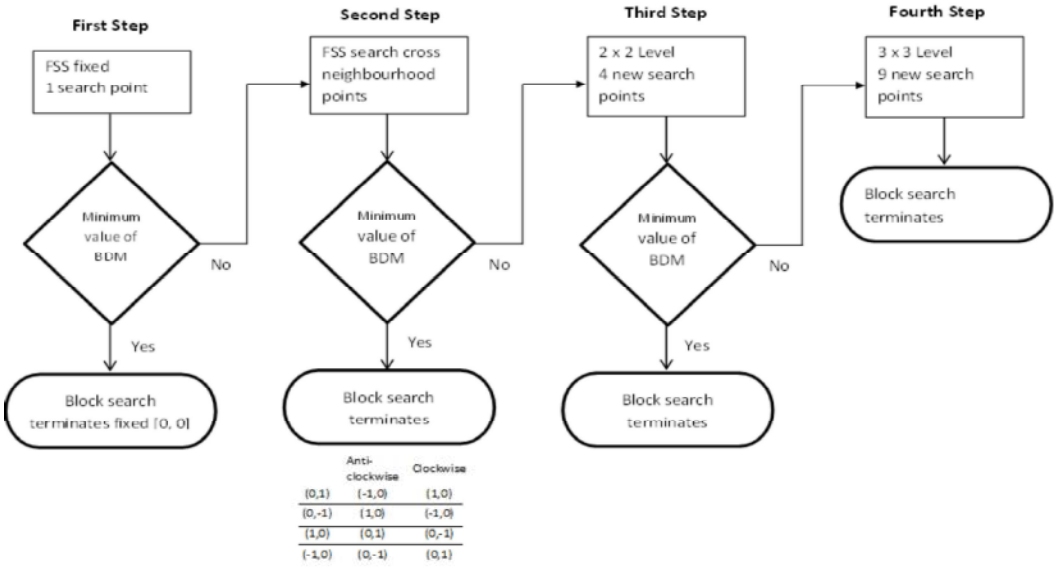


Fig. 4: Schematic Flows of FSS

with search point 1, if the block matched, then the process will stop else it would move to the next step in searching the direction by crosscheck BDM for the neighborhood four pixels. From which two minimum values will be evaluated and used in prediction of direction. Next level 2 x 2 search points will be used for MAD calculation and it will exceed up to the final level of Fibonacci series until which the block gets matched. In the worst criteria for the comparison of image m x n, total search points of 15 is more than enough to come out of the search loop. The search points are comparatively minimum to other block matching algorithms.

**Algorithm:**

**Step 1:** Current frame (cf) searching block is matched with a reference frame (rf) in the same block position. If it is matched, stop the searching process. Otherwise, it will match with 4 directional blocks.

**Step 2:** In 4 directional blocks, if any one of the blocks is matched; stop the searching process else we will find the minimum level direction.

**Step 3:** We will find the minimum of 2 values minimum2 and minimum1 from 4 directions based on this value we fix the direction either clockwise or anticlockwise.

**Step 4:** Next, we will find the matching block with the next Fibonacci level 2 x 2.

**Step 5:** In 2 x 2 levels, there are 4 search points, if any one of the search points is matched; stop the searching process. Otherwise, we will move to next Fibonacci level 3 x 3.

**Step 6:** In 3 x 3 levels, we fix the optimal level and select the best matched optimal block and stop the searching process.

**Performance Efficiency:** The proposed FSS algorithm is simulated using the luminance component of the first 90 frames of the "Football" and "Tennis" sequences. The experimental results for proposed algorithm and other BMA performance comparisons for CCIR601 sequence "football" and CIF sequence "Tennis "are summarized in Table 1. The term Peak Signal-to-Noise Ratio (PSNR) will affect the signal as it is the ratio between peak power and the power of distorting noise.

The standard deviation of the difference between predicted and observed value is represented by the Mean Square Error (MSE). The residuals are calculated from the individual difference of data sample and prediction error is found by individual difference computed from the sample. From the table, it's clearly defined that FSS is predominant of different block matching algorithms. The MSE value and PSNR are comparatively better than the others. It would directly impact in increasing the speed of block matching process and indirectly influence the compression to take place in a rapid way. Figure 5 and Figure 6 shows the performance comparison of FS, 3SS,

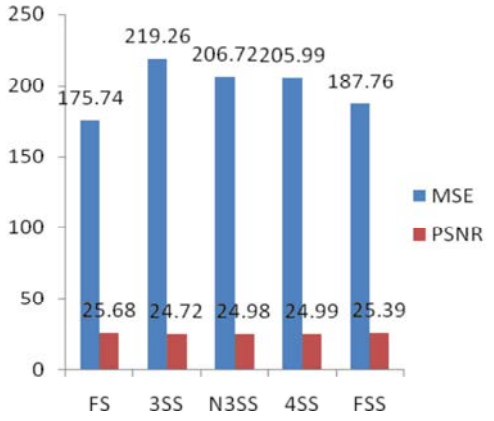


Fig. 5: MSE and PSNR analysis of football video

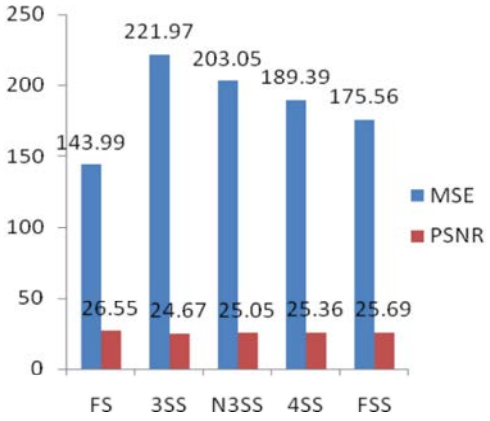


Fig. 6: MSE and PSNR analysis of Tennis video

Table 1: Simulation Results for FSS Algorithm and Other Fast BMAS

| Video Seq. | Algo. | MSE/pel | PSNR (db) | NSP/MB | Speed-up Ratio |
|---|---|---|---|---|---|
| Football | FS | 175.74 | 25.68 | 225 | 1 |
| | 3SS | 219.26 | 24.72 | 25 | 9 |
| | N3SS | 206.72 | 24.98 | 19.76 | 11.39 |
| | 4SS | 205.99 | 24.99 | 18.27 | 12.32 |
| | FSS | 187.76 | 25.39 | 15 | 15 |
| Tennis | FS | 143.99 | 26.55 | 225 | 1 |
| | 3SS | 221.97 | 24.67 | 25 | 9 |
| | N3SS | 203.05 | 25.05 | 22.66 | 10 |
| | 4SS | 189.39 | 25.36 | 19.87 | 11.32 |
| | FSS | 175.56 | 25.69 | 15 | 15 |

N3SS, 4SS and the proposed FSS for "Football" and "Tennis" sequence in terms of average MSE per pixel and PSNR.

## CONCLUSION

In this paper, search designs and search approaches of certain existing fast BMA's are examined. Based on these examines and annotations, Fibonacci Spiral Search (FSS) algorithm for motion estimation is developed. Our evaluation was based on two measures, speed up and minimize the number of searching block described the whole flow architecture is best suited in an optimized way. Generally FS algorithm provides better quality image (Minimum MSE and maximum PSNR) with less deviation from the reference frame. The proposed FSS algorithm provides better quality compared to 3SS and N3SS and closer than 4SS. and to elevate the searching process in more comprehending speed. The FSS algorithm is implemented in H.264 video-encoding environment

## REFERENCES

1. Jain, J.R. and A.K. Jain, 1981. Displacement measurement and its application in inter frame image coding, IEEE Trans. on Communications, pp: 1799-1808.
2. Koga, T., 1981. Motion Compensated Inter Frame Coding for Video Conferencing, 3.1-5.3.5.
3. Immanuel Alex Pandian, S., 2011. A Study on Block Matching Algorithms for Motion Estimation, In: Proc. International Journal on Computer Science and Engineering, pp: 34-44.
4. Renxiang Li, 1994. Circuits And Systems For Video Technology, A New Three-Step Search Algorithm for Block Motion Estimation, IEEE Trans, pp: 438-442.
5. Ghanbari, M., 1990. The Cross-Search Algorithm for Motion Estimation, IEEE Transactions on Communications, pp: 950-953.
6. Lai-Man Po Wing-Chung Ma, 1996. A Novel Four-Step Search Algorithm for Fast Block Motion Estimation, IEEE Transactions on Circuits Syst. Video Techno, pp: 313-317.
7. Shan Zhu and Ma Kai-Kuang, 2000. A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation, IEEE Transactions on Image Processing, pp: 287-290.
8. Deepak Turaga and Mohamed Alkanhal, 1998. Search Algorithms for Block-Matching in Motion Estimation, Springer.
9. Zahariadis, T.H. and D. Kalivas, 1995. A Spiral search algorithm for fast estimation of block motion vectors.
10. Gaikwad Ms. Maya, 2012. Implementation of four step search algorithm of motion estimation using FPGA, In: Proc, International Journal of Advanced Research in Computer Science and Electronics Engineering.
11. Kim, B.G., 2005. Fast-adaptive rood pattern search for block motion estimation, Electronics Letters.

12. Chandra Sekhar, C.H. and J.V.K. Ratnam, Comparison of Fast Block Matching Algorithms for Motion Estimation, In: Proc. International Journal of Electronics and Computer Science Engineering, pp: 1609-1618.
13. Barjatya, 2004. Block Matching Algorithms for Motion Estimation, Springer, DIP 6620.
14. Metkar S. and S. Talbar, 2013. Motion Estimation Techniques for Digital Video Coding, In: Proc. Springer Briefs in Computational Intelligence.
15. Puri, H.M. and D.L. Hang, 1987. An efficient block matching algorithm for motion compensated coding, Schilling, IEEE Int. Conf. Acoust. Speech and Signal Proc., pp: 1063-1066.