Low Complexity Recursive Search Based Motion Estimation Algorithm for Video Coding Applications

K. Virk, N. Khan, S. Masud, F. Nasim, S. Idris

Department of Computer Science,
Lahore University of Management Sciences,
Sector-U, D.H.A., Lahore 54792, Pakistan

ABSTRACT

A new recursive-search based motion estimation algorithm has been proposed that uses only a maximum of seven candidate motion vectors. The coding quality in terms of signal to noise ratio and compression is quite close to full search motion estimator. The algorithm is extremely light with respect to the computational load. Proposed algorithm improves upon the original three-dimensional recursive search (3DRS) algorithm that is well known as a high performance motion estimator. Results obtained from various test sequences indicate around 50% improvement in computational load over the conventional 3DRS estimator. Simultaneously, the technique provides about 8% improvement in the coded bit-rates. The convergence behavior of our improved algorithm towards variations in the motion vector field is also superior.

1. INTRODUCTION

Motion Estimation is the heart of video coding process and always requires great attention for attaining low bit rates with low computational load. For real-time systems, several faster approaches other than Full Search Block Matching have been proposed and utilized [1 - 6]. Among these, the techniques that best offer a good compromise between the coding quality and complexity are the 3-D Recursive Search (3DRS) [1] and Fast Diamond Search algorithms [3]. Our proposed algorithm improves the 3DRS technique to further reduce the computational load and improve compression. The original 3DRS algorithm is described in section 2 and the improvements are delineated in section 3. Performance results are tabulated in section 4. This is followed by conclusions in section 5.

2. EXISTING ALGORITHMS FOR RECURSIVE MOTION EXTIMATION

The seminal 3DRS motion estimator [1, 4] uses a small number of candidate vectors to find the motion vector of a macro block. Also, with the inherent smoothness constraint, it yields very coherent vector fields that closely correspond to the true motion of objects.

In block matching motion estimation algorithms, a displacement vector $\underline{D}(\underline{X}, n)$ is assigned to the center $\underline{X}=(X_x, X_y)^T$ of a block of position $B(\underline{X})$ in the current field n by searching a similar block in the previous field $n-1$ within a search area $SA(\underline{X})$ also centered at $\underline{X}$. The latter block has a center that is shifted with respect to $\underline{X}$ over the displacement vector (the motion vector) $\underline{D}(\underline{X}, n)$. To find $\underline{D}(\underline{X}, n)$, a number of candidate vectors $\underline{C}$ are evaluated using an error measure $\varepsilon(\underline{C}, \underline{X}, n)$ to quantify block similarity. Let block $B(\underline{X})$ be centered at $\underline{X}$ and is of size M*N. The displacement vector $\underline{D}(\underline{X}, n)$ resulting from the block matching process is a candidate vector $\underline{C}$ that yields the minimum value of an error function $\varepsilon(\underline{C}, \underline{X}, n)$, simply referred as $\varepsilon(\underline{C})$. Mathematically,

$$\underline{D}(\underline{X}, n) = \{ \underline{C} \in CS(\underline{X}, n) \mid \varepsilon(\underline{C}, \underline{X}, n) \leq \varepsilon(\underline{F}, \underline{X}, n) \forall \underline{F} \in CS(\underline{X}, n) \} \quad (1)$$

The most popular choice for the error function is Sum of Absolute Difference (SAD) criterion:

$$\varepsilon(\underline{C}, \underline{X}, n) = SAD = \sum_{\underline{x} \in B(\underline{X})} |f(\underline{x}, n) - f(\underline{x} - \underline{C}, n-1)| \quad (2)$$

The 3DRS algorithm tries to reduce the search strain by choosing a limited amount of candidate motion vectors (MV) based on the motion vectors found previously for neighboring blocks in the current and previous frames. These blocks are shown in figure 1.

The candidate set $CS(\underline{X}, n)$ consists of five vectors: three predictor vectors from spatio – temporal neighborhood and remaining two vectors are obtained by adding a random update to motion vector estimated for

previous block. Such a selection implicitly assumes spatial and/or temporal consistency and is given below in equation 3:

$$CS(\underline{X},n) = \left\{ \begin{array}{l} C_1 = D\left[\underline{X} - \begin{pmatrix} M \\ N \end{pmatrix}, n\right], \\ C_2 = D\left[\underline{X} - \begin{pmatrix} -M \\ N \end{pmatrix}, n\right], \\ C_3 = D\left[\underline{X} - \begin{pmatrix} 0 \\ -2N \end{pmatrix}, n-1\right], \\ C_4 = D\left[\underline{X} - \begin{pmatrix} M \\ 0 \end{pmatrix}\right] + U_a(\underline{X}), \\ C_5 = D\left[\underline{X} + \begin{pmatrix} M \\ 0 \end{pmatrix}\right] + U_b(\underline{X}) \end{array} \right\} \quad (3)$$

Here the vectors $U_a(\underline{X})$ and $U_b(\underline{X})$ are short and large random update vectors with elements having integer values in the interval [-3, +3]. Some offshoots of the standard 3DRS algorithms have been proposed such as [4] in which further compression improvement of up to 20% has been reported but at the expense of increasing the number of candidate motion vectors to thirteen from five candidates as employed in the original 3DRS. This results in a corresponding increase in the computation load.

3. PROPOSED ALGORITHM

Our proposed algorithm enhances and improves the 3DRS algorithm [1, 4] in several ways. Firstly, the search for the best motion vector is split in two stages: an original 3DRS algorithm like stage followed by a local refinement stage. This improves the convergence property of our algorithm compared with the 3DRS algorithm. Section 3.1 covers this aspect in more depth. Secondly, to enhance the speed of our algorithm, we propose certain shortcuts in the motion vector evaluation processes. These are explained in detail below in section 3.2. These checks enhance the speed of our algorithm without any compromise on the compression efficiency of the algorithm. Thus, in total we evaluate seven candidate vectors but with a higher speed than the five candidates evaluated in the original 3DRS algorithm.

3.1 Motion Vector Search Strategy

Our improved 3DRS algorithm is based on a set of *seven* candidate vectors evaluated in two stages. In the first stage three carefully chosen candidates from the neighborhood of the current block are evaluated. Let $\underline{X} = (X_x, X_y)^T$ be the coordinates of the current block. Our proposed algorithm first estimates the motion vector $D_1(\underline{X}, n)$ such that:

$$\underline{D}_1(\underline{X},n) = \{ \underline{C} \in CS_A(\underline{X},n) \mid \varepsilon(\underline{C}) \leq \varepsilon(\underline{F}) \forall \underline{F} \in CS_A(\underline{X},n) \} \quad (4)$$

where $\varepsilon(\underline{C}) = \varepsilon(\underline{C}, \underline{X}, n)$ and

$$CS_A(\underline{X},n) = \left\{ \begin{array}{l} C_{S1} = D\left[\underline{X} - \begin{pmatrix} M \\ N \end{pmatrix}, n\right], \\ C_{S2} = D\left[\underline{X} - \begin{pmatrix} -M \\ N \end{pmatrix}, n\right], \\ C_{T1} = D\left[\underline{X} - \begin{pmatrix} 0 \\ -2N \end{pmatrix}, n-1\right] \end{array} \right\} \quad (5)$$

In this spatio-temporal candidate vectors set used for the first stage, C_{S1} and C_{S2} are available for the current frame and C_{T1} is available from the previous frame. We have not included the random updated candidate vectors as against the 3DRS algorithm and its offshoot approaches previously proposed [1, 4]. We instead employ a second refinement stage as explained below.

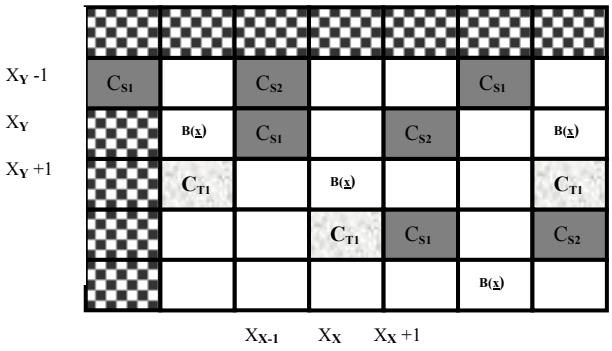

Figure 1: Neighbour of $B(\underline{X})$ that are used to derive three candidate motion vectors for the improved 3DRS algorithm and hence for second stage refinement

In the second stage motion vector refinement process, short diamond search is applied over $\underline{D}_1(\underline{X}, n)$ found at the first stage at one pel offset. This yields the fine refinement of the motion vector $\underline{D}_1(\underline{X}, n)$ selected in the first stage. The set of four spatial candidates of the second stage Short Diamond Search around the best matching motion vector $\underline{D}(\underline{X}, n)$ is given as:

$$\underline{D}(\underline{X},n) = \{ \underline{C} \in CS_B(\underline{X},n) \mid \varepsilon(\underline{C}) \leq \varepsilon(\underline{F}) \forall \underline{F} \in CS_B(\underline{X},n) \} \quad (6)$$

$$CS_B(\underline{X},n) = \left\{ \underline{D}_1(\underline{X},n) + \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \underline{D}_1(\underline{X},n) + \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \underline{D}_1(\underline{X},n) + \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \underline{D}_1(\underline{X},n) + \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \right\} \quad (7)$$

where $\varepsilon(\underline{C}) = \varepsilon(\underline{C}, \underline{X}, n)$

This second stage refinement helps in two ways. Since it selects the best candidate from the candidate set of predicted motion vectors and then refines it to get a better match for the current macroblock, the chance of finding a better match is increased. Moreover, the second stage refinement also helps to get a faster convergence of the recursive algorithm because a more accurate motion vector for the current macroblock also serves as a better predictor for the succeeding macroblock.

Note that the stated algorithm is somewhat modified at the edges of the frame. Blocks at the borders of the frame have fewer surrounding blocks. The non-existing blocks are clipped to the nearest existing blocks. For example, the MV of block (0, -1) is used instead of that of (-1, -1). Also MVs pointing outside of the frame are clipped to the nearest border position.

The candidates set for 3DRS algorithms may have more than one common motion vectors. Our algorithm obviates the need to evaluate them again. This results in a load reduction without any degradation in quality and video bit-rates.

3.2 Further Improvements in Motion Vector Computation

Evaluation of duplicate motion vectors in the candidate set is avoided. This saves considerable motion estimation time since such duplications exist frequently in the candidate set. Full computation of SAD for each and every candidate motion vector is also avoided by saving the minimum SAD ($SAD_{\min}$) found until this stage. While SAD is under computation, by accumulating the pixel value differences row by row for the current macroblock, its intermediate value at the end of each row is compared with $SAD_{\min}$. If the intermediate value exceeds $SAD_{\min}$; further computation of SAD for this candidate is meaningless and is therefore aborted.

During motion estimation, sometimes it is possible to find a very good match at an early stage. We have also exploited this fact to speed up motion estimation. If a certain candidate yields SAD value ($\varepsilon(\underline{C}, \underline{X}, n)$) lower than a certain threshold SAD_{LW_TH}, further motion vector search is aborted and the candidate $\underline{C}$ being evaluated is directly selected as the final motion vector for the current macroblock. The value SAD_{LW_TH} was determined through intensive testing.

3.3 Better True Motion Vectors Generation with Half Pel Refinements

We have also implemented Half-pel refinements in both algorithms. The inherent constraint of traditional 3DRS is its slow converging motion vectors, which results in deviation from true motion vectors [1]. We have addressed this problem through the use of small diamond search and apply further improvements by the use of half pixel precision based motion estimation algorithm around the best matched motion vectors of 3DRS. This helps in convergence to true motion vectors. Figure 3 depicts these results.

4. EXPERIMENTAL RESULTS

To evaluate the performance of our algorithm, we have applied our algorithm and reference 3DRS algorithm on different video sequences in QCIF resolution. The results are elaborated below.

4.1 Motion Estimation Performance Results

Figure 2 shows better convergence performance in recursion process in the case of *Coastguard* video sequence compared with the original 3DRS algorithm. The graph shows the number of 'Inter' coded blocks. Larger Inter coded blocks per frame implies the success of the motion estimator in finding more numbers of acceptable matching blocks in the previous frame leading to coding of more blocks in the current frame as 'Inter'. The *Coastguard* graph also depicts that from frame 65 to 80, the reference 3DRS shows discontinuous behavior as compared to improved implementation. Our algorithm thus converges to true motion vectors and hence results in lower bitrates.

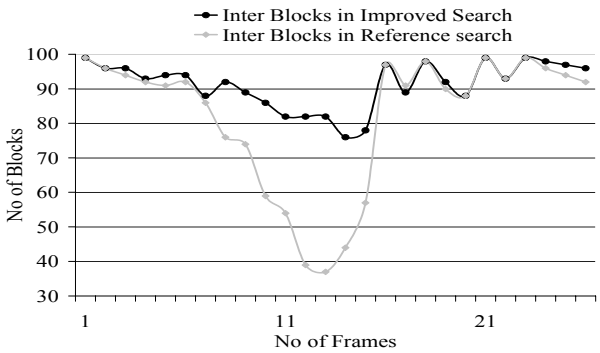

Figure 2: Convergence Performance in 'Coastguard'

4.2 Computational Load, Compression and True Motion Vectors Generation Results

Superior performance of our proposed algorithm is clear from Table 1 that quantifies the improvement in bit-rates and loads of our algorithm with respect to the original 3DRS algorithm. The SNR remains virtually the same. There is a reasonable reduction of 2% to 8% in bit stream sizes especially for moderate motion scenes. The load

reduction compared to 3DRS is 41-51%. This is due to the optimizations earlier suggested in section 3.2. Load performance results for various QCIF sequences are shown in figure 4.

Table 1: Performance comparison of the improved 3DRS and original 3DRS algorithms with half pel refinements

Scene	Computation load improvement over 3DRS (%)	Bit stream size improvement over 3DRS (%)
Coastguar	44.1	7.6
Foreman	41.4	5.8
CarPhone	44.3	2.2
Akiyo	51.4	1.7

We have also analyzed motion vectors for our improved search strategy and the reference. Here, it was observed, as depicted in figure 3, that our algorithm gives better true motion vectors. Even for minute motion that is not detected by the reference implementation, our algorithm results in further macroblock matches.

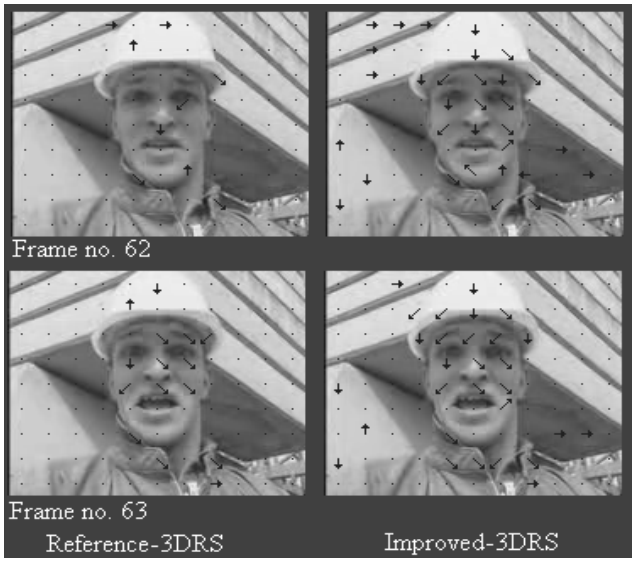

Figure 3: Motion vectors for improved and reference frames

5. CONCLUSIONS

Three dimensional recursive search algorithm and its derivatives have been shown to perform good motion estimation with reduced computational complexity. We have introduced further improvements in original 3DRS algorithm that result in an even faster motion estimator for standard video codecs at low bit-rate. The compression performance of proposed algorithm has been found to show an improvement of 2% to 8%. The computational complexity has shown a reduction in load of around 50%. The proposed algorithm is thus extremely useful for real-time video processing and implementation on a general-purpose computer.

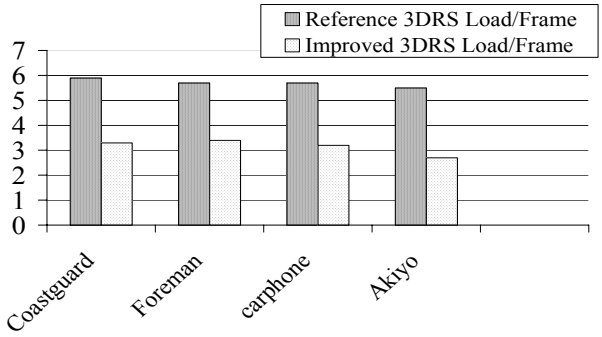

Figure 4: Actual average loads in MHz for different sequences

REFERENCES

[1] G. de Haan, P.W.A.C. Biezen, H. Huijgen, O. A. Ojo, "True motion estimation with 3D recursive search block matching", IEEE Trans. Circuits and Systems for Video Technology, Vol. 3, October 1993, pp. 368-379.

[2] J. Lu and M. L. Liou, "A Simple and Efficient Search Algorithm for Block-matching Motion Estimation", IEEE Trans. Circuits and Systems for Video Technology, vol. 7, no. 2, April 1997.

[3] J. Y. Tham, S. Ranganath, M. Ranganath and A. A. Kassim, "A Novel Unrestricted Center Biased Diamond Search Algorithm for Block Motion Estimation", IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 4, pp. 369-377, Aug. 1998.

[4] G. de Haan, L. lbaniy, S. Olivieriy, "Noise Robust Recursive Motion Estimation for H.263 based Video Conferencing Systems", Proceedings International Workshop on Multimedia Signal Processing, Sep. 1999, Copenhagen, pp. 345-350.

[5] Shan Zhu, Kai-Kuang Ma, "A New Diamond Search Algorithm for Fast Block Matching", IEEE Transactions on Circuits and Systems for Video Technology, vol.9, No.2, Feb. 2000, pp 287-290.

[6] R. Srinivasan, K. R. Rao, "Predictive Coding Based on Efficient Motion Estimation", IEEE Transactions on Communications, vol. com-33, No.8, Aug. 1985.