A brief survey of block matching motion estimation techniques

Sanjay Nair
Santa Clara University (ssnair@scu.edu)

Abstract: *This paper presents brief overview of several block-matching motion estimation techniques used in video compression systems. This paper also compares various algorithms used in H.264/AVC standard in terms of computational complexity and video quality. Test results with H.264/AVC software show that Fast block matching Motion estimation techniques reduce Motion estimation time/complexity with a minimum impact on quality.*

1. Introduction

Motion estimation and compensation plays a critical role in many compression systems as it exploits temporal redundancy that exists between successive video frames, which results in high degree of compression. The basic idea of block matching motion estimation algorithm is to divide the current video frame into several blocks and by using a block matching criterion, find the best match for these blocks in one or more reference frames. The matching block in reference frames are used as the predictor for the block in current frame. The displacement between current block and the reference block is known as Motion Vector (MV) and the difference between current block and reference block is known as prediction error (PE). Encoder sends only the Motion vectors and the prediction error residue data to the decoder which in turn uses motion compensation techniques to reconstruct the block from reference frames. Motion estimation and Compensation technique requires substantially fewer data than the intra coding methods used for coding the current frame.

As we studied in the class, if d_{max} represents maximum displacement for the search region, then to search all possible displacements in the reference frame it requires $(2d_{max}+1)^2$ block matching operations. This exhaustive search method is known as Full search which is computationally very expensive leading up to 90% or more of the encoding time. Over the years several fast motion estimation algorithms have been proposed to address the complexity of the full search motion estimation.

In the following section we'll examine several fast block based motion estimation techniques including algorithms used in H.264/AVC JM software which were not covered as part of the course. Section 3 compares the motion search techniques used in H.264 standard along the lines of computational complexity and objective quality based on the simulation results obtained using H.264 JM reference software implementation. Finally conclusions are drawn in section 4.

2. Fast Block Matching Algorithms

We have studied number of fast motion estimation algorithms such as 2D log search [1], three-step search (TSS) [2] modified conjugate direction search [3] etc. as part of the course. Some of these conventional algorithms are based on the assumption that the block distortion measure (BDM) increases monotonically as the search moves away from the global minimum distortion point and the distortion surface inside the search windows is unimodal. However, this assumption is not usually true as the error surface contains multiple local minimum points within the search range. Therefore, if a full search is not performed, there is a chance that the search will be trapped into one of the local minimum points which may not be the true minimum distortion point. Also, chances of trapping into one of the local minimum points increases as the search range increases. Block distortion measure is typically calculated using the SAD method (Sum or Absolute differences) given by the equation:

$$\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left| f(x+i, y+i) - f'(x+i+dx, y+i+dy)\right|$$

Where f and f' are current and reference frames and dx and dy are the x and y displacements. Following subsections cover various fast motion estimation algorithms.

2.1. New Three Step Search (NTSS)

TSS algorithm [2] is not very efficient in finding small motions since it uses a uniformly allocated search pattern in its first phase. NTSS algorithm [4] is based on the assumption that the global minimum distribution is center-biased as opposed to a uniform distribution. NTSS algorithm improves over TSS by using a center based checking pattern in its first step and providing a halfway-stop mechanism to stop the search upon finding the minimum BDM. As shown in figure 1, 17 points are checked as part of the first step (9 filled circles and 8 squares) including the origin point at the center.

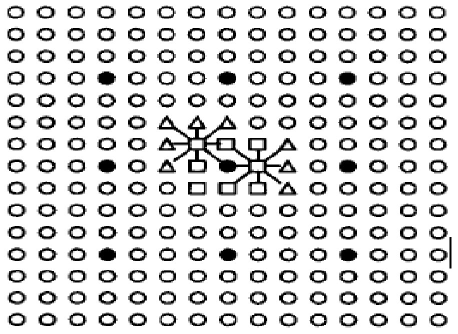
Figure 1: NTSS - Source [4]

If the minimum BDM is at the center, the motion vector is set to (0, 0). Otherwise, if the minimum BDM is at the square positions, NTSS sets the origin to a square position that has the lowest BDM. The algorithm then examines the BDM for the triangular positions (3 or 5 searches depending on the square position) and picks a position that has the lowest BDM. On the other hand, if the BDM was found at one of the filled circles, then NTSS follows regular TSS approach.

Although NTSS in worst case scenario checks more points than TSS, NTSS performs better than TSS in many cases because of its center based checking in the first phase and its early termination process.

2.2. Unrestricted Center-biased diamond Search (UCBDS)

UCBDS [5] is also based on the assumption that the global minimum BDM is non-uniformly biased towards the center point. UCBDS uses a diamond pattern as opposed to a square pattern used in NTSS or TSS. This is because the diamond pattern is more suitable for exploiting the center-biased characteristics of global minimum BDM. As shown in figure 2, first step in UCBDS algorithm is to compute and compare BDM for each of the 9 candidate search points marked as 1. If minimum BDM is found at the center point, then additional 4 points marked as a, b, c, and d are checked. On the other hand, if the BDM is found at one of the edges of the diamond, search center will be moved to the minimum BDM point. This process will continue until a minimum point is found within the search window.

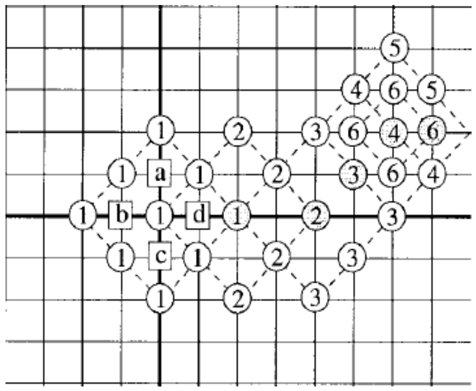

Figure 2: UCBDS - Source [5]

In the best case scenario, UCBDS algorithm needs only 13 search points to find the minimum BDM point.

2.3. Hexagon based Search (HEXBS)

HEXBS [6] method attempts to address the search point redundancies associated with the Diamond Shaped (DS) pattern due to its shape. For example, the horizontal and vertical distance of points in DS pattern is 2 and the diagonal distance is $\sqrt{2}$. This will result in inconsistent number of search points in different search directions.

HEXBS pattern consists of seven checking points including the center point. Two horizontal points are 2 steps away from the center point and the remaining 4 points are $\sqrt{5}$ steps away from the center point. As show in the figure below, there exists an approximate uniform distribution among 6 outer points.

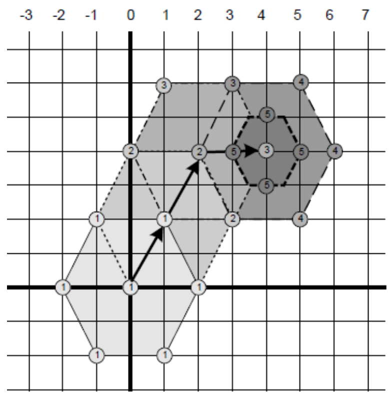

Figure 3: HEXBS-Source [6]

As shown in the figure 3, first step in HEXBS algorithm is to compute and compare BDM for each of the 7 candidate search points. If minimum BDM is found at the center point, then a small hexagon pattern is formed around the center point. 4 additional points are checked for minimum BDM. On the other hand, if the BDM is found at one of the edge points of the large hexagon, search center will be moved to the minimum BDM point. This process will continue until a minimum point is found within the search window.

Analysis shows that in some cases HEXBS based search can result in speed of greater than 80% compared to the DS algorithm.

2.4. Enhanced Predictive Zonal Search (EPZS)

EPZS algorithm is fundamentally different than the algorithms discussed earlier because of its usage of predictors to determine the initial search point. Instead of examining the motion vector at center point (0, 0), several predictors are examined to determine the initial search point. EPZS is based on the assumption that the motion vectors of the current block are highly correlated with motion vectors of adjacent blocks in spatial and temporal domain. EZPS algorithm examines the motion vectors at (0, 0), median motion vector and motion vectors of temporal and spatial adjacent blocks. EZPS also examines an accelerator motion vector derived from motion vectors of previous two frames.

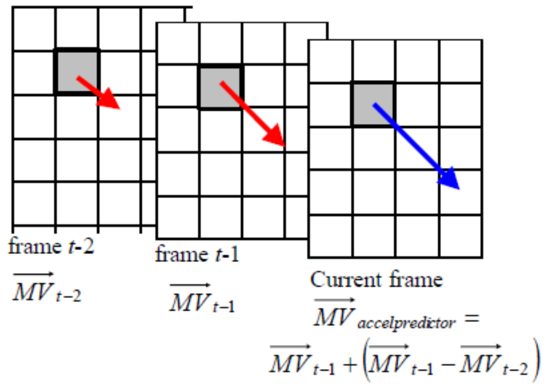
frame t-2
$\overrightarrow{MV}_{t-2}$
frame t-1
$\overrightarrow{MV}_{t-1}$
Current frame
$\overrightarrow{MV}_{accelpredictor} = \overrightarrow{MV}_{t-1} + \left(\overrightarrow{MV}_{t-1} - \overrightarrow{MV}_{t-2}\right)$

Figure 4: EZPS - Source [7]

The idea behind this motion vector is to determine if an object is accelerating or it is moving with a constant speed. EZPS algorithm uses a 3 stage thresholding process based on fixed and adaptive thresholds to determine the early search termination point. If early termination point is not found, it uses a refinement process based on a small diamond or square pattern around the minimum BDM point. Use of predictors and thresholding process allows EZPS to reduce the motion estimation complexity and achieve better output quality.

2.5. Unsymmetrical-cross Multi hexagon grid search (UMHexagonS)

Similar to EZPS, UMHexagonS [8] algorithm uses motion vector prediction techniques to determine the start point of the search. UMHexagonS algorithm also employs a faster integer pel algorithm combined with a center biased fractional pel search known as Center Biased fractional-pel search (CFBP). Predicting the starting point of the search and strategy for early termination of search is very important for the performance of a fast search algorithm.

UMHexagonS algorithm uses 4 prediction modes- Median Prediction (MP), UpLayer Prediction (UP), Corresponding-block prediction (CP), and Neighboring reference picture prediction (NRP) - to predict the initial motion vector. MP uses median MV values of nearest block on left, top, top left (or top right). UP uses hierarchical search of the up layer block based on H.264/AVC [9] intra prediction modes. CP uses MV of corresponding block from the previous frame. NRP uses MV from 2 previous frames. UMHexagonS algorithm uses 4 prediction modes, similar to that of MV prediction, for determining the SAD value that’ll be used in early termination decision. UMHexagonS algorithm starts with integer pel search to predict the motion vector. This algorithm involves four steps with different search patterns.

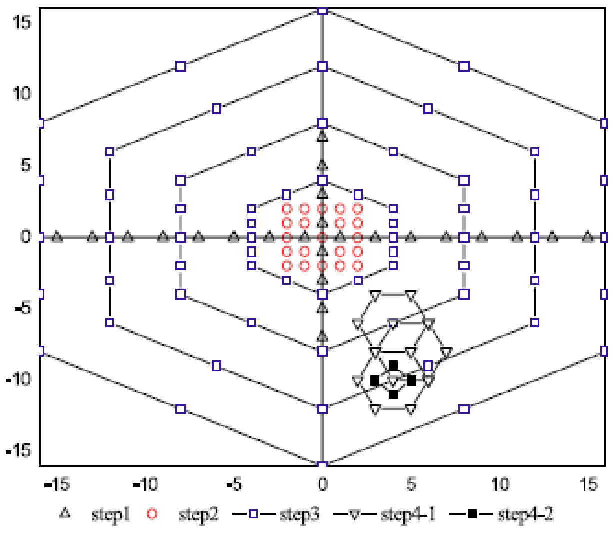

Figure 5:UMHexagonS: search range 16 – source [8]

As shown in the figure below, once the starting point is identified based on the prediction step, UMHexagonS algorithm starts with an unsymmetrical cross search. This is because changes in horizontal direction are heavier than changes in vertical direction. Once a minimum cost point is found, it will be used as the start search point for the next step, *small rectangular full search*. For the next step, a *multi-hexagon grid search* is employed. Basic search uses 16 point hexagon pattern. The search point derived from this step will be used as the start search point for the next step, *Extended Hexagon-based search (EHS)*. EHS is a modified version of HEXBS algorithm which is used as the center biased search algorithm.

Because of its prediction features, UMHexagonS resolves most of the local minimum issues with motion estimation. Also, this algorithm cuts down the search time by more than 90% when compared to the full search with PSNR loss less than 0.056 dB.

3. Block Matching ME Results

Results are obtained by running H.264/AVC reference software (JM) [10] on Windows platform. Default configuration file (encocder.cfg) for encode is used to obtain the results. Input file is of format IYUV and 59 frames are used for encoding. Parameter *SearchMode* in the configuration file is modified for setting different motion estimation search parameters. Several experiments are conducted. Details of the results from few of the experiments are shown below (*FS – Full Search, FFS – Fast Full Search*):

Sequence	**ME Search Algorithm**	**Results**					
		Encode Time	***ME Time***	***ME Time %***	***ME Speedup***	***PSNR***	***BitRate***
Akiyo_352x288	*FS*	307.544	278.204	91%		40.400	78.44
	FFS	366.056	337.114	92%	-58.910	40.400	78.44
	UMHex	61.240	31.273	51%	246.931	40.360	78.04
	SimpleHex	45.740	17.612	39%	260.592	40.340	78.3
	EPZS	57.662	29.933	52%	248.271	40.390	78.06
Mobile_352x288	*FS*	728.461	688.652	95%		34.610	1236.04
	FFS	372.271	331.611	89%	357.041	34.620	1236.55
	UMHex	89.372	51.231	57%	637.421	34.610	1236.57
	SimpleHex	80.877	42.558	53%	646.094	34.600	1240.61
	EPZS	83.282	44.544	54%	644.108	34.620	1221.53
Football_352x240	*FS*	1257.288	1215.514	97%		34.420	4557.18
	FFS	403.642	362.201	90%	853.313	34.410	4586.82
	UMHex	125.686	82.880	66%	1132.634	34.410	4592.07
	SimpleHex	64.210	34.450	54%	1181.064	34.390	4586.86
	EPZS	68.213	39.031	57%	1176.483	34.430	4584.45

Please note that the first sequence (Akiyo) does not have large motions. However, second (Mobile) and third (Football) sequences contain large motions between frames. Algorithms such as UMHex (UMHexagonS), SimpleHex (similar to HEXBS) and EPZS algorithms are covered in section 2. As shown in the table, in case of full search, ME takes than 90+% of the total encoding time. By using fast search methods this time can be cut down drastically.

4. Conclusion

In this paper, we briefly looked at various block matching motion estimation algorithms. Some of the algorithms presented in section 2 are used in H.264/AVC reference software. Experiments with H.264/AVC reference software (JM) showed that fast motion estimation techniques reduce the motion estimation time/complexity with minimal or no impact on the video quality.

REFERENCES

[1] J. R. Jain and A. K. Jain, “Displacement measurement and its application in interframe image coding,” IEEE transactions on communications. VOL. COM-29, NO. 12, DECEMBER 1981

[2] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion compensated interframe coding for video conferencing,” in Proc. NTC81, New Orleans, LA, Nov.1981, pp. C9.6.1-9.6.5

[3] S. Kappagantula and K. R. Rao, “Motion compensated interframe image prediction,” IEEE Transactions on communications. vol. COM-33, pp. 101 1-1015, July. 1985

[4] R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, no: 4, pp. 438-442, Aug. 1994.

[5] J Y Tham, S Ranganath, M Ranganath, A A Kassim “A Novel Unrestricted Center-Biased Search for Block Motion Estimation” IEEE Transactions on Circuits and Systems for Video Technology, Aug 1998

[6] C Zhu, L-P Chau, X Lin "Hexagon-based search pattern for fast block motion estimation" IEEE Transactions on Circuits and Systems for Video Technology, 2002

[7] Tourapis, Alexis M. "Enhanced predictive zonal search for single and multiple frame motion estimation." Electronic Imaging 2002. International Society for Optics and Photonics, 2002.

[8] Chen, Zhibo, Yun He, and Yidong Chen. "Fast Integer and fractional pel motion estimation." Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-F017 (2002).

[9] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, “Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC) – Joint Committee Draft”, document JVTE022d3. doc, Sep’02

[10] H.264 Reference Software, JM and KTA http://iphome.hhi.de/suehring/tml/download