

Variable-Size Block Selection in H.264

Lin Huang

Email: huanglinus@yahoo.com

Abstract

The method of block matching for motion compensation has been used in many video coding systems, including H.264. In H.264, the feature of variable-size block motion compensation has been proved to improve coding efficiency and reduce computational complexity. In this paper, several schemes of variable-size block motion compensation (VSBM) will be introduced. And this paper will focus on the scheme of Bottom-up VSBM and a proposed optimized Bottom-up VSBM.

Index terms: motion compensation, variable-size block, VSBM, covering quadtree, full search, three step search, Top-down VSBM, Bottom-up VSBM, Mixed VSBM

1. Introduction

The main objective behind the H.264 video standard is to develop a high-performance video coding system by adopting a “back to basics” approach where simple and straightforward design using well-known building blocks is used. But there are differences between H.264 and the other standards. Especially on the motion estimation/compensation side, H.264 employs blocks of different sizes and shapes, higher resolution 1/4-pel motion estimation, multiple reference frame selection and multiple bi-directional mode selection. And this paper will concentrate on the selection of VSBM in inter frame predictions to exploit the temporal redundancy between frames in an image sequence, which can get better results in the system coding efficiency and computational complexity than fixed-size block motion compensation (FSBM).

H.264 supports motion compensation block sizes ranging from 16x16 to 4x4 luminance pixels with many options between the two sizes (Fig.1). The luminance component of each macro-block (16x16 pixels) may be partitioned to 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4 blocks by *covering quadtree* decomposition. A covering quadtree is a

quadtree where each node has four children or none (Fig. 2). Choosing a large block size means a small number of bits for motion vectors and is often used in stationary image contents, uniform motion, or homogenous areas; on the other hand, the small block size will have a large number of bits for motion vectors, and be used in complex motion and image detail areas.

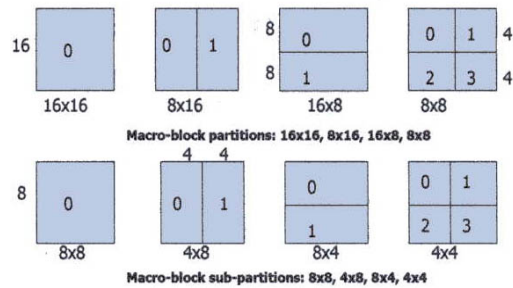


Fig. 1 Variable-size blocks in H.264

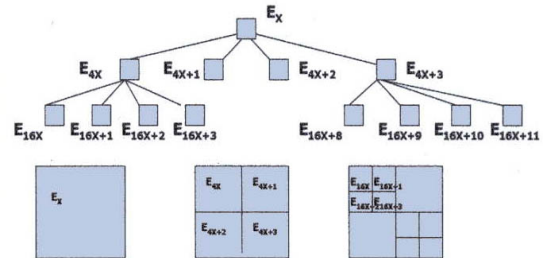


Fig. 2 Covering quadtree structure

Motion compensation is the relative displacement between two blocks. In H.264, for each block in the frame, a search is made in a predefined area of the reference frames which can be previous and/or future frames in an image sequence. The search is for the best matching block, the position which minimizes a distortion measure among the sets of pixels comprising the blocks. The distortion measure is called *cost function* which can be mean square error (MSE) of the corresponding pixels, or Mean Absolute Error (MAE), or Sum of Absolute Difference and something like that. And the search schemes can be Full search, Three-step search, Four-step search, Hierarchical search, Two-dimensional logarithm search, Block-based gradient descent search and so

on.

A lot of block based motion compensation have been proposed, including FSBM scheme originally proposed by Jain [1], top-down VSBM by Chan et al. [2], a region-wise motion compensation techniques by J.Zhang et al. [3], mixed VSBM and adaptive threshold selection schemes by Yu-Kuang et al. [4], Bottom-up VSBM by Injong Rhee et al. [5], and here will propose an optimized Bottom-up VSBM based on [5].

In this paper, Section 2 describes FSBM briefly, Section 3 and Section 4 present top-down VSBM and mixed VSBM respectively, and Section 5 introduces bottom-up VSBM and optimized Bottom-up VSBM in detail. The evaluation is stated in Section 6. And Section 7 will summarize the work.

2. FSBM

The conventional FSBM is used in block based motion compensation of international standards, MPEG-1, 2, 4, and H.261 / H.263. It usually divides a frame into 16x16 pixels macro-blocks. Assuming that each block of pixels undergoes the same motion, it compensates the motion of a frame block by block.

In a real image sequence, the boundaries of moving objects seldom coincide with the block boundaries. So FSBM produces blocking artifacts and poor prediction for complicated motion sequences. A *blocking artifact* is a straight-line discontinuity generated when adjacent blocks use different motion vectors.

Block prediction error means the difference of pixel values between the actual block in the current frame and the “matched” block in reference frames. There is a trade-off between the number of blocks and the prediction error. The smaller the block size is, the smaller the block prediction error is, but it will bring up larger block number and more motion vector bits. In FSBM, increasing the block size is the only way to reduce the number of motion vectors, but with bigger block size, there is less chance to get a good match. So, in H.264, inter frame motion compensation will use variable-size block to get coding efficiency.

3. Top-down VSBM

Chan et al.[2] proposed a top-down VSBM technique for high performance video coding systems. Large area with uniform motion will be

represented by few large blocks to reduce the number of motion vectors.

The steps of top-down VSBM are:

- Find matched large blocks initially
- If for the best match of any block, the resulting error is above a prescribed threshold, then split the block into 4 smaller blocks
- Repeat the above step till the maximum number of blocks, or locally minimum errors are obtained
- Re-merging small blocks to form large ones to remove those blocks which don't contribute to improving image quality

The problem of top-down VSBM suffers from “*majority effect*”. The reason is: a large block may be considered matched when having an acceptable block error, but a small area of the block may represent a feature with disparate motion, and the disparate motion is ignored because the error is biased by the majority of the pixels.

Meanwhile, there is a confliction between performance and computational complexity. If one large block needs to be split, one should perform motion estimation for those split smaller blocks to find best motion vectors of them to achieve higher performance. One can determine a smaller threshold to achieve better performance, but it sacrifices that more large blocks need to split to smaller blocks, the computational complexity will be higher.

4. Mixed VSBM

Yu-Kuang Tu et al. adopted mixed VSBM algorithms to select variable-size block for video systems, and considered the quantization parameter and Lagrange cost function to determine the threshold for comparing the distance between the motion vectors of the two blocks. The computational complexity can be reduced because the scheme only performs the motion full search for one block size 8x8.

The steps of mixed VSBM are:

- Use 8x8 block size as initial block size for motion estimation (Motion estimation in this step uses full search methods)
- Neighboring blocks are then merged in a quadtree manner if they have at least one vector in common
- Based on the 8x8 search results, split procedures can be applied to the areas with complex motions.

5. Bottom-up VSBM and Optimization

Rhee et al [5] proposed this bottom-up VSBM algorithm. It is based on the following observation. When an object moves, the true motion of the object can be recovered by using not only local information, but also the global information. For example, in Fig. 3, as the object moves to the top-right corner, both of the motion vectors shown are equally probable candidates of the true motion within that window.

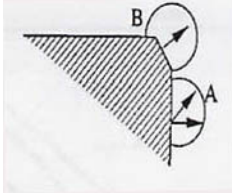


Fig. 3 motion vectors

The steps of Bottom-up VSBM are:

- Given an image sequence f_1, f_2, \dots , the algorithm first divides the image f_i frame into small fixed-size blocks (4x4 block size)
- Motion information for each of these small blocks is obtained by block matching
 - Denote each block as (x, y, s) , where (x, y) is the coordinate of the upper-leftmost pixel of the block, s is the length of a side of the square block
 - Given a block A (x, y, s) and a predefined search window of motion vector V , calculate the MSE between blocks A (x, y, s) in f_i and B $(x+x', y+y', s)$ in f_{i-1} for each (x', y') in V , using one-time full search technique
 - For each block b in f_i , obtain a set of motion vectors I_b , called initial set of b . Its matching error is less than a prescribed threshold.
- Merge four sibling blocks into a parent block if and only if the intersection set of each sibling block is not empty and the intersection set of parent block is not empty
- Repeat this process at each level of the tree, from the bottom level to the top till the motion vector for a block which cannot be merged into its parent or for a block which does not have any candidate motion vector.

Fig. 4 illustrates the merging process. Sibling blocks 11,12,15,16 have one common vector, the four blocks will be merged and the merged block's motion vector is the common vector. The sibling blocks 1, 2, 5, 6 have two common vectors, the

vector of the merged block will be the one with the smallest block error. In sibling blocks 9, 10, 13, 14, and another sibling block set 3, 4, 7, 8, there is no common vector among them, they will be kept, and the motion vector for each non-merged block is selected as the one associated with the minimum error.

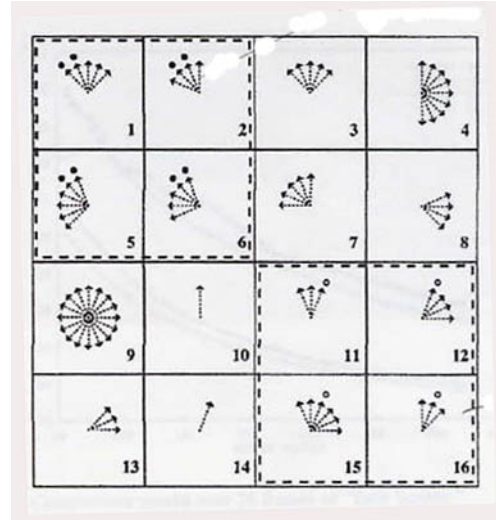


Fig.4 Initial sets of motion vectors and merging process

Due to the above Bottom-up VSBM uses full search method to get the initial vector sets within the search window area, it can get a more accurate motion compensation than other search process, but it needs to search more points within the search window, in other words, it takes more computation time than some other search methods. This paper proposes an optimized Bottom-up VSBM which will adopt 3 step search algorithm to find best match much faster than full search within the search area. The drawback is the error will be a little bit larger than the one by full search. But it can be acceptable.

The 3 step search algorithm is:

- Step 0: Set step size d to $w/2$, where w is the maximum motion displacement of pixel per frame.
- Step 1: Compute the value of the cost function in the following nine positions: center $(0,0)$ and $(0,d), (d,d), (d,0), (-d,d), (-d,0), (-d,-d), (0,-d), (d,-d)$.
- Step 2: Search the minimum from the values above and set the new center $(0,0)$ to the minimum.
- Step 3: If d is greater than 1 then set the step size d to $d/2$ and repeat Step 1, otherwise proceed.
- Step 4: The displacement of the current center from the original center is the

motion vector for the current macro-block.

The optimized Bottom-up VSBM can be done by the following steps:

- Given image sequence, and divides the frame i into 4×4 block size
- Using 3-step search to find initial motion vector sets for these block in frame i . The reference frames can be previous or future frames. And proposed threshold is the mean of the errors.
- Merging process in frame i :
 - if there exists a common vector among any 4 sibling small blocks, those block can be merged to a larger block with that common vector;
 - if there are two or more common vectors among the 4 sibling blocks, those blocks still can be merged with the vector which causes the smallest MSE;
 - if there is no common vectors, those blocks can be kept.
- Repeat merging process level by level till no more common vectors can be merged to the parents

Due to 3-step search is much more efficient than full search, proposed complexity should be better, although it sacrifices prediction error.

6. Evaluation

Fig 5 shows the block structures for Top-down VSBM and Bottom-up VSBM of "split Screen". The Bottom-up VSBM method evaluate the performance by using image sequences: "Miss America", "Split Screen", and compares with Top-down VSBM, FSBM. The distortion measure is MSE. Fig. 6 illustrates the comparative results over 28 frames of "Miss America", Fig. 7 is comparative results for 256 blocks/frame of "Split screen".

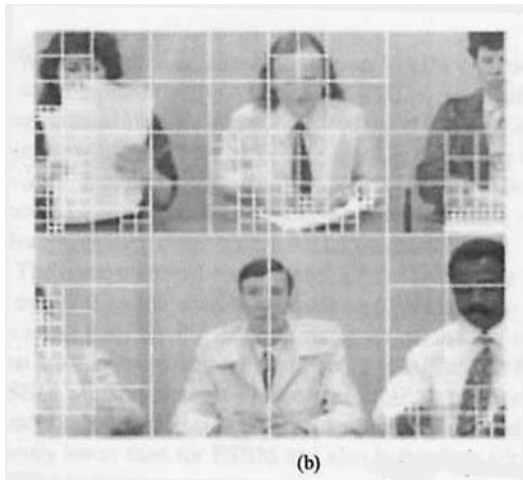


Fig. 5 Comparison of Bottom-up VSBM (b) and Top-down VSBM (c) for "Split Screen"

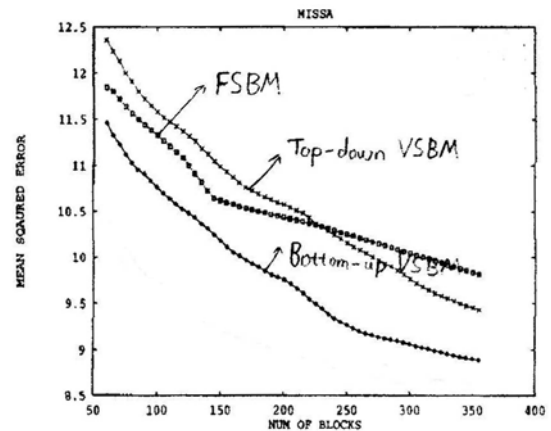


Fig. 6 Comparison of FSBM, Top-down VSBM, and Bottom-up VSBM for "Miss America" (28 frames)

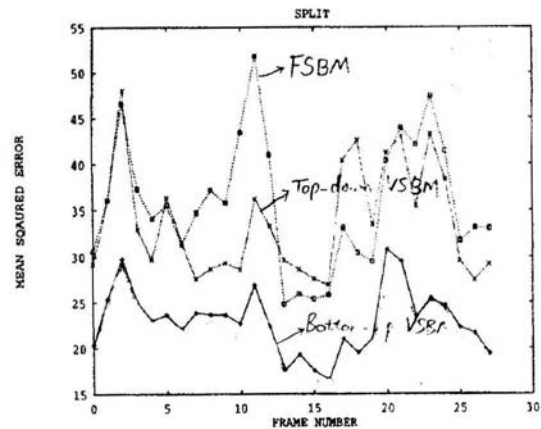


Fig. 7 Comparison of FSBM, Top-down VSBM, and Bottom-up VSBM for "Split screen" (256 blocks/frame)

7. Conclusion

In the above schemes of variable-size block selection, threshold is an important issue for the prediction error. The project uses the mean error as the threshold. But if the proposed scheme uses adaptive threshold method in [2], that will be better.

This project tries to evaluate and test the several previous algorithms, and proposes an optimized bottom-up VSBM by using 3 step search algorithm instead of the traditional full search algorithm. In fact, there are some other search methods which can improve the search performance with smaller MSE and computation complexity. These search methods can be new 3 step search, 4 step search, Hierarchical search, Two dimensional Logarithm search, Block-based gradient descent search. But, the future work still is to fix the bugs of the programs if possible.

The last issue is to select the cost function to evaluate the prediction error. There are three ways to be used in the above VSBM schemes: MSE, MAE, SAD. So the future work should include the problem about the selection of the cost function.

References

1. J.R.Jain and A.K.Jain, "Displacement measurement and its application in interframe image coding", IEEE Trans. Commun, vol. COM-29, pp.1799-1808, Dec. 1981
2. M.H.Chan, etc. " Variable size Block Matching Motion Compensation with Application to Video Coding"
3. J.Zhang, M.O.Ahmad, etc. " New Windowing Techniques for Variable-size Block Motion Compensation" IEE Proc.-Vis. Image Signal Process., Vol. 145, No. 6, Dec 1998
4. Yu-Kuang, etc. " Fast Variable-size Block Motion Estimation Using Merging Procedure with an Adaptive Threshold", IEEE ICME 2003, pp789-792
5. Injong Rhee, etc. "Quadtree-Structured Variable-size Block-Matching Motion Estimation with Minimal Error", IEEE Trans. Circuit and Systems for Video Tech, Vol.10, No.1, pp42-50, Feb 2000
6. <http://www.vcodex.com>
7. <http://www.ee.cityu.edu.hk/~lmpo/publications/#code>
8. <http://blackboard.fau.edu>
9. <http://bs.hhi.de/~suehring/tml/>
10. <http://www.envivio.com/products/h264.html>