On Projection-Based Block Matching Motion Estimation

Chengjie Tu and Trac D. Tran*

Abstract— **This paper introduces a fast block-based motion estimation algorithm based on matching projections. The idea is simple: two blocks cannot match well if their corresponding 1D projections do not match well. Taking advantage of this observation, the expensive 2D block matching problem can be translated to a simpler 1D matching one by quickly eliminating a majority of matching candidates. The proposed motion estimation algorithm offers computational scalability through a single adaptive parameter and global optimum can still be achieved. Moreover, an efficient implementation to compute projections and to buffer recyclable data is also presented. Experiments show that our algorithm is several times faster than the full search algorithm with nearly identical prediction performance.**

Keywords— **Motion Estimation, Fast Block-Matching Algorithm, Projection.**

I. INTRODUCTION

MOTION estimation/compensation (ME/MC) can effectively eliminate temporal redundancy in a video sequence and is the key to high-quality video coding. Despite its simplicity, block-based or block-matching ME (BME) is a critical component in most state-of-the-art video codecs, including MPEG-2 [1], H.263+ [2], MPEG-4 [3] and the upcoming H.26L [4].

BME partitions the current frame into non-overlapped blocks. For each block, BME searches all displaced blocks within a search window in the reference frame to find the best matched block. The displacement of the best matched block is called a motion vector. Motion vectors and prediction residuals are coded. The mean of absolute differences (MAD) is the most popular matching criterion because it is computationally inexpensive. The mean square error (MSE) is occasionally used to obtain better objective coding performance [5]. In either case, an expensive 2D block matching process is involved. The simplest search method is the full search algorithm (FS), where full-search 2D matching is applied to all candidate blocks. FS certainly achieves global optimum for the given translational motion model. However, the computational complexity for FS is tremendous. Many fast algorithms have been developed to reduce the amount of computation.

Early termination is a BME framework that can guarantee optimal matching performance by maintaining the lower bound of the matching errors. The calculation of the matching criterion can be terminated immediately when it is greater than the known lower bound obtained so far. This way, most candidates can be excluded by matching only part of the pixels. The tighter the lower bound and the earlier we can find it, the faster the algorithm becomes. So the search order is of great importance. The spiral full search is a typical example of this philosophy. It starts with zero displacement and moves spirally to candidates with larger displacements. Hence, the spiral search tries to optimize the search order. The combination of early termination and spiral search can significantly speed up BME without losing any prediction performance and is commonly employed in high-performance codecs. The successive elimination algorithm (SEA) [6], [7] is another example of optimal block matching where the authors present a method to obtain much tighter lower bounds.

Most of the popular fast BME algorithms in practical systems sacrifice prediction performance to reduce BME complexity. Algorithms such as the three-step search (TSS) [8], the logarithmic search [9], and the conjugate direction search [10] all subsample the motion displacement space and thus reduce the number of candidates. They can be extremely fast at the expense of prediction performance. Hierarchical search, which uses reduced resolution for the first one or more stages of the search, has been investigated not only in the spatial domain [11] but also in the transform domain [12]. Algorithms with this philosophy such as the telescopic search [13] and the predictive pattern search [14] assume temporal continuity of motion displacements and use prediction to initialize the search and concentrate most of their later searches locally around that point. Needless to say, all of these aforementioned algorithms cannot guarantee matching optimality.

The computational complexity of BME is the direct consequence of the expensive 2D block matching process. This paper presents a BME algorithm based on projections (PBME) which can reduce the number of 2D matchings tremendously. The relationship between motion and projections has been established before within the context of global motion estimation for medical imaging applications [15] as well as block-matching motion estimation for video coding [16], [17]. Our proposed application of this theory is different comparing to those in [16], [17]. Our algorithm relaxes the optimal constraint in [6], [7], [17]: different elimination criteria are used and a single adaptive scalar is introduced to control the complexity-performance trade-off of the algorithm. By controlling the percentage of candidates excluded by 1D matching, the speed of the PBME algorithm is readily controllable. We also propose an efficient computation of projections and describe the buffering of recyclable data – two important aspects of the algorithm.

Manuscript received ___. The authors are with the Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218. Email: {cjtu, trac}@jhu.edu. This research was supported by NSF under CAREER Award CCR-0093262 and by FastVDO Inc.

II. Projections

A. Definition

The projection of $f(x,y)$ at angle α is defined as

$$g(p,\alpha) = \int\int_D f(x,y)\delta(p - xcos(\alpha) - ysin(\alpha))dxdy. \quad (1)$$

Only the discrete version with $\alpha = 0$ (or $\alpha = \frac{\pi}{2}$) is involved in the proposed algorithm.

Suppose the frame size is $W \times H$, the block size is $B_w \times B_h$ and the search window size is $W_w \times W_h$. To predict one block, there are $W_w \times W_h$ candidates ($B_w \times B_h$ blocks) to search.

Let $B^{x,y}$ denote a 2D block of pixels with the top-left position at (x,y) in a video frame and $B^{x,y}_{i,j}$, where $0 \le i < B_w$ and $0 \le j < B_h$, be the pixel at the j^{th} row and the i^{th} column of the block. Define the (vertical) projection of $B^{x,y}$ be $LB^{x,y}$, a 1D row vector whose i^{th} value is the sum of the i^{th} column of $B^{x,y}$ (Fig. 1):

$$LB^{x,y}_i = \sum_{j=0}^{B_h-1} B^{x,y}_{i,j}, i = 0,1,...,B_w - 1. \quad (2)$$

By projection, a $B_w \times B_h$ 2D block is reduced to a B_w-component 1D vector and only the DC information of each column is preserved.

We can further define the sum to be:

$$PB^{x,y} = \sum_{i=0}^{B_w-1} LB^{x,y}_i = \sum_{i=0}^{B_w-1}\sum_{j=0}^{B_h-1} B^{x,y}_{i,j}. \quad (3)$$

Hence $PB^{x,y}$ preserves the DC information of $B^{x,y}$.

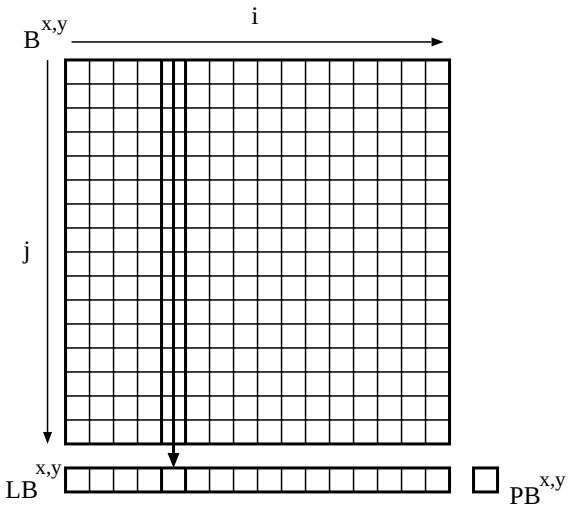

Fig. 1. Projection.

B. Fast Projection

In the current frame, there are only $\frac{WH}{B_w B_h}$ non-overlapped blocks and the computational complexity of the 1D projection above is small (total $\mathcal{O}(WH)$ operations). In other words, the complexity increases linearly with the video size only.

In the reference frame, there are $W \times H$ different blocks. $B^{x,y}$ and its direct right (lower) neighbor $B^{x+1,y}$ ($B^{x,y+1}$) share all pixels except two columns (or rows). If $LB^{x,y}$ is known, $LB^{x+1,y}$ and $LB^{x,y+1}$ can be updated efficiently as illustrated in Fig. 2:

$$LB^{x,y+1}_i = LB^{x,y}_i - B^{x,y}_{i,0} + B^{x,y+1}_{i,B_h-1} \quad (4)$$

$$LB^{x+1,y}_i = \begin{cases} LB^{x,y}_{i+1} & \text{if } i < B_w - 1, \\ \sum_{j=0}^{B_h-1} B^{x+1,y}_{B_w-1,j} & \text{if } i = B_w - 1. \end{cases} \quad (5)$$

Starting with $B^{0,0}$, with proper buffering, only 2 operations per pixel are required on average to compute the projection of a block in this updating manner. The cost for projection is therefore only $\mathcal{O}(2WH)$ operations.

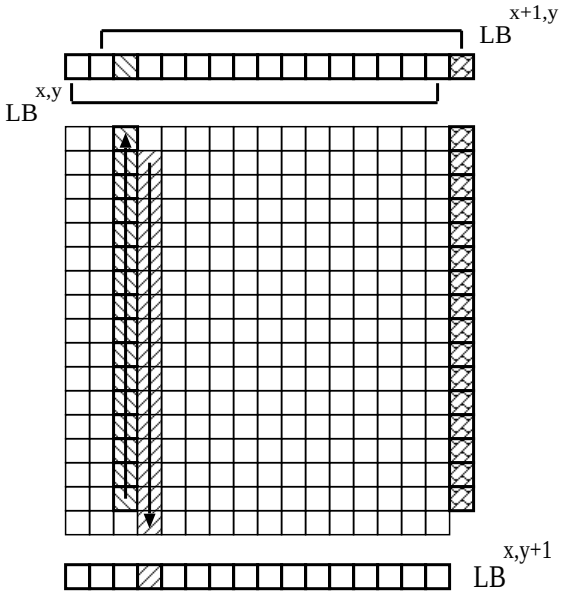

Fig. 2. Fast projection.

C. Buffering Scheme

To search for the motion vectors of a strip of blocks with their top corners at the same position y in the current frame, only the blocks with top corners within $[y - W_h/2, y + W_h/2)$ are involved in the reference frame, which is a $W \times W_h$ strip. So a $W \times W_h$ buffer instead of $W \times H$ (usually $H >> W_h$) buffer is sufficient to store all recyclable projections. When moving to the next strip, we need to slide the buffer up by B_h lines, discard the B_h lines moving out and update the buffer with B_h lines moving in using fast projection. An additional B_w-point buffer is necessary for the current block in the current frame.

III. Projection-based BME

A. 2D Matching

For a block $C^{x,y}$ in the current frame, BME searches all displaced blocks $R^{x+dx,y+dy}$ in the search window in the

reference frame for the best matched block. The matching error (MAD) is:

$$MAD(dx,dy) = \sum_{i=0}^{B_w-1}\sum_{j=0}^{B_h-1} |C_{i,j}^{x,y} - R_{i,j}^{x+dx,y+dy}|. \quad (6)$$

The cost of this expensive 2D block matching is $\mathcal{O}(B_w \times B_h)$ operations.

BME tries to find the minimum value of MAD, labeled MAD_{min}, and the corresponding displacement (dx, dy) is the optimal motion vector of $C^{x,y}$:

$$\begin{aligned} MAD_{min} =& arg \min_{dx,dy} MAD(dx,dy), \\ & \text{subject to } |dx| < \frac{W_w}{2},\ |dy| < \frac{W_h}{2}. \end{aligned} \quad (7)$$

There are $W_w \times W_h$ candidates to search and we need $\mathcal{O}(B_w \times B_h \times W_w \times W_h)$ operations to find just one motion vector.

B. 1D Matching

The matching error (MAD) of the projections of block $C^{x,y}$ and block $R^{x+dx,y+dy}$ is

$$LMAD(dx,dy) = \sum_{i=0}^{B_w-1} |LC_i^{x,y} - LR_i^{x+dx,y+dy}|. \quad (8)$$

This is a low-complexity 1D matching problem and only $\mathcal{O}(B_w)$ operations is involved per given projection, which is only $\frac{1}{B_h}$ of the 2D block matching cost.

Define $LMAD_{min}$ as

$$\begin{aligned} LMAD_{min} =& arg \min_{dx,dy} LMAD(dx,dy), \\ & \text{subject to } |dx| < \frac{W_w}{2},\ |dy| < \frac{W_h}{2}. \end{aligned} \quad (9)$$

This cost function will serve as a threshold that we can take advantage of in eliminating most of the matching candidates as described in the next section.

C. Candidate Exclusion via 1D Matching

It is intuitive that the projections of two similar blocks must be quite similar. In other words, two blocks cannot match well if their projections do not. According to the generalized triangular equality, we can obtain

$$\begin{aligned} MAD(dx,dy) &= \sum_{i=0}^{B_w-1}\sum_{j=0}^{B_h-1} |C_{i,j}^{x,y} - R_{i,j}^{x+dx,y+dy}| \\ &\geq \sum_{i=0}^{B_w-1} |\sum_{j=0}^{B_h-1} C_{i,j}^{x,y} - \sum_{j=0}^{B_h-1} R_{i,j}^{x+dx,y+dy}| \\ &= \sum_{i=0}^{B_w-1} |LC_i^{x,y} - LR_i^{x+dx,y+dy}| \\ &= LMAD(dx,dy). \end{aligned} \quad (10)$$

Hence, if $LMAD(dx,dy) > MAD_{min}$, then $MAD(dx,dy) > MAD_{min}$ and the current (dx,dy) candidate cannot be the best displacement. Usually, the majority of the candidates satisfies this $LMAD(dx,dy) > MAD_{min}$ constraint. Therefore, they can be excluded from future searches. To obtain the optimal motion vector, 2D matching is then only necessary for those candidates with $LMAD(dx,dy) \leq MAD_{min}$, which often comprise a very small percentage.

D. MAD_{min} Estimation

From the previous discussion, we see that the value MAD_{min} is necessary in the exclusion of candidates by 1D matching. Unfortunately, we do not know MAD_{min} in advance. An estimate MAD_e is used instead. The smaller the MAD_e value, the more candidates we can eliminate from 1D matching, and the faster the algorithm is. If $MAD_e \geq MAD_{min}$, the optimality of the motion vectors is still preserved. However, if $MAD_e < MAD_{min}$, we might lose optimality.

Our emprirical results show that a good estimation of MAD_{min} is based on the value of $LMAD_{min}$:

$$MAD_e = S \times LMAD_{min}, \quad (11)$$

where $S \geq 1$ is a scaling factor. A $W_w \times W_h$ buffer is necessary to keep the $LMAD(dx,dy)$ values for all candidates. This estimation method turns out to be quite robust since it is partially adaptive, i.e., at least $LMAD_{min}$ contains the information about the block.

When performing the necessary 2D matchings, the MAD of some candidate, say MAD^t , might be smaller than MAD_e. The larger S is, the more frequently this happens. This means that MAD^t is a closer estimate of MAD_{min} than MAD_e. Hence, MAD_e can be safely replaced by MAD^t without losing prediction accuracy. This way, MAD_e will approach MAD_{min} when more and more candidates are searched.

When $S = 1$, PBME performs only 1D matchings and the candidate with $LMAD_{min}$ is selected. In this case, PBME achieves the fastest speed, but unfortunately, also yields highest prediction error. When S gets larger, PBME becomes slower and prediction error decreases. By varying S, the complexity of the search can be controlled and computational scalability can be achieved.

E. Overall Algorithm

Let us recapitulate the overall procedure of the proposed projection-based BME algorithm.

Start with the top-left block $B^{0,0}$ in the current frame, scan blocks from left to right and from top to bottom. For block $B^{x,y}$,

1. Compute and buffer necessary projections using the fast projection algorithm as discussed in Section II-B.
2. Compute and buffer 1D matching error $LMAD(dx,dy)$ for all candidates as described in Section II-C.
3. Estimate MAD_e based on $LMAD_{min}$. Set $MAD_{min} = MAD_e$.
4. Scan $LMAD(dx,dy)$ one by one. If $LMAD(dx,dy) \leq MAD_e$,

(a) compute $MAD(dx, dy)$.
(b) if $MAD(dx, dy) < MAD_{min}$, set $MAD_{min} = MAD(dx, dy)$.
(c) If $MAD(dx, dy) < MAD_e$, set $MAD_e = MAD(dx, dy)$.
5. The candidate with MAD_{min} is the best matched block.

F. MSE Extension

If MSE instead of MAD is used as the matching criterion, we can still derive a fast PBME algorithm. In the MSE case, the 2D matching error is now

$$MSE(dx, dy) = \sum_{i=0}^{B_w-1} \sum_{j=0}^{B_h-1} (C_{i,j}^{x,y} - R_{i,j}^{x+dx,y+dy})^2, \quad (12)$$

and the 1D matching error is

$$LMSE(dx, dy) = \sum_{i=0}^{B_w-1} (LC_i^{x,y} - LR_i^{x+dx,y+dy})^2. \quad (13)$$

Although the condition $MSE(dx, dy) \geq LMSE(dx, dy)$ does not always hold and there is no guarantee that the candidate is not the best matched block even if $LMSE(dx, dy) > MSE_{min}$, PBME still works with MSE well since the 1D matching assumption is still quite reasonable.

G. Discussion

There are several implementation issues as well as variations of our proposed PBME algorithm.
1. In the proposed algorithm, the early termination and spiral searching concepts can be applied in both 1D matching and 2D matching to further speed up BME.
2. Only vertical projection is demonstrated in this paper since most videos have more horizontal motions. However, horizontal projection might be better for some video data.
3. A more greedy approach, trying to eliminate candidates by only matching the sums of two blocks using $PB^{x,y}$ in Eq. (3), does not work well. This can be viewed as DC matching and we can derive fast methods to compute the DC as well as to manage the 1D buffer. Although DC matching is very low cost (only one operation is involved per match operation), the sum contains too little information about a block. Our empirical data indicate that DC matching yields too many mismatches.
4. By using both vertical and horizontal projections, a few more candidates can be eliminated. However, the saving is just enough to compensate for the cost of the extra projections. Besides, extra buffers are necessary. Hence, matching two 1D projections is not any better than using just one projection.
5. Advanced methods to predict MAD_{min} can be incorporated to improve PBME further.

IV. Experimental Results

Three popular QCIF (176×144) test sequences – Foreman, News and Miss America – are used to compare the prediction performance as well as the speed of PBME with those of the full exhaustive search algorithm (FS) and the three-step search algorithm (TSS). We use a typical H.263+ [2] BME setup: *(i)* the block size is 16×16 pixels and the search window size is 32×32 pixels; *(ii)* the reference frame is symmetrically extended by 16 pixels to allow motion vectors to point outside the frame; *(iii)* the extended reference frame is bilinearly interpolated to enable half-pixel accuracy BME; *(iv)* to obtain half-pixel motion accuracy, the best matched block with whole-pixel displacement is found first and then only its 8 direct neighbors with half-pixel displacement are checked to find the approximated best matched block with half-pixel accuracy; and finally *(v)* the frame skip is 2 (every third frame is predicted).

For each block, we always apply 2D matching to the 8 candidates with half-pixel motion and the candidate with zero motion. To find one motion vector, $32 \times 32 + 8 = 1032$ and $25 + 8 = 33$ 2D matchings are needed for FS and TSS, respectively. For PBME, $32 \times 32 = 1024$ 1D matching plus at least 9 2D matchings are necessary for PBME. All algorithms has early termination incorporated. FS and PBME employ spiral searching as well. To favor the zero motion vector, the corresponding 2D matching error for zero displacement is reduced by a constant amount (100 for MAD and 1000 for MSE). PBME is tested with 4 different speed factors: $S = 1, 2, 4$, and 8.

Fig. 3 – 5 show the prediction error measured by PSNR:

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \quad (14)$$

where MSE is the mean-squared error of the prediction error. The PSNR curves for PBME with $S = 2, 4$, and 8 nearly overlap with the curves for FS indicating that the prediction performance of PBME with $S \geq 2$ is nearly identical to that of FS. However, the PSNRs for PBME with $S = 1$ are even worse than those of TSS, suggesting that the small amount of necessary 2D matchings are critical to PBME.

Fig. 6 – 8 demonstrate the percentage of whole-pixel candidates (not including the ones with zero motion) eliminated by just 1D matching. These percentages are 0% for FS, 98% for TSS, and 100% for PBME with $S = 1$. The smaller the value of S, the higher the elimination percentage, and thus the faster the PBME algorithm is. The elimination percentage drops a lot when some blocks can not be predicted well from the reference frame. Example of such scenario can be found in frames 250 to 350 of Foreman, where the camera is panning quickly, and in frames 90, 151 and 241 of News, where there exists a subsequence with frequent scene change.

Table I – III tabulate the overall PSNRs and elimination percentages. The actual executing speed (including file I/O and bi-linear interpolation for half-pixel ME) in frames per second (fps) based on a DELL Dimension L600r PIII 600Mhz workstation with 128M RAM running RedHat Linux 6.1 is also presented. We can observe that when the elimination percentage is less than 97%, the performance of PBME is practically identical to that of FS, while PBME

is about 3 to 5 times faster than FS. When $S = 2$, PBME is slightly slower but it outperforms TSS by around 0.6 dB. The prediction performance only degrades slightly when S reduces from 8 to 2. However, the computational complexity decreases drastically.

V. Conclusion

We have presented a projection-based fast BME algorithm called PBME in this paper. An efficient method to compute and to buffer projection data is also described. The algorithm greatly reduces the computational complexity of BME while maintaining prediction integrity since most candidates can be quickly eliminated by matching 1D projections, which is much faster than matching 2D blocks. The prediction performance is close to the global optimum and the executing speed is several times faster than the full search algorithm. All-software real-time high-performance encoding is certainly practical for QCIF-size videos. Computational scalability, which is often difficult to achieve in other algorithms, can be easily obtained by controlling one single scaling parameter.

References

[1] "Information technology - generic coding of moving pictures and associated audio information," *ISO/IEC JTC1 IS 13818-2 (MPEG-2)*, 1996.

[2] G. Cote, B. Erol, M. Gallant, and F. Kossentini, "*H*.263+: Video coding at low bit rates," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, NO. 7, pp. 849–866, Nov. 1998.

[3] MPEG-4 Video Group, "Generic coding of audio-visual objects: Part 2 – visual," *ISO/IEC JTC1/SC29/WG11 N1902, FDIS of ISO/IEC 14496-2*, Nov. 1996.

[4] "Telenor H.26L porposal and software," *ITU Q.15/16 Q15-H-08*, 1999.

[5] M. Brünig and B. Menser, "The mean square error as a matching criterion for efficient motion estimation," *in Proc. IEEE Int. Workshop on Intelligent Signal Processing and Communication Systems ISPACS'98*, Nov. 1998.

[6] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Processing*, vol. 4, pp. 105–107, Jan. 1995.

[7] M. Brünig and B. Menser, "Fast full search block matching using subblocks and successive approximation of the error measure," *in Proc. SPIE Image and Video Communications and Processing*, vol. 3974(2000), pp. 235–244, Jan. 2000.

[8] T. Koga, "Motion compensated unterframe coding for video conferencing," *NTC '81, National Telecommun. Conf.*, pp. G.5.3.1–G.5.3.5, Dec. 1981.

[9] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Image Process.*, vol. COM-29, pp. 1799–1808, Dec. 1981.

[10] S. Kumar, "A simple FPGA-based conjugate search motion estimator," *IEEE Asia-Pacific Conf. on Circuits and Systems*, pp. 109–114, Dec. 1994.

[11] K. W. Chun and J.B. Ra, "An improved block matching algorithm based on successive refinement of motion vector candidates," *Signal Proc: Image Communications*, vol. 6, pp. 115–122, 1994.

[12] Y. Zhang and S. Zafar, "Motion-compendated wavelet transform coding for color video compression," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 2, pp. 285–296, Sept. 1992.

[13] "Information technology - coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbits/s," *ISO/IEC 1172-2, Part 2*, 1993.

[14] S. Zafar, Y. Zhang, and J. S. Baras, "Predictive block-matching motion estimation for TV coding - Part I: inter-block prediction," *IEEE Trans. Broadcasting*, vol. 37, pp. 97–101, Sep. 1991.

[15] P. Milanfar, "A model of the effect of image motion in the Radon transform domain," *IEEE Transactions on Image Processing*, vol. 8(3), pp. 1276–1281, Sept. 1999.

[16] R. H. Park J. S. Kim, "A fast feature-based block matching algorithm using integral projections," *IEEE J. Selected Areas in Communications*, vol. 10(5), pp. 968–971, Jun. 1992.

[17] S. C. Tai Y. C. Lin, "Fast full-search block-matching algorithm for motion-compensated video compression," *IEEE Transactions on Communications*, vol. 45(5), pp. 527–531, May 1997.

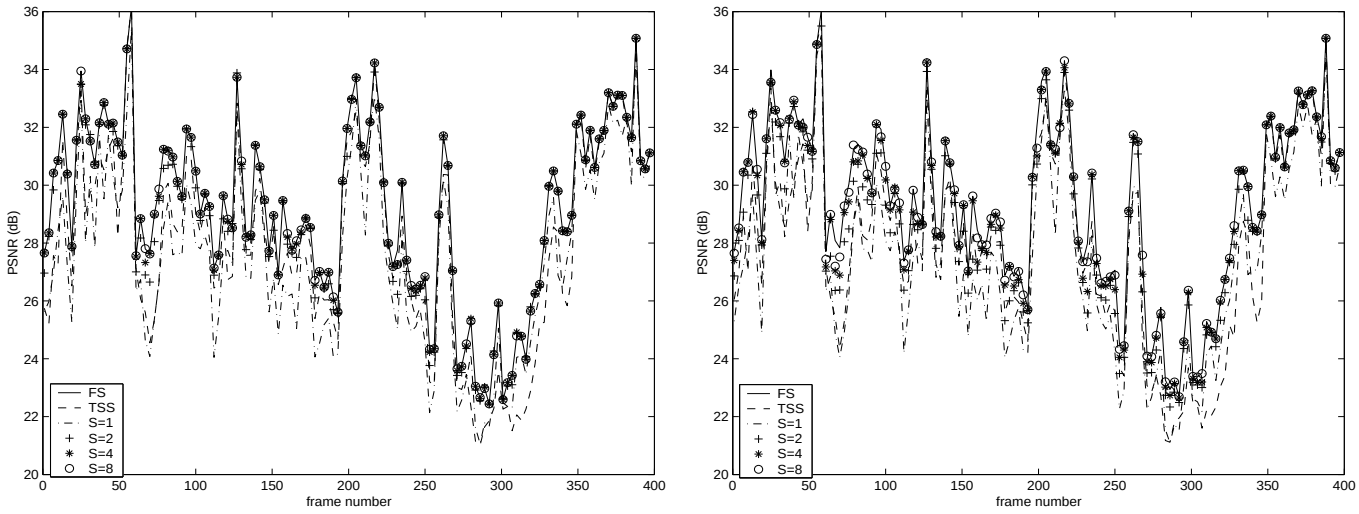

Fig. 3. PSNR per frame of Foreman produced by FS, TSS and PBME with S=1, 2, 4 and 8: MAD (left) and MSE (right).

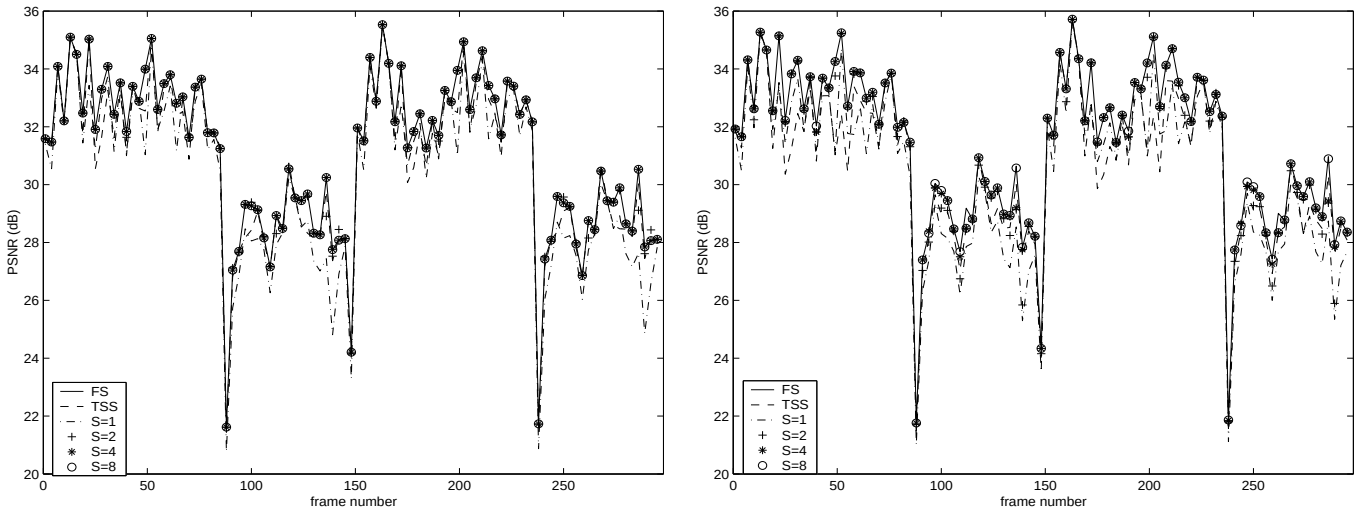

Fig. 4. PSNR per frame of News produced by FS, TSS and PBME with S=1, 2, 4 and 8: MAD (left) and MSE (right).

TABLE I
SEARCH RESULTS FOR FOREMAN

			PBME				
		FS	S=8	S=4	S=2	S=1	TSS
MAD	PSNR (dB)	29.1417	29.1390	29.1230	28.9674	27.5385	28.0046
	elimination %	0.0	85.0143	91.3697	97.2273	100.0	98.0
	speed (fps)	8.4	18.7	26.7	42.2	78.6	92.3
MSE	PSNR (dB)	29.3425	29.2877	29.1660	28.8040	27.6041	28.0007
	elimination %	0.0	97.0785	98.8773	99.6695	100.0	98.0
	speed(fps)	8.5	39.9	45.7	53.9	72.4	98.7

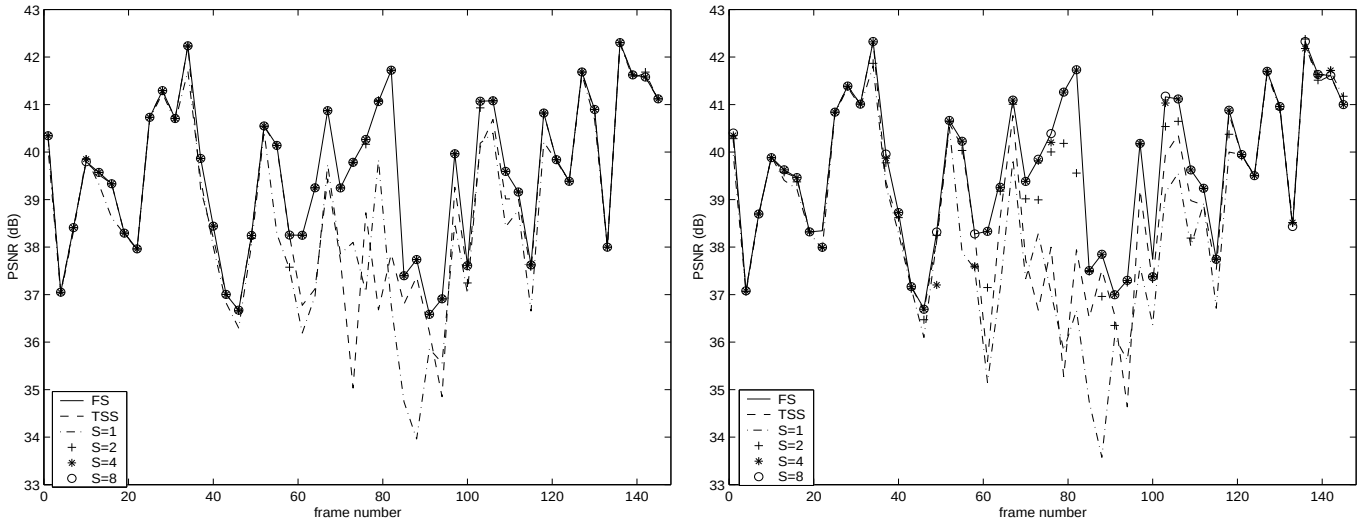

Fig. 5. PSNR per frame of Miss America produced by FS, TSS and PBME with S=1, 2, 4 and 8: MAD (left) and MSE (right).

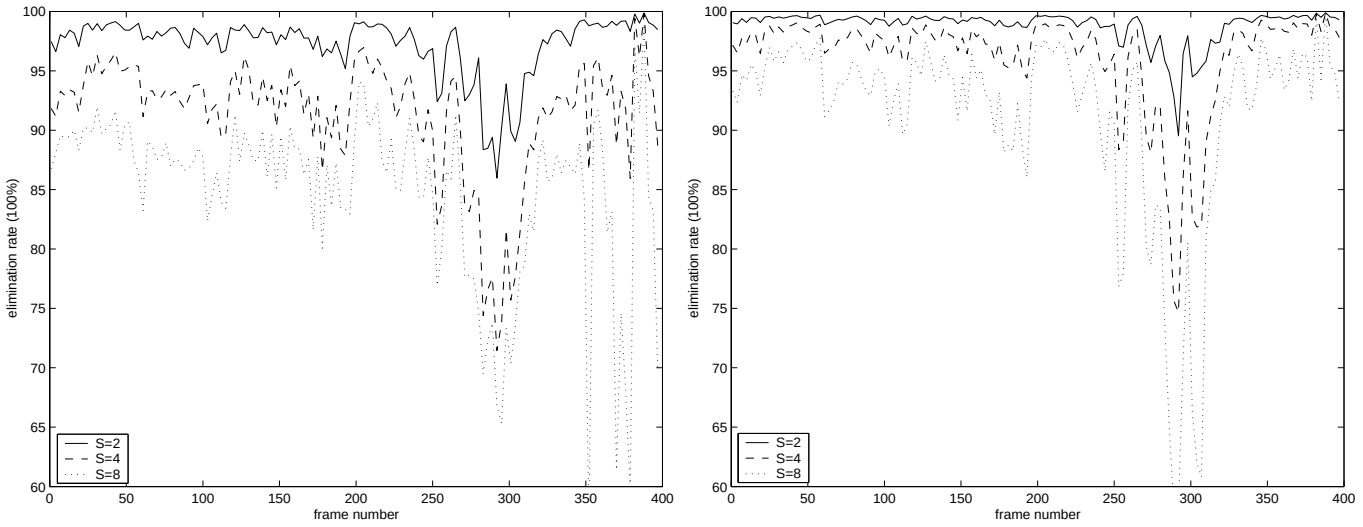

Fig. 6. Elimination percentages for Foreman: MAD (left) and MSE (right).

TABLE II
SEARCH RESULTS FOR NEWS

		FS	PBME				TSS
			S=8	S=4	S=2	S=1	
MAD	PSNR (dB)	31.0439	31.0440	31.0442	31.0061	30.2675	30.7194
	elimination %	0.0	95.4508	97.7903	99.2848	100.0	98.0
	speed (fps)	15.4	39.7	46.7	54.5	88.6	98.9
MSE	PSNR (dB)	31.3363	31.3241	31.2769	31.1087	30.3499	30.6778
	elimination %	0.0	97.0785	98.8773	99.6659	100.0	98.0
	speed(fps)	16.3	37.9	42.7	53.9	86.4	103.7

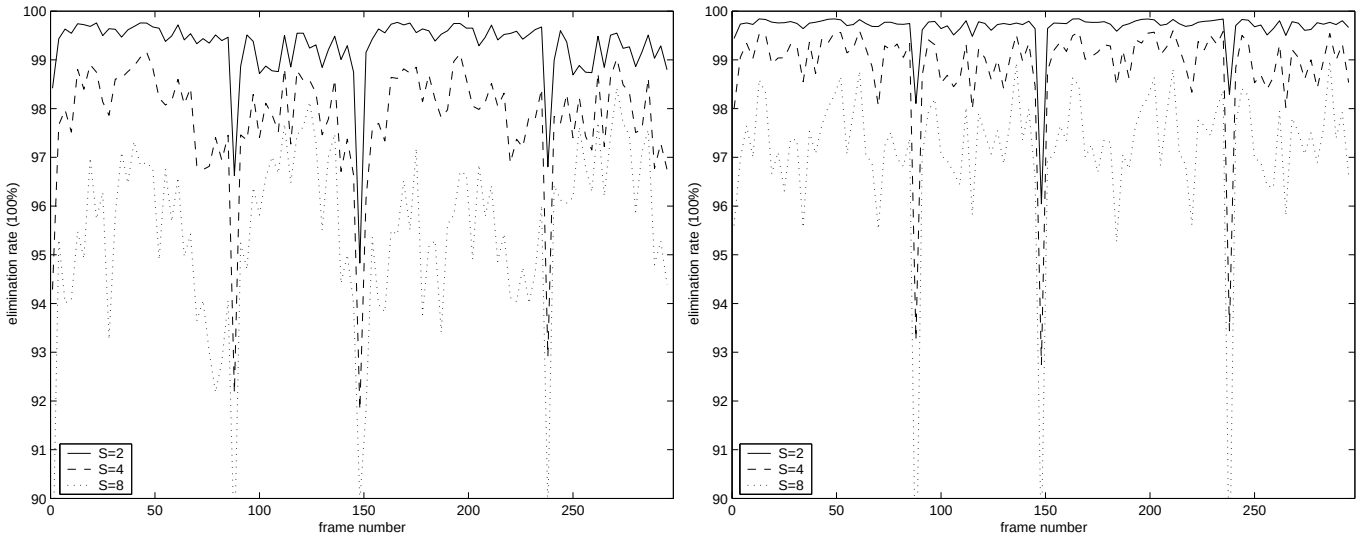

Fig. 7. Elimination percentages for News: MAD (left) and MSE (right).

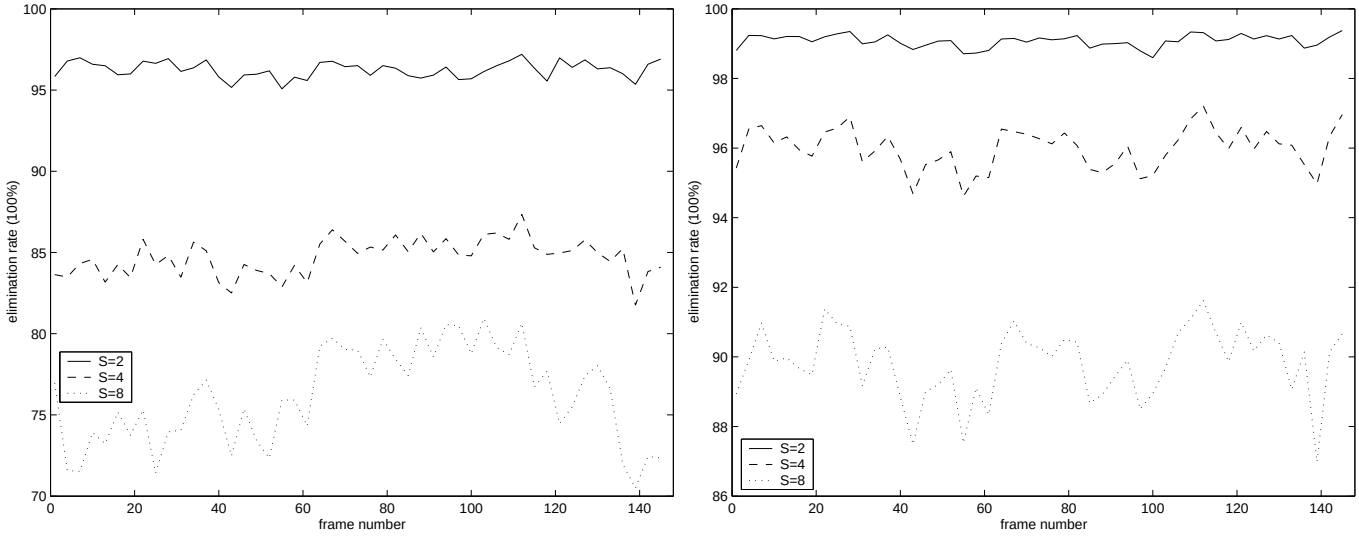

Fig. 8. Elimination percentages for Miss America: MAD (left) and MSE (right).

TABLE III
SEARCH RESULTS FOR MISS AMERICA

			PBME				
		FS	S=8	S=4	S=2	S=1	TSS
MAD	PSNR (dB)	39.5372	39.5372	39.5380	39.5118	38.7240	38.9515
	elimination %	0.0	76.1429	84.7034	96.2572	100.0	98.0
	speed (fps)	7.9	16.7	28.7	34.2	71.5	91.4
MSE	PSNR (dB)	39.6597	39.6417	39.5950	39.3768	38.5936	39.0133
	elimination %	0.0	89.8182	95.9890	99.0785	100.0	98.0
	speed(fps)	6.2	26.9	35.7	42.9	68.4	93.5