Multiple Block-Size Search Algorithm for Fast Block Motion Estimation

Ka-Ho Ng, Lai-Man Po, Ka-Man Wong, and Chi-Wang Ting
Department of Electronic Engineering,
City University of Hong Kong,
Hong Kong SAR, China

Kwok-Wai Cheung
Department of Computer Science,
Chu Hai College of Higher Education,
Hong Kong SAR, China

***Abstract*—Although variable block-size motion estimation provides significant video quality and coding efficiency improvement, it requires much higher computational complexity compared with fixed block size motion estimation. The reason is that the current motion estimation algorithms are mainly designed for fixed block size. Current variable block-size motion estimation implementation simply applies these existing motion estimation algorithms independently for different block sizes to find the best block size and the corresponding motion vector. Substantial computation is wasted because distortion data reuse among motion searches of different block sizes is not considered. In this paper, a motion estimation algorithm intrinsically designed for variable block-size video coding is presented. The proposed multiple block-size search (MBSS) algorithm unifies the motion searches for different block sizes into a single searching process instead of independently performing the search for each block size. In this unified search, the suboptimal motion vectors for different block sizes are used to determine the next search steps. Its prediction quality is comparable with that obtained by performing motion search for different block sizes independently while the computational load is substantially reduced. Experimental results show that the prediction quality of MBSS is similar to full search.**

Block matching, motion estimation, video coding, search pattern, directional search.

I. INTRODUCTION

In fixed block size motion estimation (FBSME) algorithms, a video frame is divided into non-overlapping block of equal size. The best-matched block to the current coding block is determined within a predefined search window from reference frames. In the latest video coding standards such as H.264/AVC [1], variable block size motion estimation (VBSME) is adopted to improve coding quality and efficiency significantly. However, the computational complexity required by VBSME is much higher than that of FBSME. In H.264/AVC, tree-structured block sizes are employed. Each 16×16 macroblock (MB) can be coded in 16×16, 16×8, 8×16, or 8×8 block modes. An 8×8 block may be further divided into 8×4, 4×8, or 4×4 sub-blocks. There are totally 41 sub-blocks as shown in Fig. 1. In general, large block sizes are suitable for homogenous area with slow motions while smaller block sizes are preferred for area with complex motions. By using VBSME, motion search will be performed for each of the 41 sub-block sizes. An optimum motion vector (MV) and the corresponding

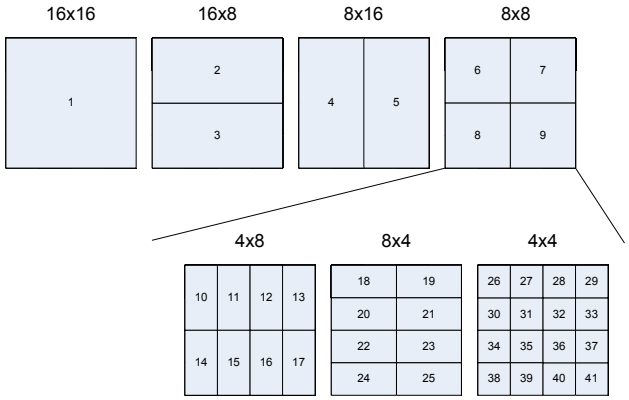

Figure 1. Macroblock partitioning in H.264.

optimum sub-block mode will be found among all different sub-block size searches. Three VBSME algorithms -- full search (FS), fast full search (FFS), and unsymmetrical-cross multi-hexagon-grid search (UMHexagonS) [2] -- are adopted in H.264 reference software (JM9.6) [3]. In FS, motion estimation for the 41 sub-blocks are performed independently [4]. Exhaustive search is performed for each sub-block in the search window. As there are seven block size modes as shown in Fig.1, about seven times the computational complexity of FS on fixed 16×16 block size is required. FFS is implemented to decrease the computational complexity of FS. In FFS, the sum of absolute differences (SADs) of the 4×4 sub-blocks are stored and the SADs of larger sub-blocks are obtained by summing up these stored SADs of smaller sub-blocks. By reuse of distortion data, FFS reduces the computation complexity of VBSME. UMHexagonS [2, 5] utilizes starting search point prediction, early termination, and multiple search patterns to further speedup the motion search of VBSME.

Two approaches are commonly adopted by fast VBSME algorithms. In the first approach, more efficient search patterns or adaptive patterns are proposed to speedup the motion search for each sub-block in VBSME. For example, cross diamond search (CDS) [6] utilizes the cross-center biased MV distribution. UMHexagonS makes use of the assumption that the movement in the horizontal direction is much heavier than that in the vertical direction for natural video sequences. Accordingly, more search points are applied in horizontal direction than in vertical direction in its uneven multi-hexagon-

grid search pattern [2]. Although the algorithms using this approach successfully reduce the computation complexity of motion search of each sub-block and thus speedup the overall FBSME process to a certain extent, the MV correlations between different block modes are not utilized.

Another approach utilizes the MV correlations between different block modes. The MVs of some motion searched sub-blocks are used to predict the MVs of other sub-blocks. Then, motion search is skipped for some of the sub-blocks or simpler search patterns in a smaller search window can be used. For example, in [7], the MVs of the larger sub-blocks are merged from the MVs of the smaller sub-blocks using a threshold related to the quantization parameter. The algorithm in [8] first applies diamond search to each of the sixteen 4×4 sub-block. MVs obtained for the 4x4 sub-blocks are used to calculate the initial MVs of larger sub-blocks. These initial MVs of larger sub-blocks are further refined using large or small diamond search patterns. This approach utilizes the MV correlations among different block modes. However, each sub-block has its own search path and the distortion data of a sub-block motion search is not always usable in another sub-block motion search. Thus, substantial computation may be wasted.

To maximize distortion data reuse, a multiple block-size search (MBSS) algorithm is proposed in this paper. The algorithm unifies the motion searches for different block sizes into a single motion search process. Full distortion data reuse can be achieved and computational complexity is thus reduced substantially. A novel stopping criterion and filled search pattern are used to ensure that all the 41 MVs are pointing to a local or global distortion minimum during convergence.

II. Multiple Block-Size Search Algorithm

MBSS employs a single search window with multiple search paths for all sub-blocks. For each search point, the sixteen 4×4 SADs of the 4×4 sub-blocks are first calculated and stored (Fig. 2). The SADs of the remaining 25 sub-blocks (4×8, 8×8, 16×8, etc in Fig. 1) for the same search point are obtained by summing up the SADs of smaller sub-blocks as shown in Fig. 3. A compact square search pattern with a novel stopping criterion is used in MBSS. At the beginning, the center of the search window is considered as a *search center*. The searching starts with center of the square search pattern positioned at the *search center* (Fig. 4). For each search point in the pattern, the SADs of the 41 sub-blocks are found as described before. The minimum distortion points for all sub-blocks are determined. The searching then continues by locating the *next search centers* which are the minimum distortion points located at the boundary of the searched region. The searched region refers to all the previously searched points. Then, the square search pattern is applied at each *next search center*. The positions of the 41 minimum distortion points are updated accordingly. Based on the updated minimum distortion points, new *next search centers* are found. This process repeats until there are no more *next search centers*. That means the motion search converges and the minimum distortion points for all sub-blocks are located. As the MVs of different block types are enclosed by at least eight searched points with higher SADs, they are guaranteed to be pointing to global or local minimum distortion points.

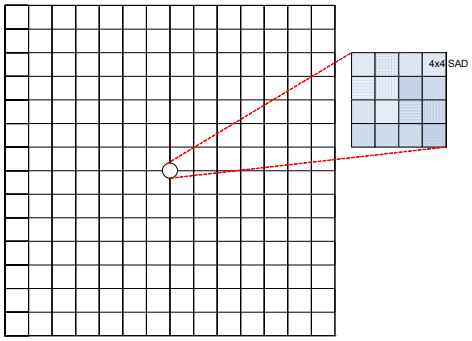

Figure 2. Sixteen 4×4 SADs are first calculated when one search point is searched.

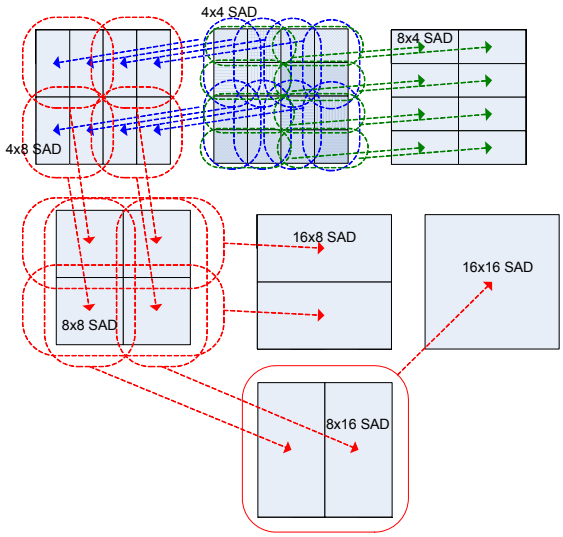

Figure 3. SADs of larger sub-blocks are obtained by summing up the SADs of smaller sub-blocks.

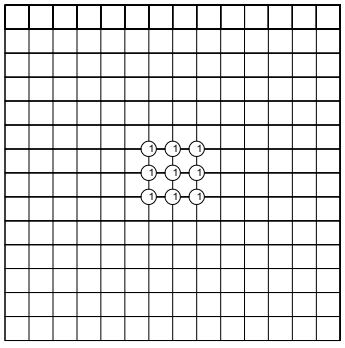

Figure 4. Initial search pattern.

Since there may be more than one *next search center* in the intermediate search steps, MBSS is in effect a multiple paths searching similar to multi-path search (MPS) algorithm [9] which is a modification of the block-based gradient descent search (BBGDS) [10]. Thus, MBSS is also a multiple path search algorithm with better prediction quality than BBGDS as it can reduce the chance of being trapped in a local minimum by considering multiple descending gradient paths.

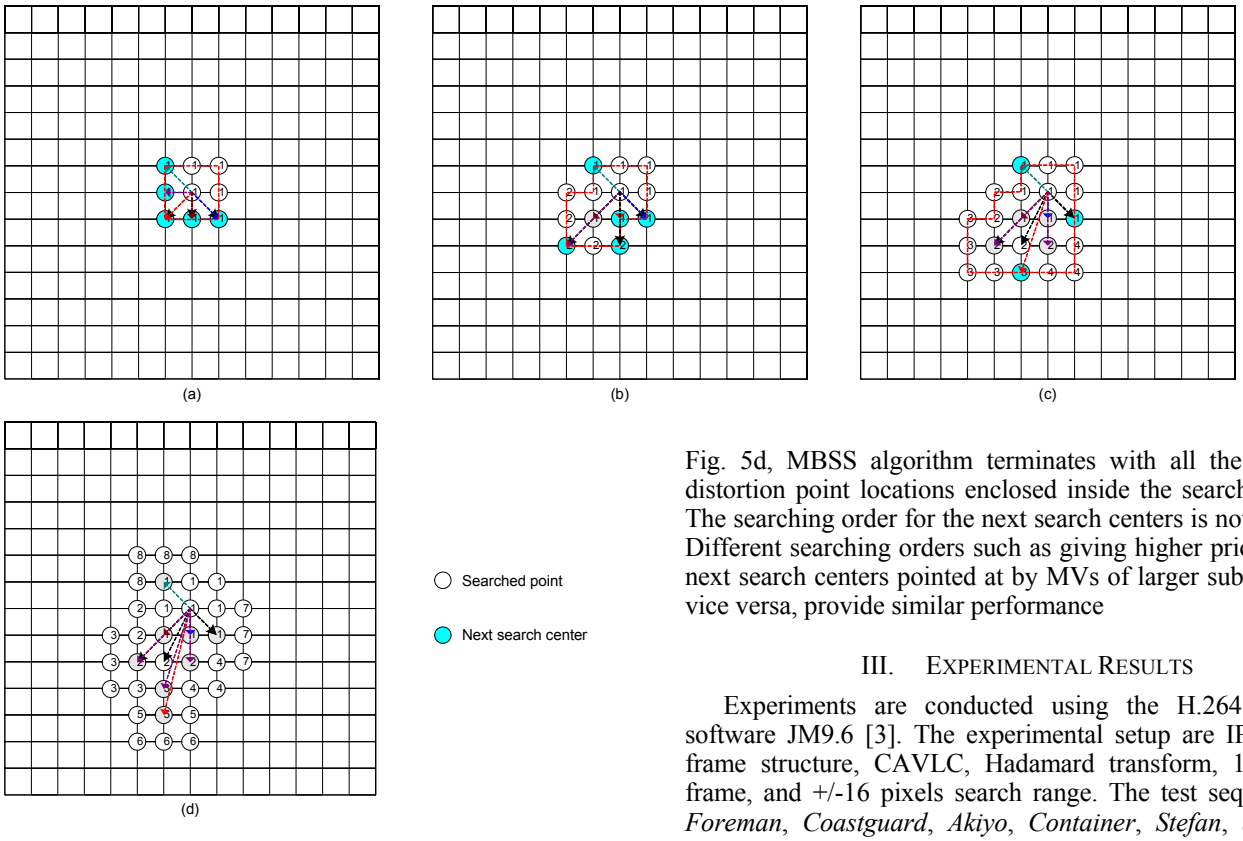

Figure 5. (a) MBSS with five next search centers. (b) Positioning the search pattern over one of the next search centers. (c) Searched region after 4 square pattern positioning. (d) MBSS convergence - all minimum distortion points are enclosed inside the searched region.

The MBSS algorithm is summarized as follows.

Step 1: Position the square search pattern at the search window center. Search all points inside the pattern for *minimum distortion points* of all sub-blocks.

Step 2: Set the *minimum distortion points* located at the boundary of the searched region as the set of *next search centers*. If the *next search centers* set is empty, go to Step 4. Otherwise go to Step 3.

Step 3: Position the square search pattern over the points in the *next search centers* set one-by-one. Search the additional points covered by the patterns. Update the *minimum distortion points* if lower distortion points are found. Empty the *next search centers* set and then go to Step 2.

Step 4: The algorithm converges and the 41 MVs are determined.

Fig. 5 shows a MBSS example. After Step 1 & 2, there are five next search centers as shown in Fig. 5a. The square search pattern is positioned over the next search center at the lower left corner of the searched region in Fig. 5b. After four times square pattern positioning, the searched region contains seven locations for the 41 minimum distortion points and three of them are next search centers as shown in Fig. 5c. The remaining four minimum distortion point locations are enclosed inside the searched region, i.e. not at the boundary. In Fig. 5d, MBSS algorithm terminates with all the minimum distortion point locations enclosed inside the searched region. The searching order for the next search centers is not important. Different searching orders such as giving higher priority to the next search centers pointed at by MVs of larger sub-blocks, or vice versa, provide similar performance

III. Experimental Results

Experiments are conducted using the H.264 reference software JM9.6 [3]. The experimental setup are IPPPIPPP… frame structure, CAVLC, Hadamard transform, 1 reference frame, and +/-16 pixels search range. The test sequences are *Foreman*, *Coastguard*, *Akiyo*, *Container*, *Stefan*, and *News*. They are in size CIF (352x288). 100 frames of each sequence are used. The sub-pel motion search in H.264 reference software is disabled so that the experimental results can clearly reflect the performances of the algorithms.

A. Search Window Center of MBSS

In H.264, the predicted MV of each sub-block is calculated by its adjacent sub-blocks and is to determine the search window center for that sub-block. Thus, it is possible to have 41 different search window centers. In MBSS, a unified motion search using a single search window center is performed for all sub-blocks. A simple approach is to use the predicted MV of one of the block mode as the search window center. Experiments are performed to find the percentages of the same predicted MVs between different block modes, using the full search in H.264 reference software.

Table I to V give the percentages of predicted MVs of different block modes that are the same as the predicted MVs of 16×16, 16×8, 8×16, 8×8, and 4×4 block modes respectively. In Table II, '16×8a' and '16×8b' refer to the upper and lower 16×8 blocks respectively. In Table III, '8×16a' and '8×16b' refer to the left and the right 8×16 blocks respectively. It is observed that a high percentage of predicted MVs of different block modes are the same as the predicted MVs of 16×16 block mode. Therefore, the predicted MV of the 16×16 block mode is used as the single search window center in MBSS.

B. Performance Comparison with Other VBSME Algorithms

MBSS algorithm is compared with full search (FS), fast full search (FFS), diamond search (DS) [11], and UMHexagonS in JM9.6. The average PSNR (dB) and bit-rate (kbits/s) are used for video quality evaluation. The computational complexity is

TABLE I
PERCENTAGES OF PREDICTED MVS BEING THE SAME AS THE PREDICTED MVS OF THE 16×16 BLOCK MODE

	16x16	16x8 a	16x8 b	8x16 a	8x16 b	8x8	8x4	4x8	4x4
Foreman	-	41.6%	69.3%	52.8%	46.3%	55.7%	49.5%	49.9%	47.9%
Coastguard	-	40.8%	65.8%	56.8%	51.8%	52.7%	46.3%	45.0%	41.5%
Akiyo	-	87.1%	93.1%	88.6%	88.3%	89.7%	85.8%	84.9%	84.4%
Container	-	88.5%	94.9%	91.5%	92.8%	90.8%	86.9%	86.5%	85.7%
Stefan	-	42.8%	74.2%	59.7%	53.9%	61.3%	54.9%	56.0%	52.8%
News	-	81.7%	89.4%	85.0%	83.3%	84.9%	81.4%	80.1%	79.5%

TABLE II
PERCENTAGES OF PREDICTED MVS BEING THE SAME AS THE PREDICTED MVS OF ONE OF THE 16×8 BLOCK MODE

	16x16	16x8 a	16x8 b	8x16 a	8x16 b	8x8	8x4	4x8	4x4
Foreman	41.6%	-	33.1%	70.1%	22.5%	36.7%	46.1%	29.5%	30.1%
Coastguard	40.8%	-	31.3%	53.7%	24.9%	37.9%	43.6%	28.9%	29.6%
Akiyo	87.1%	-	83.1%	94.3%	79.9%	83.6%	84.2%	78.3%	78.5%
Container	88.5%	-	84.8%	94.4%	83.0%	85.1%	85.0%	80.2%	80.5%
Stefan	42.8%	-	35.9%	61.8%	29.7%	42.9%	50.1%	37.2%	37.7%
News	81.7%	-	77.5%	90.2%	73.5%	77.3%	79.1%	72.6%	72.7%

TABLE III
PERCENTAGES OF PREDICTED MVS BEING THE SAME AS THE PREDICTED MVS OF ONE OF THE 8×16 BLOCK MODE

	16x16	16x8 a	16x8 b	8x16 a	8x16 b	8x8	8x4	4x8	4x4
Foreman	52.8%	70.1%	40.2%	-	24.5%	41.3%	49.4%	33.7%	34.1%
Coastguard	56.8%	53.7%	39.7%	-	27.2%	41.3%	45.4%	32.4%	32.3%
Akiyo	88.6%	94.3%	84.2%	-	80.6%	84.3%	84.6%	78.9%	79.2%
Container	91.5%	94.4%	87.6%	-	84.8%	86.9%	86.2%	81.8%	82.0%
Stefan	59.7%	61.8%	46.2%	-	31.0%	47.7%	52.6%	41.1%	40.6%
News	85.0%	90.2%	79.6%	-	74.8%	78.9%	80.2%	73.9%	74.1%

TABLE IV
PERCENTAGES OF PREDICTED MVS BEING THE SAME AS THE PREDICTED MVS OF ONE OF THE 8×8 BLOCK MODE

	16x16	16x8 a	16x8 b	8x16 a	8x16 b	other 8x8s	8x4	4x8	4x4
Foreman	43.3%	35.8%	44.1%	38.0%	28.7%	56.9%	58.8%	52.8%	51.1%
Coastguard	37.6%	34.2%	33.3%	34.2%	28.7%	49.0%	52.9%	45.9%	43.2%
Akiyo	83.8%	80.8%	83.6%	81.1%	78.3%	87.0%	86.0%	83.8%	83.5%
Container	84.3%	81.8%	83.5%	82.7%	80.6%	87.7%	86.8%	84.6%	84.2%
Stefan	50.4%	42.7%	49.1%	43.5%	38.9%	62.1%	63.5%	61.1%	58.5%
News	78.6%	73.7%	78.8%	75.0%	72.5%	83.1%	82.7%	80.2%	79.7%

TABLE V
PERCENTAGES OF PREDICTED MVS BEING THE SAME AS THE PREDICTED MVS OF ONE OF THE 4×4 BLOCK MODE

	16x16	16x8 a	16x8 b	8x16 a	8x16 b	8x8	8x4	4x8	other 4x4s
Foreman	33.5%	23.4%	36.1%	24.4%	26.8%	43.3%	37.5%	46.7%	46.0%
Coastguard	72.3%	67.0%	72.6%	68.0%	68.7%	75.5%	72.0%	76.1%	75.8%
Akiyo	76.7%	72.3%	76.8%	72.6%	73.2%	79.0%	75.3%	79.7%	79.1%
Container	77.7%	74.9%	77.2%	75.2%	75.3%	80.1%	76.9%	79.9%	79.5%
Stefan	41.4%	34.6%	41.1%	33.6%	36.0%	50.0%	46.1%	50.7%	50.9%
News	27.4%	23.9%	25.0%	23.3%	23.1%	33.5%	30.3%	34.5%	34.8%

measured in the total motion estimation time of the H.264 reference software. Table VI and VII show the performance comparison among the different algorithms for various sequences. It can be seen that MBSS is the fastest algorithm. For sequences with small motions, i.e. *Akiyo*, *News* and *Container*, it has 16.96%, 22.79%, and 26.27% speedup over UMHexagonS respectively. For sequences with large or more complex motions, i.e. *Foreman*, *Coastguard*, and *Stefan*, MBSS gives a substantial speedup over UMHexagonS from 35.72% to 48.36%. With regard to the video coding quality, MBSS provides similar PSNR as UMHexagonS does.

IV. CONCLUSION

A new fast VBSME algorithm MBSS, which can achieve full distortion data reuse, is proposed. MBSS unifies the motion searches of different block modes into a single motion search process. Experiments show that MBSS has coding quality close to full search and is 17% to 50% faster than UMHexagonS.

TABLE VI
PERFORMANCE COMPARISON BETWEEN FS, FFS, DS, UMHEXAGONS, AND MBSS USING *FOREMAN*, *COASTGUARD*, AND *AKIYO*

	Foreman			*Coastguard*			*Akiyo*		
	PSNR (dB)	bitrate (kbits/s)	run time (sec)	PSNR (dB)	bitrate (kbits/s)	run time (sec)	PSNR (dB)	bitrate (kbits/s)	run time (sec)
FS	36.69	877.49	370.63	34.45	1896.58	373.88	40.00	404.33	371.86
FFS	36.69	887.75	47.92	34.44	1899.63	46.61	40.00	405.50	47.91
DS	36.64	898.39	17.28	34.44	1894.41	17.26	39.97	404.59	17.33
UMHexagonS	36.68	878.75	14.07	34.44	1892.34	17.43	39.99	404.38	10.27
MBSS	36.65	894.45	9.04	34.45	1896.41	9.00	39.97	404.58	8.53
Bitrate changes over UMHexagonS	1.79%			0.22%			0.05%		
Speedup over UMHexagonS	35.72%			48.36%			16.96%		

TABLE VII
PERFORMANCE COMPARISON BETWEEN FS, FFS, DS, UMHEXAGONS, AND MBSS USING *CONTAINER*, *STEFAN*, AND *NEWS*

	Container			*Stefan*			*News*		
	PSNR (dB)	bitrate (kbits/s)	run time (sec)	PSNR (dB)	bitrate (kbits/s)	run time (sec)	PSNR (dB)	bitrate (kbits/s)	run time (sec)
FS	36.50	849.04	372.36	35.22	2550.76	490.68	38.35	698.41	373.35
FFS	36.49	852.82	47.79	35.22	2558.04	47.66	38.35	699.79	48.86
DS	36.49	849.05	16.56	35.21	2570.62	17.44	38.34	699.39	16.77
UMHexagonS	36.49	851.35	11.54	35.21	2553.82	15.92	38.35	697.99	11.16
MBSS	36.49	849.06	8.51	35.20	2580.87	9.25	38.34	699.32	8.62
Bitrate changes over UMHexagonS	-0.27%			1.06%			0.19%		
Speedup over UMHexagonS	26.27%			41.89%			22.79%		

REFERENCES

[1] ITU-T_SG16_Q.6 and ISO/IEC_JTC1/SC29/WG11, "Advanced video coding for generic audiovisual services," *ITU-T Recommendation H.264 | ISO/IEC International Standard ISO/IEC 14496-10,* May 2003.

[2] Z. B. Chen, P. Zhou, and Y. He, "Fast Integer Pel and Fractional Pel Motion Estimation for JVT," in *JVT-F017, 6th Meeting: Awaji, Island, Japan, 5-13 December, 2002*.

[3] "Joint Video Team (JVT) reference software version 9.6 [Online]. Available: http://iphome.hhi.de/suehring/tml/download/old_jm/."

[4] Y. Song, Z. Liu, T. Ikenaga, and S. Goto, "Ultra Low-Complexity Fast Variable Block Size Motion Estimation Algorithm in H.264/AVC," in *IEEE International Conference on Multimedia and Expo*, 2007, pp. 376-379.

[5] Z. B. Chen, P. Zhou, and Y. He, "Fast motion estimation for JVT," in *JVT-G016, 7th Meeting: Pattaya II, Thailand, 7-14 March, 2003*.

[6] C. H. Cheung and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 12, pp. 1168-177, Dec. 2002.

[7] Y. K. Tu, J. F. Yang, Y. N. Shen, and M. T. Sun, "Fast variable-size block motion estimation using merging procedure with an adaptive threshold," in *Proceedings of ICME*, Baltimore, USA, 2003, pp. 789-92.

[8] Z. Yang, J. Bu, C. Chen, and X. Li, "Fast predictive variable-block-size motion estimation for H.264/AVC," in *Proceedings of ICME*, 2005, p. 4.

[9] S. Goel and M. A. Bayoumi, "Multi-path search algorithm for block-based motion estimation," in *Proceedings of IEEE International Conference on Image Processing*, Atlanta, USA, 2006, pp. 2373-2376.

[10] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 6, pp. 419-422, Aug 1996.

[11] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 8, pp. 369-377, Aug. 1998.