CONTENT-AWARE FAST MOTION ESTIMATION ALGORITHM

Yi-Wen Chen, Ming-Ho Hsiao, Hua-Tsung Chen, Chi-Yu Liu, Suh-Yin Lee
College of Computer Science, National Chiao Tung University
E-Mail: {ewchen, mhhsiao huatsung, liucy, sylee}@csie.nctu.edu.tw

ABSTRACT

In this paper, we propose the Content-Aware Fast Motion Estimation Algorithm (CAFME) that reduces computation of motion estimation (ME) while maintains almost the same coding efficiency. Motion estimation can be divided into two phases, searching phase and matching phase. In searching phase, we propose the Simple Dynamic Search Range algorithm (SDSR) based on video characteristics to reduce the number of search points (SP). In matching phase, we integrate the Successive Elimination Algorithm (SEA) and the integral frame to develop a new SEA for H.264/AVC video compression standard, called Successive Elimination Algorithm with Integral Frame (SEAIF). Besides, based on sum of absolute difference (SAD), we also propose the Early Termination Algorithm (ETA) to terminate motion estimation of current block early.

We implement in H.264/AVC reference software JM9.4 and the experimental results show that our proposed algorithm can reduce the number of search points about 93.1%, encoding time about 42%, while maintains almost the same bitrate and PSNR

1. Introduction

Block matching based motion estimation (ME) and compensation is a fundamental process in international video compression standards, such as MPEG-1, MPEG-2, MPEG-4, ITU-T H.263, and H.264, which can efficiently remove temporal redundancy. Since a ME module is usually the most computational intensive part in a typical video encoder (about 50%~90% of the entire system), the efficient ME module is needed.

In recent years, many fast motion estimation algorithms have been proposed. We divide these algorithms into three categories. The first one is to follow some search patterns, the second one is to reduce matching complexity, and the last one is to adjust search window size. The traditional fast motion estimation algorithms, like Three-Step Search (TSS) [1] and Diamond Search (DS) [2], are classified into the first category. They usually cannot perform well for all kinds of motion activity. The pixel decimation algorithm must determine the tradeoff between accuracy and computational cost in block matching. The Successive Elimination Algorithm (SEA) [3] is a lossless approach. It can avoid unnecessary SAD computation. However, it may suffer from substantial overhead, complex hardware design and coding efficiency degradation. The Window Follower Algorithm (WFA) [7] is classified into the third category algorithm which can reduce the number of search points but it needs thresholds and is not suitable for sudden motion change.

Because the drawbacks of previous works, we propose the Content-Aware Fast Motion Estimation (CAFME) algorithm to overcome these drawbacks. The CAFME consists of the Simple Dynamic Search Range algorithm (SDSR), Successive Elimination Algorithm with Integral Frame (SEAIF), and Early Termination algorithm (ETA). The SDSR adjusts search range adaptively according to motion activity and performs well regardless of low or high motion. The SEAIF is designed for H.264/AVC visual compression standard and the ETA terminates the search process if the up-to-date block is good enough. Although the CAFME consists of the SDSR, SEAIF, and ETA, these three algorithms can be used independently. The experimental result shows that the proposed SDSR can find a very good search range for each block and maintain almost the same coding efficiency compared with Full Search.

The paper is organized as follows. We present the details of the there parts, SDSR, SEAIF and ETA of the proposed Content-Aware Fast Motion Estimation Algorithm in section2, section3 and section4, respectively. Section 5 reports the significant experimental results. Finally, the conclusions are given in section 6

2. Simple Dynamic Search Range (SDSR)

Due to the variation of coded video type, we can adjust the best suitable SR for a frame or a block in motion estimation such that the true MVs can be found, the local minimum problem can be avoided and the computational cost of motion estimation can be reduced dramatically.

In order to adjust search range for motion estimation, some approaches have already been implemented in DSWA [5], AFSBM [6], MWFA [8], and MAS [9]. These approaches may be classified into block matching error based and motion vector based. The block matching error is usually measured in MSD, MAD or SAD. The block matching error represents the degree of matching between current block and candidate block. The value of block matching error is determined by many factors including motion activity, texture, and quantization parameter. See figure1, for example. From frame 220, the values of SAD are much higher than the rest. The reason is the complicated video texture, not the motion activity. However, from frame 150 to 170, the values of SAD are raised sharply due to the sudden motion change instead of video texture. Consequently,

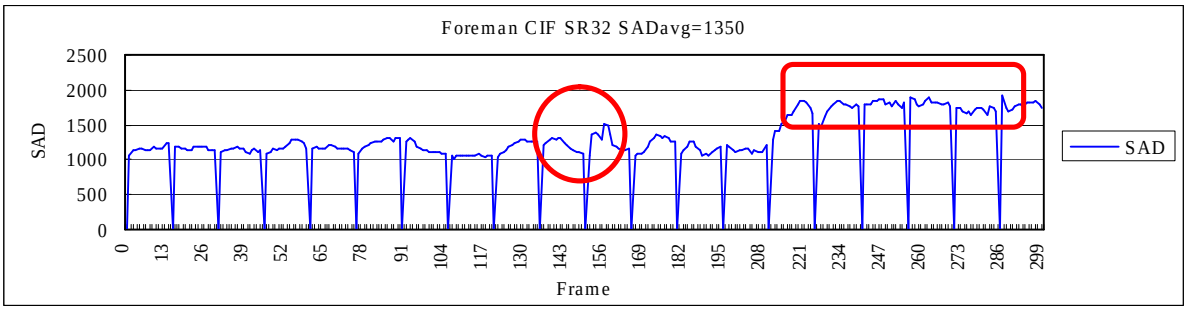

Figure 1. SAD of foreman CIF frame by frame

Step 1. the search range in frame level SR_FRAME_k is determined by

$$SR\_FRAME_k = \max[MV_{xt}, MV_{yt}] + 1$$

$$t \in \{\text{all blocks in (k-1)th frame}\}$$

Step 2. Adjust the search range in macroblock level.

$s \in$ { The left, above left, above, above right blocks of tth block }

If any of neighbor blocks is not available

$$MV\_MAX_t = \max[\max[MV_{xs}, MV_{ys}], SR\_FRAME_k]$$

Else

$$MV\_MAX_t = \max[MV_{xs}, MV_{ys}]$$

Step 3. Determine the final search range for tth block SR_BLOCK

If $(MV\_MAX_t \geq SR\_FRAME_k)$

$SR\_BLOCK_t = MV\_MAX_t + 1$

Else

$SR\_BLOCK_t = MV\_MAX_t + (SR\_FRAME_k - MV\_MAX_t)/$

If $(SR\_BLOCK_t \leq 1)$

$SR\_BLOCK_t = 1$

Else if $(SR\_BLOCK_t \geq$ max search range)

$SR\_BLOCK_t =$ max search range

Table 1. Simple Dynamic Search Range Algorithm

the approaches based on block matching error are usually unsuitable to evaluate the motion activity.

On the contrary, motion vector represents the motion activity more precisely [9]. For this reason, the proposed SDSR algorithm is based on motion vector information. Due to the wide variations of motion activity in video sequences and different motion activity in various areas within a single frame, we would like to adjust search range on both frame level and block level. The adjustments of SR in frame level and block level are based on temporal correlation and spatial correlation of motion field, respectively.

The proposed Simple Dynamic Search Range algorithm is described as table 1 shows. Because the prediction of MV may not be zero MV in motion estimation, the displacement of MV may be larger than the SR. Hence the SR in frame level may increase more than one unit between frames. The adjustment of SR in block level ensures that the SR is large enough to find the true MV.

3. Successive Elimination Algorithm with Integral Frame (SEAIF)

To eliminate the unnecessary matching in matching phase in motion estimation of H.264/ACV standard, we propose Successive Elimination Algorithm with Integral Frame (SEAIF) which integrates SEA and integral frame. SEA and integral frame are described in the following subsections.

3.1 Successive Elimination Algorithm (SEA)

In order to reduce the computation of SAD in the process of motion estimation, Successive Elimination Algorithm (SEA) [3] was proposed. The SEA is a lossless fast motion estimation algorithm based on mathematical inequality. The main idea of SEA can be shown in the following equation.

$$\begin{aligned} SAD(f_c, f_r(m,n)) &= \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} |f_c(i,j) - f_r(i+m, j+n)| \\ &\geq \left| \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f_c(i,j) - \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f_r(i+m, j+n) \right| \\ &\equiv |BS_c - BS_r(m,n)| \\ &\equiv sea(f_c, f_r(m,n)) \end{aligned} \quad (1)$$

, in which BS_c and BS_r are the block sums in the current block and candidate block, respectively. Because $SAD(f_c, f_r(m, n))$ is equal to or larger than $sea(f_c, f_r(m, n))$, if $sea(f_c, f_r(m, n))$ is larger than the current minimum SAD, $SAD(f_c, f_r(m, n))$ must be larger than the current minimum SAD. Therefore, computation of $SAD(f_c, f_r(m, n))$ can be skipped.

To compute *sea* value is easier than to compute SAD, because BS_c has to be calculated only once and $BS_r(m, n)$ can be derived from the previous value of $BS_r(m-1, n)$. Hence, SEA can reduce the computation of SAD efficiently.

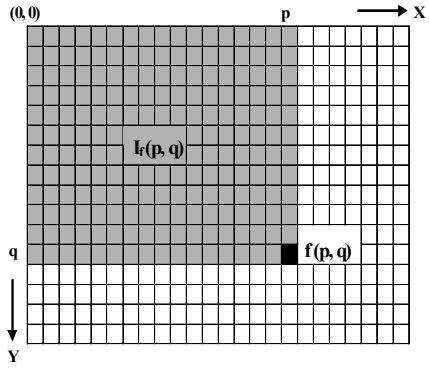

Figure 2 Integral frame

3.2 Integral Frame

Viola *et al.* [13] proposed the integral frame technique for sum of pixel values within any rectangular area in a frame. Given a video frame *f*, the value of its integral frame at pixel (*p, q*) is denoted as $I_f(p, q)$, as defined in the equation (2.5).

$$I_f(p,q) = \sum_{i=0}^{p}\sum_{j=0}^{q} f(i,j) \qquad (2)$$

The integral frame is shown in figure 2. The computational cost for an integral frame is described as follows. Let $R_f(p, q)$ be the cumulative row sum of pixel values in frame *f*. The definitions are:

$$R_f(p,q) = \sum_{i=0}^{p} f(i,q) \qquad (3)$$

$$R_f(-1,q) = 0 \qquad (4)$$

$$I_f(p,-1) = 0 \qquad (5)$$

$$R_f(p,q) = R_f(p-1,q) + f(p,q) \qquad (6)$$

$$I_f(p,q) = I_f(p,q-1) + R_f(p,q) \qquad (7)$$

By using equation (6) and (7) recursively, one can compute the integral frame I_f in one pass. For a frame with *W* x *H* pixels, 2*WH* additions are required to compute an integral frame. The sum of pixel values in any rectangular block in a frame can be computed by three arithmetic operations. For example, as illustrated in figure 3, the BS of block D can be computed by equation (8).

$$BS(D) = \sum_{i=r+1}^{p}\sum_{j=s+1}^{q} f(i,j)$$
$$= I_f(p,q) - I_f(r,q) - I_f(p,s) + I_f(r,s) \qquad (8)$$

In H.264/AVC standard, rate-distortion optimization (RDO) is recommended for mode selection. The modes include nine intra modes and seven inter mode. In inter-coding, 41 motion estimations is required for a 16x16 macroblock while the RDO is enabled. (One for 16x16, two for 16x8, two for 8x16, four for 8x8, eight for 8x4, eight for 4x8, and sixteen for 4x4) Therefore, the ME cost increases dramatically.

In order to reduce the intensive computation caused by RDO. In the H.264/AVC reference software JM 9.4[14], a Fast Full Pel Search algorithm is implemented by reusing SAD values of the smallest 4x4 block. Before a new macroblock is motion estimated, it computes the SAD values for all 4x4 block at all search points within the search window. After that, it merges the SAD values to get the SAD values of larger blocks. In this way, computation of SAD for a macroblock with all block size enabled is about equal to the computation of SAD with only a 16x16 block.

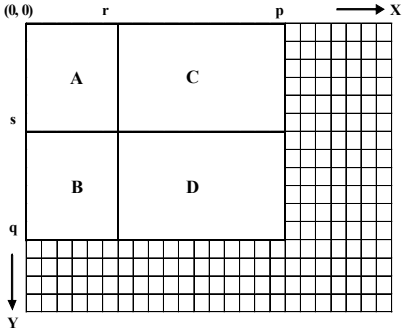

Figure 3 Computation of block sum

We take the concept of reusing SAD and integrate it into our proposed SEAIF. The main idea of the SEAIF for H.264/AVC is to reuse *sea* values and SAD values.

3.3 Reusing of *sea* value

For each search point, calculate the *sea* values of sixteen 4x4 blocks of the current macroblock by using integral frame technique proposed by Viola et al. [13]. These *sea* values of 4x4 blocks are the basis for *sea* values of larger blocks. Then the *sea* values of larger blocks are derived from these *sea* values of 4x4 blocks, described as follows.

- For 8x4 or 4x8 block, sum up *sea* values of two 4x4 blocks.
- For 8x8 block, sum up *sea* values of two 8x4 blocks.
- For 16x8 or 8x16 block, sum up *sea* values of two 8x8 blocks.
- For 16x16 block, sum up *sea* values of two 16x8 blocks.

In this way, we can get all *sea* values of all blocks. These *sea* values of larger blocks are always equal to or larger than the *sea* values computed directly from BS of corresponding blocks. Therefore, the *sea* values of larger blocks derived from 4x4 block *sea* values are lower bound of SAD and the more computations of SAD can be skipped.

3.4 Reusing of SAD value

In SEAIF, if the *sea* value is less than the current minimum SAD value, complete calculation of SAD will be preformed. In H.264/AVC, overlapped blocks are used in motion estimation. In order to reduce the computations of SAD, we take the 4x4 block SAD values as the basis of the larger block SAD values. In

this way, there is no redundant computation of SAD. The following is the approach

3.5 Analysis of complexity

The reason of adopting SEA is to reduce the computational cost in block matching measurement. The overhead of SEA should be considered and analyzed. The overheads of SEA are mainly the computations of block sum. In SEA [3], Salari et al. proposed a fast algorithm to compute the block sums. We compare three approaches and present the analysis of overhead as follows.

Let W denote image width, H image height, M block width, and N block height. Operations required for block sums of all M x N blocks in a reference frame are:

- **Straightforward approach:**
  Number of block sum in a frame: $(W-M+1)(H-N+1)$
  Operations required for a block sum: $MN-1$
  Total cost: $(MN-1)(W-M+1)(H-N+1)$
  Approximate cost: $MNWH$
- **SEA approach in [3]:**
  Total cost: $4WH-(H-N)(M+3)-3W(N+1)$
  Approximate cost: $4WH$
- **Integral frame approach:**
  Operations required for an integral frame: $2WH$
  Operations required for all block sum: $\approx 2(W-M+1)(H-N+1)$
  Total cost: $2WH+2(W-M+1)(H-N+1)$
  Approximate cost: $4WH$

Although integral frame approach and the SEA approach in [3] have approximately the same complexity, there is an advantage in integral frame approach. Integral frame approach is flexible to get block sum of any rectangle block. For example, if we want to use the multilevel SEA for each block size in H.264/AVC, the implementation will be easier with integral frame approach. (Note that our approach uses the tighter lower bound in SEA, not multilevel SEA.) Computing *msea* value of 16x16 block with level L=0 only needs 5 operations (5 = 3 for get BS + 1 subtraction + 1 absolute). Nevertheless, merging 16 4x4 *sea* values to get the *sea* value of 16x16 block with level L=0 needs 15 addition operations while the *sea* value is tighter lower bound. Trade-off is between the tighter lower bound and computational complexity.

4. Early Termination Algorithm (ETA)

The early termination scheme defines a criterion to early terminate the search processing to help existing the motion search algorithms by further reducing the amount of computation. In [10], Siou-Shen Lin *et al.* introduce the variance of motion vectors. They show the probability is about 79% in average when the variance of the current block and neighbor blocks is smaller than 3. They consider that it is high probability that the current block and the neighbor blocks might belong to the same object when the variance of the motion vectors in the neighbor blocks is small.

We exploit and modify the variance of motion vectors proposed in [10] to classify the motion activity of current block and neighbor blocks into simple motion and complex motion. The variance of motion vectors is defined in equation (10).

$$MV_{mean} = (MV_a + MV_b + MV_c + MV_d)/4 \quad (9)$$

$$MV_{\text{var}} = |MV_a - MV_{mean}| + |MV_b - MV_{mean}| + |MV_c - MV_{mean}| + |MV_d - MV_{mean}| \quad (10)$$

If any of neighbor blocks is not available, *MVvar* is set to a large value (999999). For accuracy, we compare the *MVvar* with 5 instead of 3 to classify motion activity, shown in equation (11).

$$\text{If}(MV_{\text{var}} \le 5)\ Mactivity = simple\_motion \ \text{Else}\ Mactivity = complex\_motion \quad (11)$$

If motion activity is simple motion, we consider the current block and neighbor blocks are in the same object for simple. On the contrary, the current block and neighbor blocks are considered not in the same block. The SAD values of blocks within the same object should be similar and the SAD values of blocks not in the same object should be different largely. Based on the concept, the lower bound for the condition of termination is determined in equation (12).

$$\text{If}(Mactivity == simple\_motion)\ SAD\_threshold = SAD\_prediction \ \text{Else}\ SAD\_threshold = SAD\_prediction - SAD\_std\_dev \quad (12)$$

The *SAD_prediction* and *SAD_std_dev* represent the prediction of SAD of current block and the standard deviation of SAD of all blocks in the previous frame, respectively. The definitions are defined in equation (13) and (15):

$$SAD\_prediction = (SAD_a + SAD_b + SAD_c + SAD_d)/4 \quad (13)$$

$$SAD\_mean = \frac{1}{Number\_MB} \sum_{t=0}^{Number\_MB-1} SAD_t \quad (14)$$

$$SAD\_std\_dev = \left(\frac{1}{M-1} \sum_{t=0}^{M-1} (SAD_t - SAD\_mean)^2 \right)^{1/2} \quad (15)$$

The SAD_t is the SAD value of tth block in a frame. *Number_MB* is the total number of MB in a frame. If there is no any neighbor block near the current block, *SAD_prediction* is set to a small value (-999999). Note that the *SAD_prediction* and *SAD_std_dev* are calculated for 16x16 macroblock. In H.264/AVC standard, there are seven block sizes used in motion estimation. We determine the *SAD_prediction* and *SAD_std_dev* for other block size according to the area occupied by the block.

Finally, the condition of termination is tested when a new up-to-date best-matched block is found. If the SAD

value of the up-to-date block is equal to or smaller than *SAD_threshold*, the motion estimation is terminated.

5. EXPERIMENTAL RESULTS AND DISSCUSIONS

In this chapter, we present the experimental results of the proposed approaches. We modify the H.264/AVC reference software JM 9.4 and add the proposed algorithms in it. In the experiments, we observe the number of search points for each block to measure the performance of the proposed algorithms. We also measure the coding efficiency. In order to measure the coding efficiency, we compare the bitrates of encoded sequences with the same quantization parameter and disabling rate control. Besides, we exploit the SAD value as a criterion to measure whether the determined search range is large enough. Finally, we compare the total encoding time to measure the improvement in practical situation.

The descriptions of test video sequences are listed in table 2. Except specifically described parameters, the following parameters are applied to all experiments. Note that the maximum search range is set to 24.

- Platform: H.264/AVC reference software JM 9.4 [14]
- Machine: Athlon XP 1700+ with 512 MB memory
- Profile: baseline
- Level: 3.0
- Block match algorithm (BMA): Full Search
- Group of picture (GOP): 15
- Quantization parameter (QP): 36
- Frame rate (FPS): 30
- Max search range: 24
- Frame structure: IPPP
- Number of reference frame: 1
- Hadamard transform: enable
- All block size (16x16, 16x8, 8x16, 8x8, 8x4, 4x8, and 4x4): enable
- Rate-distortion optimized (RDO): enable
- Fast ME (UMHexagonS) [15]: disable
- Fast mode selection [16]: disable
- Rate control (RC): disable

In the experiments, we compare our proposed algorithms with Fast Full Pel Search which is implemented by reusing SAD values of the smallest 4x4 block. Before a new macroblock is motion estimated, it computes the SAD values for all 4x4 block at all search points within the search window. After that, it merges the SAD values to get the SAD values of larger blocks. In this way, computation of SAD for a macroblock with all block size enabled is about equal to the computation of SAD with only a 16x16 block.

Note that the performances of the Fast Full Pel Search and a conventional Full Search are the same but the Fast Full Pel Search is faster than a conventional Full Search in H.264/AVC. In the following experiments, we denote the Fast Full Pel Search as FFS.

ID	Name	Resolution	# of Frames	Motion activity
A	Foreman	QCIF	150	Medium
B	Mobile	QCIF	150	Slow
C	Coastguard	QCIF	150	Medium
D	Foreman	CIF	150	Medium
E	Tempete	CIF	150	Slow, Zooming
F	Flower	CIF	90	Slow
G	Stefan	SIF	150	High
H	Football	CIF	90	Very High
I	Table tennis	SIF	90	Medium, Scene change, Zooming

Table 2 Descriptions of test video sequences

In the table 3, 4 and 5, the number of search points can be reduced more than 90% in most of the sequences. Especially, for the slow and median motion, the reduced rates of search points are about 99%. For high motion, the reduced rates of search points should be lower. The reduced rate of search points is 73.8% for football sequence. In average, the increment of bit rate in CAFME is very small, about 0.26%. The total encoding time is reduced about 41.9%, and the number of SP is reduced about 93.1%.

Sequence ID	Number of Search Points		Improvement
	FFS	CAFME	
A	2401	37	− 98.5%
B	2401	12	− 99.5%
C	2401	29	− 98.8%
D	2401	100	− 95.8%
E	2401	69	− 97.1%
F	2401	199	− 91.7%
G	2401	184	− 92.3%
H	2401	628	− 73.8%
I	2401	224	− 90.7%
Average			**− 93.1%**

Table 3 Search Points of FFS and CAFME

Sequence ID	Bitrates (Kbps)		Improvement
	FFS	CAFME	
A	69.203	69.118	− 0.12%
B	173.016	173.285	+ 0.16%
C	76.134	75.862	− 0.36%
D	188.773	188.784	+ 0.005%
E	425.392	425.955	+ 0.13%
F	669.312	670.211	+ 0.13%
G	505.450	504.782	− 0.13%
H	413.301	419.357	+ 1.5%
I	256.259	258.939	+ 1.04%
Average			**+ 0.26%**

Table 4 Bitrates of FFS and CAFME

Sequence Name	Total Encoding Time (Second)		Improvement
	FFS	CAFME	
A	156	69	− 56%
B	151	77	− 49%
C	151	68	− 55%
D	602	314	− 48%
E	583	318	− 45%
F	374	224	− 40%
G	508	324	− 36%
H	363	325	− 10%
I	298	184	− 38%

Table 5 Total Encoding Time of FFS and CAFME

6. Conclusion

The motion estimation plays an important role in the video compression. However, motion estimation module is usually the most computational intensive part in a typical video encoder. Hence, the efficient motion estimation algorithm is needed. We proposed a fast algorithm called Content-Aware Fast Motion Estimation Algorithm (CAFME). CAFME consists of the Simple Dynamic Search Range (SDSR), Successive Elimination Algorithm with Integral Frame (SEAIF), and Early Termination Algorithm (ETA). The SDSR adjusts the search range for every block adaptively. The SEAIF reduces the number of computation of SAD without loss. The ETA terminates the search process early when finding a good candidate block.

CAFME outperforms the FFS in the experiments and the overall encoding time is reduced about 41.9%.

REFERENCES

[1] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing" *Proc. Nat. Telecommun. Conf.*, pp. G5.3.1–5.3.5, New Orleans, LA, Nov. 29–Dec. 3 1981.

[2] S. Zhu and K.-K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", *IEEE Trans. on Image Processing*, Volume 9, Issue 2, pp. 287–290, Feb. 2000.

[3] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation" *IEEE Trans. on Image Processing*, Volume 4, Issue 1, pp. 105–107, Jan. 1995.

[4] X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation" *IEEE Trans. on Image Processing*, Volume 9, Issue 3, pp. 501–504, Mar. 2000.

[5] L.-W. Lee, J.-F. Wang, J.-Y. Lee, and J.-D. Shie, "Dynamic Search-Window Adjustment and Interlaced Search for Block-Matching Algorithm" *IEEE Trans. on Circuits and Systems for Video Technology*, Volume 3, Issue 1, pp. 85–87, Feb. 1993.

[6] J. Feng, K.-T. Lo, H. Mehrpour, and A.E. Karbowiak, "Adaptive Block Matching Motion Estimation Algorithm for Video Coding" *IEE Electronics Letters*, Volume 31, Issue 18, pp. 1542–1543, Aug. 31 1995.

[7] J. Minocha and N.-R. Shanbhag, "A Low Power Data-Adaptive Motion Estimation Algorithm" *IEEE 3rd Workshop on Multimedia Signal Processing*, pp. 685–690, Sep. 13-15 1999.

[8] S. Saponara and L. Fanucci, "Data-Adaptive Motion Estimation Algorithm and VLSI Architecture Design for Low-Power Video Systems" *IEE Proc. on Computers and Digital Techniques*, Volume 151, Issue 1, pp. 51–59, Jan. 15 2004.

[9] P.-I. Hosur, "Motion Adaptive Search for Fast Motion Estimation" *IEEE Trans. on Consumer Electronics*, Volume 49, Issue 4, pp. 1330–1340, Nov. 2003.

[10] S.-S. Lin, P.-C. Tseng, C.-P. Lin, and L.-G. Chen, "Multi-Mode Content-Aware Motion Estimation Algorithm for Power-Aware Video Coding Systems" *IEEE Workshop on Signal Processing Systems*, pp. 239–244, 13-15 Oct. 2004.

[11] V.-A. Nguyen and Y.-P. Tan, "Fast Block-Based Motion Estimation Using Integral Frames", *IEEE Signal Processing Letters*, Volume 11, Issue 9, pp. 744–747, Sep. 2004.

[12] K.-P. Lim, G. Sullivan, and T. Wiegand, "Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods" ITU-T, Doc. #JVT-N046, Jan. 2005.

[13] P. Viola and M.-J. Jones, "Robust Real-Time Object Detection" Cambridge Res. Lab., Tech. Rep. CRL 2001/01, Feb. 2001.

[14] H.264/AVC reference software, http://ftp3.itu.ch/av-arch/jvt-site/reference_software/ and http://iphome.hhi.de/suehring/tml/

[15] Z. Chen, P. Zhou, Y. He, and Y. Chen, "Fast Integer Pel and Fractional Pel Motion Estimation for JVT" ITU-T, Doc. #JVT-F017, Dec. 2002.

[16] B. Jeon and J. Lee, "Fast Mode Decision for H.264" ITU-T, Doc. #JVT-J033, Dec. 2003.