SOFTWARE PIPELINES DESIGN FOR VARIABLE BLOCK-SIZE MOTION ESTIMATION WITH LARGE SEARCH RANGE

Zhigang Yang[1], Wen Gao[1,2], Yan Liu[1], and Debin Zhao[1]

[1]Department of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[2]Institute of Computing Technology, Chinese Academy of Science, Beijing 100080, China
{zgyang, wgao, liuyan, dbzhao}@jdl.ac.cn

ABSTRACT

This paper presents some techniques for efficient motion estimation (ME) implementation on fixed-point digital signal processor (DSP) for high resolution video coding. First, challenges in large search-range ME are discussed. Then, based on the statistics analysis of the best matched block, correlations in variable block-size ME are removed on algorithm level while keeping almost the same performance as the original fast search strategy. Based on this modification and data reuse technique, this paper proposes an "index search" method for DSP implementation, which is a good solution to balance both high coding efficiency and high coding speed on the DSP platform. In "index search", ME is divided into three steps and highly parallel pipelines are designed for each step. The fully use of pipelines brings about 85% time reduction during ME. "Index search" can also be easily extended to achieve different search range and search patterns.

Index Terms— Pipelines, digital signal processors, motion estimation, video coding

1. INTRODUCTION

Motion estimation (ME) is a key to block-based hybrid video coding for reducing the high temporal redundancy between successive frames. Most of the video coding efficiency is derived from ME, while it also contributes the heaviest computational burden for the encoder. Especially in high resolution video coding, large search range (SR) and variable block size (VBS) adopted to achieve high coding efficiency, become the most serious bottlenecks in real-time applications. Thus, fast ME has always been an important and attractive topic in video compression.

Generally speaking, there are two main research interests for ME: one is fast search strategy in software and the other is efficient hardware implementation. Fast search strategy in software is focus on reducing computation through different search steps and search patterns, e.g. 3SS [1], diamond search (DS) [2], UMHexagonS search [3] etc. Further more, the early termination technique [4] which can speed up search process was accepted by the latest standards such as H.264 [5] and AVS [6]. For hardware implementation, the architecture is paid more attention on, e.g. by means of application specified integrated circuits (ASIC) or field programmable gate arrays (FPGA). Since the hardware design can be highly optimized, fast full search (FFS) [7] is usually adopted.

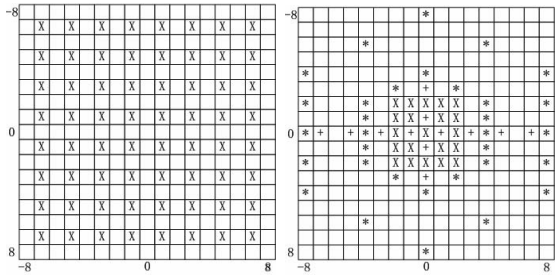
As for digital signal processor (DSP) applications in video coding, it is a combination of software and hardware. In the aspect of software, high-level program language can be used, so it is easy for DSP to add or modify functions. In the other aspect of hardware, DSP has its own very long instruction word (VLIW) CPU architecture, multi-level memory organization and assembly language. Those characteristics should be considered in optimization work. Recently, special kinds of DSP, like TI TMS320DM642, Philips PNX1700, etc, address the needs of video processing, making DSP widely used in video applications.

The purpose of this paper is to accelerate ME by weak processing correlation and efficient software pipelines. The rest of this paper is organized as follows. In Section 2, the challenges in large search-range ME (LSR-ME) are discussed. In Section 3, the block correlations in variable block-size ME (VBS-ME) are removed from the DSP-oriented viewpoint. In Section 4, software pipelines for the proposed "index search" algorithm are designed in detail, and simulated results are demonstrated to show its effectiveness. Finally Section 5 concludes the paper.

2. CHALLENGES IN LSR-ME

For high resolution video, the search range should be large enough to keep high coding efficiency. The traditional search strategies like 3SS, DS, etc, can do well in a small range ±8. But in LSR-ME, they are likely to drop into a local minimum in the early stages of search process which results in low coding efficiency. In order to avoid local minimum points, global search is usually performed with many sampled points over the whole search window. Then local search is performed to refine the motion vector.

In early time, Grid-2 [8], shown in Fig. 1(a), was used to perform global search. There is a search point every two grids vertically and horizontally. The total points are nearly 1/4 of full search. The computation is still heavy. Experiments show that the picture's quality will drop rapidly as the grid size increases. Later on, UMHexagonS [3], shown in Fig. 1(b), was presented and has been adopted in H.264 and AVS reference software. Such grids have been proved to be efficient in LSR-ME. According to Table I, the global search occupies more than 93% computation, so accelerating global search is important for fast ME.



(a) Grid-2 **(b) UMHexagonS**

Fig. 1. Global search points in Grid-2 and UMHexagonS (when SR=8)

* Supported by the National Natural Science Foundation of China under Grant No. 60672088.

TABLE I
SEARCH POINTS STATISTICS

SR	Grid-2		Grid-4		UMHexagonS	
	global	local	global	local	global	local
32	1024	8	256	16	193	10 - 15
	99.2%		94.1%		93.9%*	
48	2304	8	576	16	281	10 - 15
	99.7%		97.3%		95.7%*	
64	4096	8	1024	16	352	10 - 15
	99.8%		98.5%		96.6%*	

* Assumption: local search point number is 12.5.

TABLE II
PROBABILITY OF THE BEST MATCHED BLOCK IN THE SEARCH AREA OF BLOCK 16X16

Frame \ Mode	16x8-0	16x8-1	8x16-0	8x16-1
P frame	99.74%	97.53%	98.54%	98.30%
B frame	99.91%	99.65%	99.76%	99.73%
Frame \ Mode	8x8-0	8x8-1	8x8-2	8x8-3
P frame	98.98%	98.68%	98.41%	98.27%
B frame	99.88%	99.85%	99.78%	99.71%

Note: every value is the minimum of the seven sequences, S1-City, S2-Crew, S3-Harbor, S4-Night, S5-Raven, S6-Sailormen, and S7-Sheriff.

The early termination strategy in [4] was presented to reduce the computation of UMHexagonS. Our experiments showed that this strategy can reduce average 52% global search points for the tested 720p sequences. Unfortunately, there are lots of floating-point operations, which can be a disaster for fixed-point DSP. For example, the executing time for 8x8 and 16x16 SAD calculation are respectively 31 and 67 cycles according to DSP image/video processing library, but even if a fixed-point division needs 25 cycles [9], say noting of floating-point operations. Therefore, it needs another way to implement fast ME on fixed-point DSP.

Data reuse strategy can be adopted instead of early termination technique to reduce the computation of global search, but the block correlations introduced by VBS limit the usage of this strategy. So in the next section, we are going to analyze VBS-ME and find out solutions to remove the block correlations.

3. CORRELATIONS IN VBS-ME

The Lagrangian cost function

$$J(m) = SAD(o,r,MV) + \lambda_{motion} \cdot R(MV\text{-}PMV) \quad (1)$$

takes motion vector (MV) cost into consideration. The prediction of MV (PMV) of each block is generally gathered by the left, top, and top-right neighboring blocks, and PMV also points to the search center of each block. Therefore, the current block can be processed only after the prediction modes of the neighboring blocks have been determined. These correlations cause inevitable sequential processing flow and limit wide uses of background transfer [10], both of which result in low DSP efficiency.

Assume that *pmv0* is the PMV of the block 16x16, and VBS is from 16x16 to 8x8. The statistics in Table II shows almost every best matched block (no matter the block is finally selected or not) can be found in the search area of block 16x16. If all the blocks use *pmv0* as their own search centers, they will still find out the best point as usual. What's more, their global search paths are same, so SAD of big block can be combined with small blocks', which is important for data reuse in DSP implementation. To accord with the cost function (1), PMV of each block is temporarily replaced by *pmv0* only during the period of ME. This modified DSP-oriented algorithm can be adopted instead of the early termination technique. Table III lists the detailed performance of the improved algorithm compared with the current early termination technique. The average PSNR gains about 0.005dB and the average increment of bit rate is 0.866%.

TABLE III
PERFORMANCE COMPARISON BETWEEN THE IMPROVED ME ALGORITHM AND THE EARLY TERMINATION TECHNIQUE

(a) PSNR GAIN (dB)

QP	20	24	28	32	36	40	44
S1	0.000	-0.001	-0.001	-0.003	-0.004	-0.004	-0.006
S2	0.021	0.024	0.023	0.016	0.014	0.014	-0.004
S3	0.002	0.002	0.001	0.002	-0.001	0.001	0.000
S4	0.012	0.011	0.009	0.010	0.007	0.011	0.005
S5	0.009	0.018	0.035	0.042	0.019	-0.028	-0.036
S6	0.006	0.005	0.004	0.003	-0.002	-0.003	-0.008
S7	0.005	0.004	0.002	-0.001	0.002	0.000	0.000

(b) BIT RATE INCREMENT (%)

QP	20	24	28	32	36	40	44
S1	0.005	0.038	0.128	0.222	0.494	0.503	0.726
S2	0.282	0.559	0.909	1.236	1.314	1.784	2.341
S3	0.029	0.033	0.099	0.178	0.351	0.566	0.734
S4	0.315	0.435	0.632	0.901	1.249	1.928	2.535
S5	0.219	0.290	0.330	0.558	1.976	4.558	3.703
S6	0.251	0.348	0.488	0.653	1.189	2.272	3.043
S7	0.070	0.097	0.111	0.264	0.292	0.497	0.702

Note: Sequences S1 to S7 are same with TABLE II.

4. SOFTWARE PIPELINES DESIGN FOR ME

4.1. Index Search

According to Section 3, all the nine blocks adopt the same PMV during global search, so the MV cost $\lambda_{motion} \cdot R(MV\text{-}PMV)$ needs only to be calculated once for one position, and SADs of all 8x8 blocks need to be calculated to setup other bigger blocks. Thus, the computation is already reduced greatly. Furthermore, DSP can still save more time due to its software pipeline if carefully designed.

Before designing pipelines, an array "index" is defined first. The array "index" previously stores the horizontal and vertical offsets, since all global search paths are fixed to the start center. These offsets correspond to the points in Fig. 1(b). Thus DSP can follow the stored order in "index" to perform ME, and we call this method "index search". Then operations with the same function are processed together, and it is helpful to design efficient pipelines. Three steps of global search are listed as follows:

Step1: For all the points in "index", calculate $\lambda_{motion} \cdot R(MV\text{-}PMV)$ and store the results to array "mvcost".

Step2: For all the points in "index", calculate SAD of the 8x8 block and store the results to array "sad". Repeat this step four times to get all 8x8 blocks' SAD.

Step3: Combine four "sad" and "mvcost" to create all nine blocks' costs. Compare and store the minimum cost and the corresponding index number for each block, in order to be refined in local search.

Our design is based on TI TMS320C64x DSP. The CPU of such DSP has two similar data paths (A/B), and each data path mainly consists of 32 register files and 4 functional units. As for this architecture, high parallelism is achieved by software pipeline. There are total eight functional units in CPU, so in a single clock cycle, CPU can execute a maximum of eight instructions in parallel, reaching its peak performance. More detailed information of this kind of DSP can be found in [11].

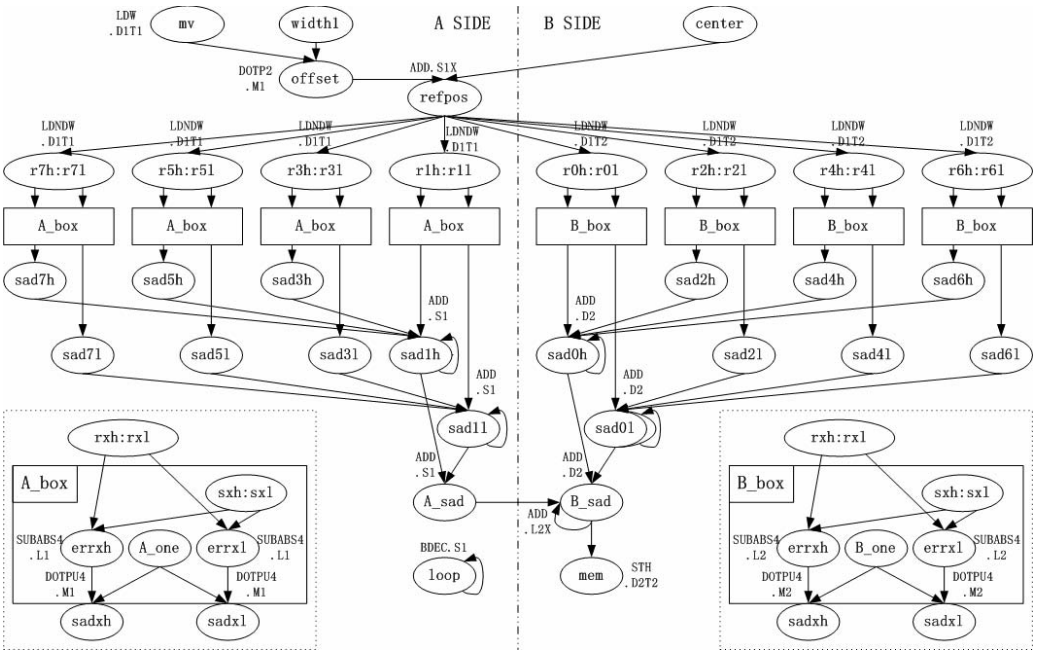


Fig. 2. Dependency graph of Step2

Note: "A_box" and "B_box" are sub-graphs only to make the whole graph clear, NOT functions.

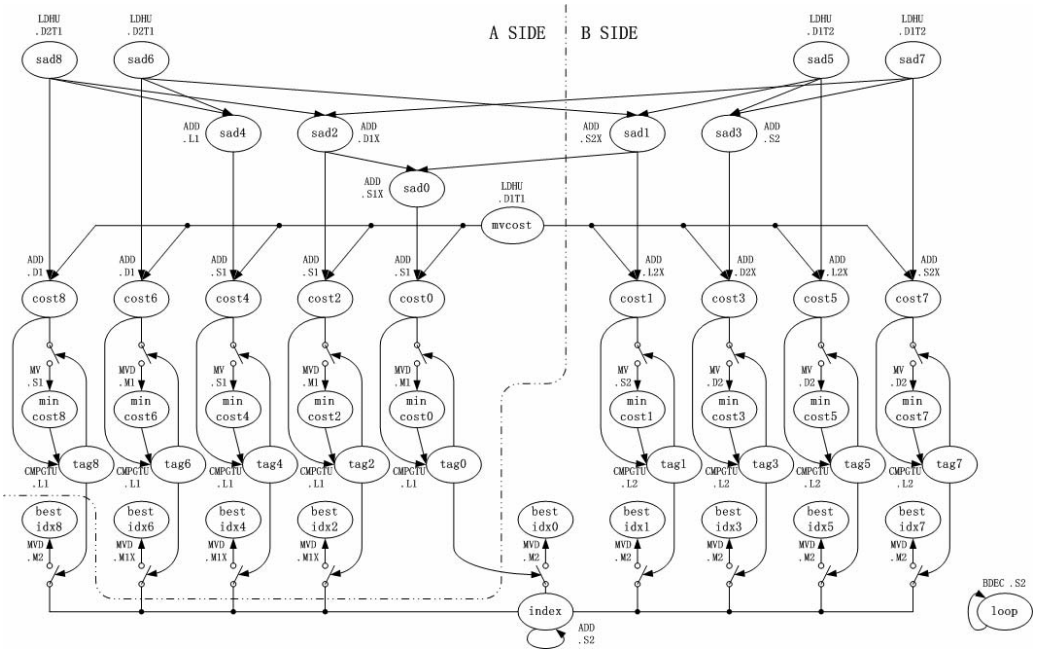


Fig. 3. Dependency graph of Step3

Dependency graph is a useful tool for efficient DSP software pipeline design. Each node in the dependency graph denotes an operand, and edges connecting the nodes denote instructions. The dependency graph can also indicate how to allocate data paths. The basic rule of drawing the graph is that all the resources on each side of CPU data path should be balanced. Since Step1 is simple, we only elaborate on Step2 and Step3.

In Fig. 2, pixels of source 8x8 block have been preloaded into eight register pairs (sxh:sxl). The DOTP2 instruction calculates the search position's offset in the reference frame and the LDNDW instruction loads eights pixels from reference frame into another eight register pairs (rxh:rxl). Then, the SUBABS4 and DOTPU4 instructions are used together to do 1x4 SAD, so 8x8 SAD is completely unrolled by 16 pairs of SUBABS4 and DOTPU4 operations, which compose the main loop kernel. At last, the STH instruction stores the summed up result to memory. In step2, functional unit, .M1, .D1, .S1 and .L2 are all used nine times, which determine the minimum loop kernel is 9 cycles.

The first part of Fig. 3 is a kind of data reuse. In the rest part of Fig. 3, the CMPGTU instruction compares the current cost with

the recorded minimum cost. If current cost is better, minimum cost is replaced by current cost and the corresponding position (index) is stored as well. In the period of cost replacement, a lot of "data move" instructions are used. MV is a single-cycle instruction and moves a value from one register to another through .L, .S or .D unit. MVD is a four-cycle instruction and moves a value from one register to another through .M unit. Although MVD instruction has more operation cycles than MV instruction, .L, .S and .D units are mostly occupied by other instructions like ADD and LDHU. In order to balance all kinds of functional units, MVD instruction is adopted to full fill the pipeline. As shown in Fig. 3, six MVD instructions are used in either side A or B. This code thus executes a maximum of eight instructions per cycle, and its loop kernel is six cycles long.

4.2. Simulated Results

Table IV shows benchmarks of each step, then the total global search cycles for index search is (3n+24)+4×(9n+34)+(6n+35) =45n+195. In order to show the effectiveness of index search, we also optimize the original ME algorithm with "DSP image/video processing library" [9] which has been highly optimized by hand-writing assembly language, and then compare them. The cycles of original algorithm are calculated as follows: the cycles of 16x16, 16x8, 8x16, 8x8 SAD are 67, 43, 43, 31 respectively according to DSP library, and 10 more cycles should be added for each block because of $\lambda_{motion} \cdot R(MV\text{-}PMV)$ calculation and costs comparison. Then the total global search cycles for original algorithm is (77+2×53+2×53+4×41)×n=453n. When search range is 32 (n=193), index search can save 89.8% of global search time. In fact the proportion is still higher, because the cycles for loop control and offset calculation are not considered. This great time reduction is achieved based on high-density pipelines, not for reasons of decreasing search points.

Having taken local search into consideration, Table V shows the average cycle per macroblock (MB) on DSP platform. Index search can reduce more than 84.4% cycles and is proved to be efficient for LSR-ME and VBS-ME.

TABLE IV
BENCHMARKS OF EACH STEP IN INDEX SEARCH

Step	Cycle	Code size
1	3n+24	172 bytes
2	9n+34	588 bytes
3	6n+35	753 bytes

Note: n is the global search point number.

TABLE V
COMPARISON OF AVERAGE CYCLE PER MB

SR	Original (use DSP Lib [9])	Index search	Cycle reduction*
32	93092	14543	84.4%
48	132956	18503	86.1%
64	165119	21698	86.9%

* (1-index search cycle/original cycle)×100%

4.3. Index Search Extension

Variable search range can be easily achieved by controlling the index search number. And different search patterns can also be achieved by modifying index array. For example, if the search points in Fig. 1 are stored into index array, the algorithm will perform Grid-2 search. Or if all points stored, it will change to be fast full search. Irregular search paths are also fully supported by index search. In a word, once all search points have been mapped into one dimensional array, DSP can setup highly parallel pipelines to perform ME.

5. CONCLUSIONS

From algorithm level and pipeline level, this paper elaborates on efficient designs for motion estimation in high resolution video coding on fixed-point DSP platform. Through the analysis of LSR-ME and VBS-ME in high resolution video coding, this paper presents a DSP-oriented search algorithm. The proposed method avoids block correlation and adopts data reuse technique. The experimental results show that the improved algorithm has almost the same performance comparing with the early termination technique in the reference software when coding 720p sequences.

Based on the improved algorithm, this paper also proposes the "index search" method for DSP implementation. This method does not decrease the search points but can still save much time, because three highly parallel pipelines can be built up with the help from "index search". The fully use of pipelines brings about 85% time reduction during motion estimation. Thus this paper provides a good solution to achieve both high coding efficiency and high coding speed on the DSP platform.

6. REFERENCES

[1] T. Koga, K. Iinuma, A. Hirano, Y. Lijima, etc, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecommunications Conf. 81*, pp.G5.3.1-G5.3.5, Nov. 1981.

[2] Shan Zhu and Kai-Kuang Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Trans. Image Processing*, Vol. 9, No. 2, pp. 287-290, Feb. 2000.

[3] Zhibo Chen, JianFeng Xu, Peng Zhou, and Yun He, "Hybrid unsymmetrical-cross multi-hexagon-grid search strategy for integer pel motion estimation in H.264," in *Proc. PCS*, May 2003.

[4] J. Xu, Z. Chen, and Y. He, "Efficient fast ME predictions and early-termination strategy based on H.264 statistical characters," in *Proc. ICICS-PCM*, Vol. 1, pp. 218-222, Dec. 2003.

[5] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC)," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050, 2003.

[6] "Final draft of information technology – advanced coding of audio and video – part 2: video," in AVS workgroup Doc. N1214, Shanghai, China, Sep. 2005.

[7] Tae Gyoung Ahn, etc, "Fast Full-Search Motion Estimation Based on Multilevel Successive Elimination Algorithm," *IEEE Trans. CSVT*, Vol. 14, No. 11, pp. 1265-1269, Nov. 2004.

[8] Fang-Hsuan Cheng and San-Nan Sun, "New fast and efficient two-step search algorithm for block motion estimation," *IEEE Trans. CSVT*, Vol. 9, No. 7, pp. 977-983, Oct 1999.

[9] TMS320C64x DSP Image/Video Processing Library Programmer's Reference,SPRU023A, Apr.2002, http://www.ti.com

[10] Zhigang Yang, Wen Gao, and Yan Liu, "Performance-Complexity Analysis of High Resolution Video Encoder and Its Memory organization for DSP Implementation," in *Proc.ICME06*, pp. 1261-1264, July 2006.

[11] TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide, SPRU732A, Jun. 2005, http://www.ti.com