Abstract—This paper presents a computationally-scalable motion estimation algorithm in which the number of operations required to code a video frame can be dynamically controlled. Unlike fast lossy search algorithms, the proposed algorithm can adaptively adjust its search strategy according to the imposed computational constraint and this algorithm can be used by consumer devices to realize real-time computational budget control for video compression.

I. INTRODUCTION

H.264/AVC[1] is the latest video coding standard, which is able to achieve higher compression efficiency compared to MPEG-2 at the expense of higher computational requirement. Hence various fast search algorithms have been proposed in literature to speed up the H.264/AVC motion estimation (ME)[1-3].

In[4], a fast lossless full search algorithm is proposed in which it adaptively adopts successive elimination algorithm (SEA)[5] and partial distortion search (PDS)[6] for H.264/AVC ME. For fast lossy search approaches, in [7], a fast block mode determination algorithm is proposed to determine the optimal block mode to code each target macro-block (MB). In [8], another fast lossy algorithm is proposed in which it adaptively adjusts the search window size for checking reference frames other than the temporal nearest one.

For the practical realization of a video encoder for consumer products, the computational power may be limited, due to the demand of various functions of a processor. Hence in this paper, a computationally-scalable ME algorithm is proposed. Unlike conventional fast algorithms, our algorithm is able to meet the change in computational constraints using a tailor-made ME structure. Depending on the allocated computational budget, the proposed algorithm can dynamically adjust its search strategy, either adopting a fast full search approach or a combination of different fast lossy search techniques, to produce the optimal predicted frame. Such computational scalability is useful for many consumer devices; e.g. a video scene may have to be encoded by a mobile device for storage, but some video identification/tracking functions may have to be carried out concurrently. Hence there is only limited processing power for the video encoding process. Experimental results show that if we impose a computational constraint to be half of the computation required by a fast full search[4], the proposed algorithm can achieve this new computational requirement within the time for 30 frames while the PSNR and bit rate performances are just slightly degraded.

II. PROPOSED ALGORITHM

A. Motion Estimation Operating States

In order to meet the allocated computational budget for video compression, the proposed algorithm provides the following ME states. The higher is the ME state, less computation will be required for the ME process.

1) ME state 1: In this state, a fast full search algorithm as described in [4] is used. Let us denote this fast full search algorithm as H.264 lossless SEA+PDS algorithm.

2) ME state 2: In this state, a lossy version of [4] is designed in which the pre-computed SEA boundary values are used “as is” the sum of absolute difference (SAD) in finding the optimal candidate. Let us denote this algorithm as H.264 lossy SEA+PDS algorithm.

3) ME state 3: In this state, H.264 lossy SEA+PDS is used together with the early termination in finding the optimal reference frame to code a target MB. That is, once the optimal candidate from a reference frame is found to have a SAD value smaller than a pre-defined threshold, this algorithm will not check the rest of the reference frames. If a larger threshold is used, there will be a higher chance to terminate the search earlier. Hence 13 sub-states are further defined with increasing threshold values used for the reference frame early termination.

4) ME state 4: In this state, the H.264 lossy SEA+PDS is used together with the early termination in finding the optimal reference frame as well as the optimal block mode to code a target MB. For block mode early termination, once the SAD value achieved in coding a target MB by a particular block mode is found to be smaller than a certain pre-defined threshold, this algorithm will not check the rest of the block modes. Similar to the reference frame early termination in ME state 3, 13 sub-states are further defined with increasing threshold values used for block mode early termination.

5) ME state 5: In this state, similar search techniques as in ME state 4 are used. Besides these, an adaptively adjusted search window size [8] is also used for checking the reference frames other than the temporal nearest one. If the algorithm uses a smaller search window for farther reference frames, the ME process can be completed earlier. Hence 13 sub-states are further defined with decreasing search window size used for checking farther reference frames.

B. Core Procedure of the Proposed Algorithm

Let \( C_f \) be the target computational budget allocated to code each video frame and \( C_f \) be the actual number of operations required by the ME algorithm to code the \( f^{th} \) video frame. The pseudo-code for determining the transition between different ME states is described as follows:

Step 1: After coding the \( f^{th} \) video frame, the following \( x_1, x_2 \)
From the 21st frame onwards, the computational constraint is a frame by frame un-constrained H.264 lossless SEA+PDS. Let $C_1$ be the average number of operations to code SEA+PDS (i.e. ME state 1) without any computational hardware based H.264/AVC verification model JM11.0[9] for coding various frames will be coded. After coding ($j+1$)th frame, the following $x_3$, $x_4$, and $C_{avg}'$ are computed:

$$x_3 = \frac{1}{3} \sum_{i=j}^{j+3} C_i, \quad x_4 = \frac{1}{3} \sum_{i=j}^{j+4} C_i$$

$$C_{avg} = \frac{1}{2}(x_3 + x_4)$$

IF ($x_2 > 1.2*C_1$) or ($C_{avg} > C_T$) THEN
Change_To_Faster_ME_state($x_3$); Go to Step 2;
ELSE
IF ($x_2 < 0.8*C_1$) and ($C_{avg} < 0.85*C_1$) THEN
Change_To_Slower_ME_state($x_3$); Go to Step 2;
ELSE
/* No need to change the ME strategy */
Repeat this step 1 after coding ($j+3$)th frame.
ENDIF
ENDIF

Step 2: After changed the ME state or sub-state, six more frames will be coded. After coding ($j+6$)th frame, the following $x_3$, $x_4$ and $C_{avg}'$ are computed:

$$x_3 = \frac{1}{3} \sum_{i=j}^{j+3} C_i, \quad x_4 = \frac{1}{3} \sum_{i=j}^{j+4} C_i$$

$$C_{avg}' = \frac{1}{2}(x_3 + x_4)$$

IF (ME state has been changed to faster state previously) THEN
IF ($x_2 > x_3$) and ($x_2 > x_4$) ) or ($C_{avg} > C_{avg}'$) THEN
Go to step 1;
ELSE
Change_To_Faster_ME_state($x_3$);
Repeat step 2 after coding ($j+12$)th frame;
ENDIF
ELSE
/* ME state has been changed to slower state previously */
IF ($x_2 < x_3$) and ($x_2 < x_4$) ) or ($C_{avg} < C_{avg}'$) THEN
Go to step 1;
ELSE
Change_To_Slower_ME_state($x_3$);
Repeat step 2 after coding ($j+12$)th frame;
ENDIF
ENDIF

III. EXPERIMENTAL RESULTS

In this paper, we have proposed a computationally-scalable motion estimation algorithm for H.264/AVC video coding in which we can dynamically control the ME process to be completed within a pre-allocated computational budget. Such computational scalability is embraced into the firmware of the multimedia platforms for video products, and allows real-time control of computational budget for various video compression applications.

REFERENCE