AN EFFICIENT IMPLEMENTATION OF UNRESTRICTED MOTION COMPENSATION IN VIDEO ENCODER

Krit Panusopone, Yinqing Zhao, and Limin Wang

Motorola Inc., Connected Home Solutions
6420 Sequence Drive, San Diego, CA 92121

ABSTRACT

This paper proposes an implementation method to reduce complexity of motion estimation in the padded areas for unrestricted motion compensation. The proposed method avoids repeating the same calculation of motion estimation cost in the padded areas with identical search outcome as the conventional motion estimation. The savings realized in this method depends only on the block size, image size, and the search range. The proposed method can be applied to several video compression standards such as H. 263, and H.264.

Index Terms— Video codecs, Video coding, Motion compensation, Standards

1. INTRODUCTION

Motion Estimation (ME) is widely used in video compression algorithms. ME is a computationally intensive process for video encoder, especially for the recently approved H.264 [1]. H.264 allows several flexibilities (variable block size, multiple references, and unrestricted motion compensation) in temporal prediction process which helps improve the coding efficiency but adds considerable complexity to the overall process. This paper proposes an efficient method to reduce complexity of full search ME in the padded area. By exploiting the definition of the padding process, the proposed method reuses partial ME cost results in calculating the final ME cost in the padded area without any loss in coding efficiency.

2. UNRESTRICTED MOTION COMPENSATION

Unrestricted Motion Compensation (UMC) was first introduced in H.263 standard [2], and it has since been an integral part of all subsequent video coding standards (e.g., MPEG-4 part 2 [3], H.264 [1], VC-1 [4]). UMC allows the encoder to select motion vector (MV) pointing outside the boundary of the reference picture. This feature is useful when coding an object that is moving in and out of the picture. Video coding standard has to define the intensity value of a pixel outside the picture so that it is identical for both encoder and decoder to maintain synchronization of the reconstruction outputs.

In general, padding is used to determine the pixel values outside the reference picture. In the padding process, the value of a pixel on the boundary of the reference picture is repeated for all the pixels outside the boundary along either the vertical or horizontal direction. Reference picture is padded so that it is large enough to cover all possible MVs. There are several methods to store the padded area in the reference buffer.

Another method for ME outside the reference picture is to clip the motion vector [5]. Since the pixel value in the padded area is the same as the value of the pixel on the boundary, ME uses that pixel value to determine matching cost instead of using the actual out-of-bound pixel.

With either of the above two methods, the same ME process (e.g., full search, fast search) inside the reference picture can be extended into the padded area. However, this simple extension approach is inefficient as it wastes too many computational cycles in repeating the same calculations over the padded area. The main computational block in ME process is the calculation of the difference between the current block and a candidate prediction block pointed by a motion vector, for example, the sum of absolute difference (SAD).

This paper proposes a method that reduces complexity of ME in the padded area by reusing some partial calculation outputs in subsequent operations. The proposed method exploits the fact that the same operations between pixels in the current block and pixels in the padded area occur repeatedly. These calculation results can be stored and reused to eliminate repeating of the same operations in the future. The following sections describe the method to utilize these partial results efficiently in the right, left, top, and bottom padded areas.

3. THE PROPOSED METHOD

Assume that the reference pictures are of size $M \times N$ pixels and the search window radius $\pm W_x$ pixels horizontally and $\pm W_y$ pixels vertically with its center at (C_x, C_y), as shown in Fig. 1. In the following explanation, SAD is used in measuring the ME cost and the raster scan is used in searching the best MV over the search window, that is, the

search window is traversed from left to right, and top to bottom. Given a block of $m \times n$ pixels at location (u,v), the corresponding SAD with an associated $MV(x,y)$ is calculated as

$$SAD(x,y) = \sum_{j=0}^{n-1} \sum_{i=0}^{m-1} |X(u+i,v+j) - \hat{X}(u+i+x,v+j+y)| \quad (1)$$

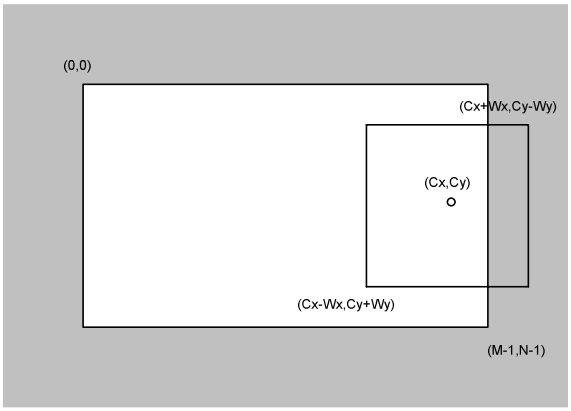

Figure 1: Example of a ME in unrestricted MC mode (shaded area indicates padded area)

3.1. Right padded area

With a raster scan search pattern, the searching process crosses the right boundary of the reference picture from the picture area into the padded area. The SAD calculation for ME in the right padded area can be improved by reusing some partial results. Let $\Delta_i(x,y)$ be the sum of the absolute pixel differences in the vertical dimension, that is,

$$\Delta_i(x,y) = \sum_{j=0}^{n-1} \left| X(u+i,v+j) - \hat{X}(u+i+x,v+j+y) \right| \quad (2)$$

where Δ_i, $i = 0,1,...,m-1$ can be considered as partial ME costs.

For a block of $m \times n$ pixels at location (u,v), the prediction block pointed by $MV(M-m-u,y)$ is the last candidate before the searching moving into the padded area. The SAD for $MV(x,y)$, $x > M-m-u$, can be computed efficiently as follows:

1. Set $m' = m$ and $\Delta = 0$
2. Calculate the SAD with $MV(M-m'-u,y)$ by
$$SAD(M-m'-u,y) = \Delta + \sum_{i=0}^{m'-1} \Delta_i(M-m'-u,y)$$
3. Set $\Delta = \Delta + \Delta_{m'-1}(M-m'-u,y)$ and $m' = m'-1$
4. If $m' \geq 1$, return to step (2), otherwise, continue to step (5).
5. For all $MV(x,y)$ with $x \geq M-u$, $SAD(x,y) = \Delta$.

The proposed method produces an identical result as the traditional method. It can work with any other cost functions (e.g., MSE, MAD) and with any block size (e.g., 4x4, 8x8, 16x16). The proposed method can be summarized by the flowchart shown in Fig. 2.

The savings realized in the right padded area in terms of number of pixel difference, $S(Right)$, can be determined as follows:

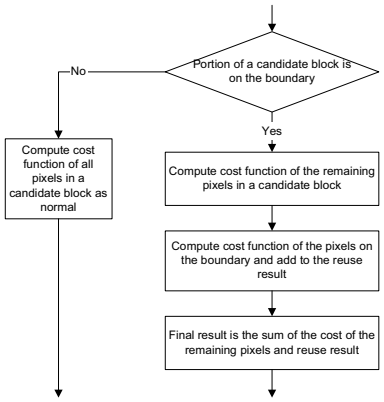

Figure 2: Flowchart for right padded area

$$S(Right) = S(Outside) + S(Transition)$$

$$S(Outside) = [2W_x + 1 - B_x] \times [2W_y + 1] \times m \times n$$

$$B_x = \begin{cases} 1 & if \quad C_x - W_x \geq M-1 \\ M - C_x + W_x & if \quad M - W_x - 1 \leq C_x < M + W_x - 1 \\ 2W_x + 1 & if \quad C_x + W_x < M-1 \end{cases}$$

$$S(Transition) = [2W_y + 1] \times [\sum_{t=1}^{T} t \times n]$$

$$T = \begin{cases} 1 & if \quad C_x + W_x \leq M - m + 1 \\ C_x + W_x + m - M & if \quad M - m + 1 < C_x + W_x \leq M-1 \\ m-1 & if \quad C_x + W_x > M-1 \end{cases} \quad (3)$$

3.2. Left padded area

For the padded area on the left, in a raster scan fashion, the ME process crosses the left boundary of reference picture from the padded area into the picture area. Since the search pattern (raster scan) encounters the padded area first, it acts in an opposite manner compared to ME in right padded area. The proposed method for the left padded area hence works in an opposite way. That is, for left padded area, partial ME cost is subtracted as the search traverses into reference picture area.

For a block of $m \times n$ pixels at location (u,v), all $MV(x,y)$ pointing outside the reference picture with $x \leq -(u+m)+1$ have the same SAD value. The SAD for $MV(x,y)$ with $x > -(u+m)+1$ can be calculated as follows.

1. Set $m' = m$ and $\Delta = \sum_{i=0}^{m-1} \Delta_i(-(u+m)+1, y)$,
2. For all $MV(x, y)$ with $x \le -(u+m)+1$, $SAD(x, y) = \Delta$.
3. Set $m' = m'-1$ and $\Delta = \Delta - \Delta_{m'-1}(-(u+m')+1, y)$.
4. Calculate the SAD with $MV(-(u+m')+1, y)$ by

$$SAD(-(u+m')+1, y) = \Delta + \sum_{i=m'}^{m-1} \Delta_i(-(u+m')+1, y)$$

5. If $m' > 1$, return to step (3), otherwise, stop.

Notice that the partial ME cost is first subtracted at position *(-(u+m)+1,y)*. This position is where the candidate block first moves inside the picture boundary (i.e. its rightmost pixel is at *(0,y)* position). Compared to the algorithm for the right padded area, the order of the operation for the left padded area is reversed, i.e., the SAD of the entire block is computed first (step 1) and then partial ME cost is iteratively subtracted (step 3). Fig. 3 shows the flowchart of the modified algorithm for the left padded area.

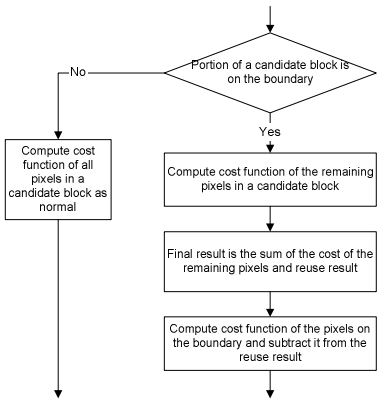

Figure 3: Flowchart for left padded area

The savings realized in the left padded area in terms of number of pixel difference, *S*(*Left*), can be determined as follows:

$$S(Left) = S(Outside) + S(Transition)$$

$$S(Outside) = [2W_x + 1 - B_x] \times [2W_y + 1] \times m \times n$$

$$B_x = \begin{cases} 1 & if \quad C_x + W_x \le -(m-1) \\ C_x + W_x + m & if \quad 1 - m - W_x < C_x \le 1 - m + W_x \\ 2W_x + 1 & if \quad C_x - W_x > -(m-1) \end{cases}$$

$$S(Transition) = [2W_y + 1] \times [\sum_{t=1}^{T} t \times n]$$

$$T = \begin{cases} 1 & if \quad C_x - W_x > -1 \\ -(C_x - W_x) & if \quad -(m-1) < C_x - W_x \le -1 \\ m-1 & if \quad C_x - W_x \le -(m-1) \end{cases} \quad (4)$$

3.3. Top and bottom padded areas

With a raster scan manner, ME for the top and bottom padded areas has one major difference from the left and right padded areas in that the partial ME cost are different as the search traverses from one pixel to the next in the padded area of the same row. As a result, searching through the top and bottom padded areas in raster scan order does not allow the partial ME cost to be reused and hence requires the same amount of computation as working inside the picture boundary.

One way to maintain the effectiveness of the proposed method is to transpose both the current block and the associated search area when traversing in the top and bottom padded areas. In the transposed domain, searching in the top and bottom padded areas behaves the same as the left and right padded areas so that UMC in the top and bottom padded areas can be done efficiently by reusing partial ME cost results as described in the previous sections.

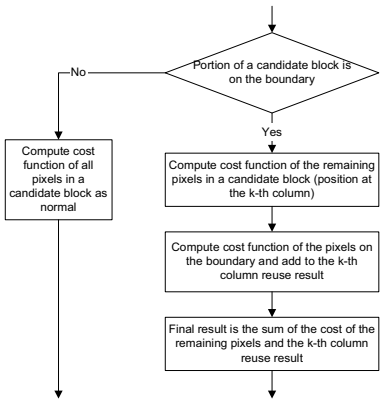

Figure 4: Flowchart for top padded area

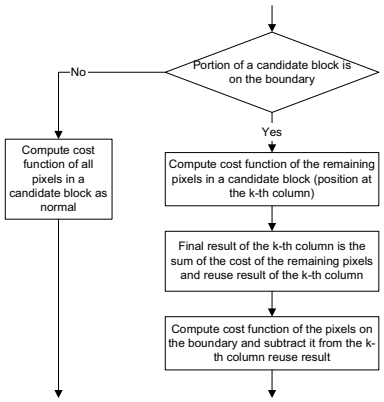

Figure 5: Flowchart for bottom padded area

Another solution for UMC in the top and bottom padded areas is to store all partial ME costs of the entire row in the search window. As the search traverse through the padded area, the partial ME costs in the same column

position are updated. This solution works without any change in the search pattern. Figures 4 and 5 show the flowchart of this solution for the top and bottom padded areas, respectively. The savings realized for top and bottom padded areas can be derived in the same way as for left and right padded areas and will be omitted in this paper due to the space limitation.

3.4. Field picture

In H.264, the padding process for field and frame macroblock (MB) is slightly different for the top and the bottom padded areas. For field MB, the top and the bottom padded areas are formed by repeating two boundary lines (one is in top field and the other in bottom field) alternatively. Hence, it is necessary to modify the proposed method in the top and the bottom areas for field MB. Specifically, two sets of partial ME costs need to be stored and reused in calculating the final ME costs, one is for the top field and the other for the bottom field.

4. IMPLEMENTATION

This section describes an implementation of the proposed algorithm in video encoder. One main difficulty is the region where redundancy can be exploited through both horizontal (right or left) and vertical (top or bottom) padded directions. To handle this issue, we divide the search window into three parts: top, middle, and bottom region. Top region covers all candidates in search window such that their vertical coordinates locate beyond the upper boundary of the reference picture. Similarly, bottom region includes all candidates in search window such that their vertical coordinates locate beyond the lower boundary of the reference picture. The remaining candidates belong to the middle region.

Redundancy in the middle region can be handled by the same procedure described in sections 3.1 and 3.2. Top and bottom regions are handled differently to exploit redundancy without sacrificing too much complexity. Transposed version described in section 3.3 is used in top and bottom regions.

5. EXPERIMENTAL RESULTS

We implemented two efficient ME algorithms to determine their performance. The first algorithm (METHOD 1) uses the idea illustrated in Sections 3.1 and 3.2. The second algorithm (METHOD 2) uses the idea from Section 4. ME using clipping method similar to [5] is also implemented to serve as a benchmark. In these simulations, SAD is used as the matching cost and search center is aligned with the coordinate of the current block. Table 1 demonstrates the savings of METHOD 1 and METHOD 2 in terms of relative reduction in pixel difference operations over the benchmark.

According to the simulation results, we can see that METHOD 2 achieves the most savings. The result demonstrates the impact of redundancy that can be exploited jointly in both horizontal and vertical padded areas. Moreover, it shows that the savings increases as the search range increases but decreases as the picture size increases, which reflects the ratio of the search area outside the reference picture to the picture size. The MV outputs of both methods are identical to those of the benchmark.

TABLE 1: SAVINGS IN PIXEL DIFFERENCE OPERATIONS

METHOD	PICTURE SIZE	SEARCH RANGE		
		64X64	32X32	16X16
METHOD 1	SD	6.7%	3.4%	1.7%
	CIF	11.2%	5.6%	2.9%
	QCIF	22.4%	11.3%	5.7%
METHOD 2	SD	10.9%	5.5%	2.8%
	CIF	19.2%	9.9%	5.1%
	QCIF	36.0%	19.2%	10.0%

6. CONCLUSION

This paper describes an efficient method for UMC in the padded area using the reused results. The proposed method significantly reduces the computation cost without any loss in search quality. It does not incur significant overhead. Moreover, it maintains the best practice of the straightforward UMC. First, it checks the boundary condition once per every candidate (not for every pixel). Second, it does not require any additional memory to store padded area since all SAD computation in the padded area comes from reused result.

7. REFERENCES

[1] "Information Technology – Coding of Audio-Visual Objects – Part 10: Advanced Video Coding," *ISO/IEC International Standard 14496-10*, June 2003.

[2] ITU-T Recommendation H.263, "Video Coding for Low Bit Rate Communication," Jan. 1998.

[3] ISO/IEC JTC1, 2000, "Coding of audio-visual objects – Part 2: Visual," *ISO/IEC 14496-2*, Feb. 2000.

[4] "VC-1 Compressed Video Bitstream Format and Decoding Process," *SMPTE 421M-2006*, Apr. 2006.

[5] "Video Acceleration API Software Developer's Guide," *White Paper for the Intel 2700G Multimedia Accerelator,* Intel, Jan. 2005.