An Iterative Algorithm for Accurate Motion Estimation in Very Low Bit Rate Video Coding

David Wendi Pan
Dept. of ECE
Univ. of Alabama in Huntsville
Huntsville, AL 35899
Tel: 1-256-824-6642
dwpan@ece.uah.edu

Andrea Basso
Dept. of ECE
University of Victoria
P.O. Box 3055, Victoria, B.C.
V8W 3P6 Canada
abasso@ece.uvic.ca

Leon Bottou
NEC Research Institute
4 Independence Way
Princeton NJ 08540
Tel: 1-609-951-2732
leon@bottou.org

ABSTRACT

In video coding, accurate motion estimation is very important since good temporal prediction can significantly eliminate temporal redundancy and save bits in coding the motion. In block-based motion estimation systems, we can increase the estimation accuracy by using smaller block sizes. However, more bits are required to code the motion information due to increased number of blocks. It is desirable to be able to set a quota on the total number of motion blocks in very low bit rate coding. In this paper, we propose an iterative algorithm that seeks to achieve very accurate estimation on moving objects, while ensuring the number of motion blocks does not exceed the given quota. Simulations demonstrate the effectiveness and robustness of the proposed method.

Keywords

Motion estimation, video coding

1. INTRODUCTION

Motion Estimation is an important part of video compression systems, where the estimated motion vectors are used to produce a motion-compensated prediction of a frame to be coded from a previously coded reference frame [1]. Our goal is to minimize the total number of bits used to specify the motion. In very low bit rate video coding, accurate estimation of the motion between frames becomes more important since it can lead to better temporal prediction and thus temporal redundancy can be eliminated to the greater extent. On the other hand, if the estimated motion is not an accurate representation of the true physical motion, rather than saving bits by coding the motion compensated frame difference, we waste bits in coding the false motion blocks caused by inaccurate motion vectors.

Block-based motion estimation has been adopted in many international standards such as MPEGx and H.26x, etc. Generally, the smaller the block size, the more accurate the motion vector is in representing the motion of each pixels within the block. However, smaller block size leads to greater number of motion vectors, which in turn costs more bits in coding. For instance, an image of CIF format contains $6,336$ blocks (4×4). We may desire to set a quota on the number of motion blocks to be coded in very low bit rate video coding applications. We attempt to find the set of "best" motion blocks, for which bits will be allocated to code their motion vectors and the associated displaced block differences. We treat the rest of the blocks as still blocks and thus bits for coding the motion vectors are saved.

Therefore, motion estimation should be focused on the moving objects in an image scene. Typically, there are multiple moving objects. Region-based motion estimation has been proposed to segment the image frame into several regions and estimate the motion parameters of each region [3]. However, conventional motion-based segmentation approaches have inherent difficulty since motion field tends to be noisy and difficult to interpret (see Figure 1).

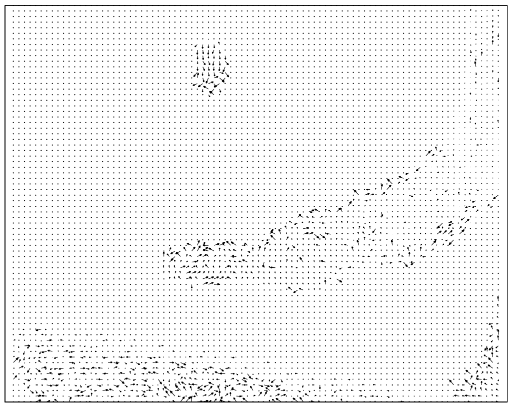

Figure 1: An example of motion vector field obtained after motion estimation (full search, search window size ± 15 pixels) of the third frame based on the second frame of the "Table Tennis" sequence.

In this paper, we propose an iterative algorithm that is capable of detecting moving regions and get very accurate motion estimation on them. In the meanwhile, the number of motion blocks are kept below a given quota.

The paper is organized as follows. Section 2 describes the iterative algorithm. Section 3 presents the simulation results. Section 4 gives a brief summary of the paper.

2. ALGORITHM

First, we take the difference of two neighboring frames I_t and I_{t-1} and use an *a priori* threshold $\mathcal{H}$ to separate the moving foreground from the still background. We divide each $M \times N$ frame into (MN/B^2) blocks (of size $B \times B$), and classify each block into two mutually exclusive categories: (i) *motion* blocks in the foreground, and (ii) *still* blocks in the background.

Definition 1. A block in the frame I_t is said to be a motion block if there exits at least one pixel $I_t(x, y)$ in the block that satisfies the following condition:

$$|I_t(x, y) - I_{t-1}(x, y)| \geq \mathcal{H}, \tag{1}$$

A block is decided to be a still block otherwise. A bit map $\mathcal{T}$ that characterizes the block types can be formed:

$$\mathcal{T}(m, n) = \begin{cases} 1 & \text{the (m,n)-th block is a motion block;} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $m \in [0, M/B)$ and $n \in [0, N/B)$.

We obtain the motion vectors associated with each motion block. For the block (m, n), we measure the prediction error in terms of *mean square error* (MSE). The motion vector (v_x, v_y) of block (m, n) are found to be

$$\arg \min_{v_x, v_y \in [-R, R]} \left\{ \sum_{i,j=0}^{B-1} [I_t(x, y) - I_{t-1}(x + v_x, y + v_y)]^2 \right\}, \tag{3}$$

where R is the search window size in motion estimation, and $x = m \times B + i$, $y = n \times B + j$ are the coordinates of the pixel.

In the special case where a block has $(v_x, v_y) = (0, 0)$, we claim the block to be a still block.

If we choose a large threshold $\mathcal{H}$, then only a small percentage of blocks will be classified as motion blocks – some true motion blocks might be left out. On the other hand, if we choose a small $\mathcal{H}$, then many blocks will be labeled as motion blocks ($\mathcal{H} = 0$ will turn all blocks into motion blocks). In very low bit rate video coding applications, we cannot afford to code too many motion blocks. An attempt to find motion vectors for an excessive number of blocks is not only computationally expensive, but also wasteful since we would have to force some blocks to become still blocks in order to save bits in coding the motion vectors.

Hence our goal is to identify a set of c motion blocks in frame I_t that have the best match in the previous frame I_{t-1}, where $c \leq \mathcal{Q}$, and $\mathcal{Q}$ is the quota. Here, best match motion block is defined as a block with the smallest MSE (Eq. 3). Given the initial set of c ($<< \mathcal{Q}$) candidate motion blocks produced by the frame difference operation, the following dilution algorithm seeks to recover the most suitable motion blocks that were classified as still blocks initially.

We use a sorted linked list (Figure 2) to store the information of motion blocks, with the key of a node being the MSE associated with the motion vector (a smaller key value means a better candidate motion block). We insert the initial set of motion blocks into an empty list in an ascending order. We traverse the list from its head (smallest MSE) to its tail (largest MSE).

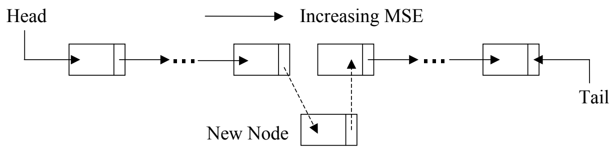

Figure 2: Sorted Linked List. A new node is inserted in such a way that the list remains sorted in an ascending order.

For each node we visit, we consider its neighboring eight blocks in frame I_t (Figure 3), among which some blocks may have already been included in the list. We ignore such blocks. For those that are not in the list, motion estimation is applied and motion vectors are obtained. Based on the associated MSE values, new blocks are inserted into the list with the updated list remaining sorted (Figure 2).

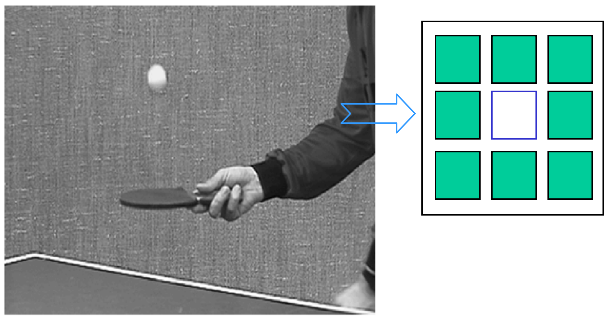

Figure 3: Dilution operation: eight neighbors of a node in the linked list are examined for potential inclusion into the list.

The same procedure is then repeated on an updated list. The iteration stops if the number of blocks in the list hits the quota, or we can not find any more motion blocks that are eligible to join the list. The algorithm is summarized in Figure 4.

It can be seen that this method of progressive refinement of the motion estimation has the following properties: (i) Blocks that are in the list (motion blocks) are more important than blocks that are not in the list (still blocks). Thus we should code the motion vectors of blocks in the list since they allow for the best reduction in temporal redundancy; for blocks that are excluded in the list, we set their motion vectors to zero and only code the non-displaced block differences. (ii) If an moving object occupies multiple connected blocks, dilution in the vicinity of a node would bring in good new motion blocks. The list will grow at wherever the true

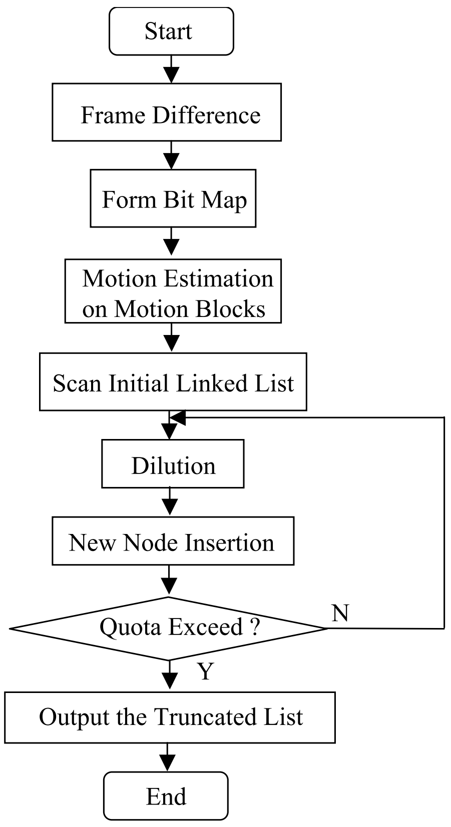

Figure 4: Flowchart of the algorithm.

motion is most likely to occur. (iii) The closer a block is to the head of the list, the more important it is (smaller MSE). When we truncate the list at an intermediate node, the partial list output from the head to that node will yield a set of best motion blocks possible in the given number. (iv) Motion estimation is applied only when necessary (to consider recruiting candidate nodes during dilution), thereby achieving significant savings on the computational cost of having to motion estimate every block. Note that if the initial list is not too long (which is typically the case when we use a relatively large threshold $\mathcal{H}$), then the sorting would not be a large computational overhead. In summary, our method can achieve very accurate estimation of the moving objects in the scene, while meeting the budget requirement on the number of motion blocks.

3. EXPERIMENTAL RESULTS

The block size is chosen to be 4×4. This choice is a compromise of motion estimation accuracy and the overhead of coding excessive number of motion vectors. For instance, 4×4 block size has been adopted by the emerging H.26L standard [2]. We use the second and the third frame of the "Table Tennis" sequence to show the simulation results. Similar results have been observed for other frames and test sequences.

In Figure 5, the initial linked list contains only 447 motion blocks. As we increase the quota from 447 to 802, the output partial list grows in length. Consequently, more and more motion blocks that were classified as still blocks on the initial bit map are recovered so that we can finally detect moving objects as a whole (e.g., the arm, the bat, and the table) when the quota reaches 1507. As we further increase the quota, no additional blocks are found to be eligible to join the list. This is when the iteration stops. By examining the final full list output, we are able to identify moving objects, each of which corresponds to a cluster of motion blocks. In other words, if we do not introduce a quota constraint, then the linked list is allowed to grow freely to its largest possible length - the dilution algorithm behaves like an effective motion-based segmentation approach.

Figure 6 and Figure 7 illustrate how the associated root MSE values increase as more and more nodes are inserted into the list. Note that for both thresholds, the total numbers of motion blocks attainable are almost the same (about 1510).

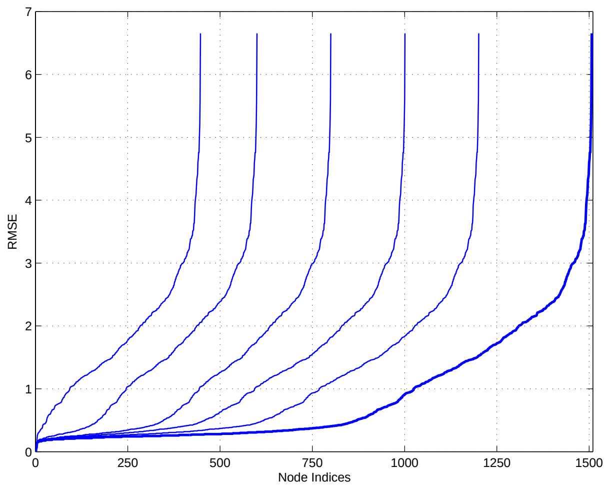

Figure 6: Square root of MSE associated with each node in the linked list. Threshold $\mathcal{H} = 30$, and $\mathcal{Q} = 447, 603, 802, 1002, 1200, 1507$, respectively. $\mathcal{Q} = 1505$ (thicken curve) corresponds to the longest list possible.

Figure 8 also demonstrates that the final number of motion blocks (divided by the total number of blocks in a frame) tends to be insensitive to the threshold used.

Figure 9 shows the tradeoff between number of the motion blocks allowed and the accuracy of temporal prediction (as measured by $PSNR = 10 \log_{10} \frac{255^2}{MSE}$). For large threshold ($\mathcal{H} = 30$), the initial small set of motion blocks (only 7.1% of the blocks are labeled as motion blocks) is a very rough estimate of the motion, therefore, as more and more motion blocks are added to the list, the prediction accuracy keeps improving (over 1dB increase in PSNR can be achieved with about 24% of the blocks being labeled as motion blocks). By contrast, the improvement is less pronounced for $\mathcal{H} = 10$ since we start with a much larger set (11.3%) of motion blocks. Again, both thresholds can converge to the same prediction accuracy eventually.

4. SUMMARY

This paper describes an iterative dilution algorithm useful in very low bit rate video coding applications. The algorithm

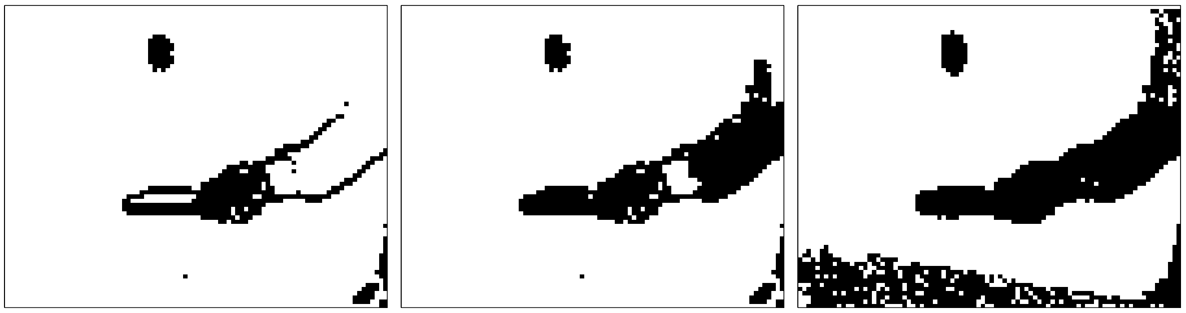

Figure 5: Bitmaps (threshold $\mathcal{H} = 30$). Quota $\mathcal{Q} = 447, 802, 1507$ (from left to right).

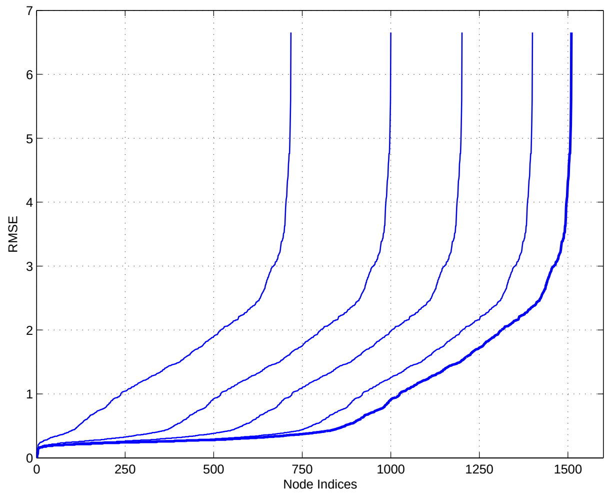

Figure 7: Square root of MSE associated with each node in the linked list. Threshold $\mathcal{H} = 10$, $\mathcal{Q} = 718, 902, 1002, 1200, 1400, 1510$, respectively. $\mathcal{Q} = 1510$ (thicken curve) corresponds to the longest list possible.

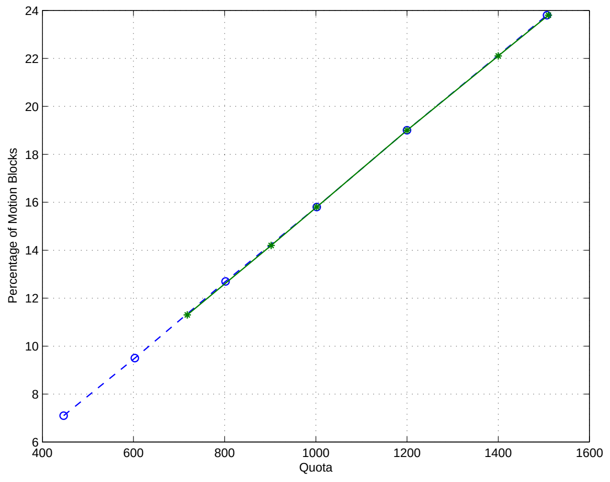

Figure 8: Percentage of motion blocks. Threshold $\mathcal{H} = 10$ (solid), and $\mathcal{H} = 30$ (dashed).

is effective in tracking down moving objects, while keeping the total number of motion blocks below a given limit. The algorithm is robust since its final output is insensitive to the thresholds used in block classification. We show that our algorithm is also a good moving object segmentation scheme when operating in an unconstraint manner.

5. ACKNOWLEDGMENTS

The idea of this paper was formed when the authors were working at AT&T Labs-Research.

6. REFERENCES

[1] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall. *MPEG Video Compression Standard*. Chapman & Hall, 1996.

[2] J. Shen and J. Ribas-Corbera. Benefits of adaptive motion accuracy in H.26L video coding. In *Proc. of International Conference on Image Processing*, pages 1012–1015, 2002.

[3] Y. Wang, J. Ostermann, and Y.-Q. Zhang. *Video Processing and Communications*. Prentice Hall, 2002.

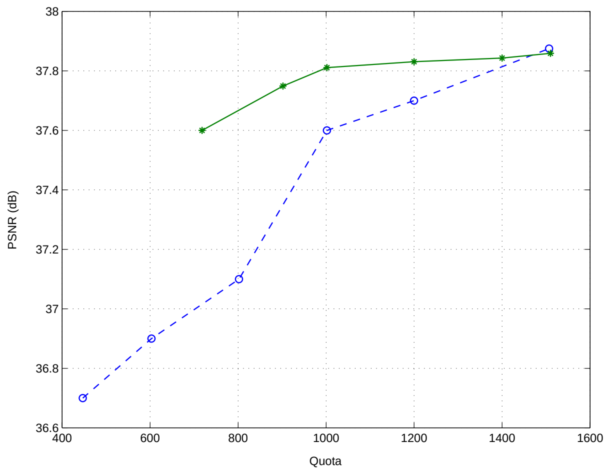

Figure 9: PSNR improvements. Threshold $\mathcal{H} = 10$ (solid), and $\mathcal{H} = 30$ (dashed).