Coding of Motion Compensation Residuals Using Edge Information

Yao Wang
Dept. of Electrical Engineering
Polytechnic University
Brooklyn, NY 11201
yao@vision.poly.edu

Michael T. Orchard
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08540
orchard@ee.princeton.edu

Abstract

In most current video coders, a block is first predicted from its best matching block in a previous frame, and the prediction error is then coded using Discrete Transform Coding (DCT). Because of the inadequacy of the block-wise translational motion model, edges in the predicted block are often shifted from their true positions, leading to errors that are clustered around edges in the predicted block. DCT is inefficient for coding such errors. Independent searching of block motion vectors also lead to discontinuities of edges across block boundaries. Existing coders ignore such correlation between error location and edge discontinuity. In this paper, we describe a coder that corrects edge-misalignment before applying DCT coding. The correlation between edge-discontinuity and edge-misalignment is exploited in the coding of the misalignment parameters.

1 Introduction

Presently, the most popular and successful technique for video coding uses block-based motion-compensated prediction and DCT coding of motion compensation residuals. A widely recognized problem with this approach is that the DCT coding method is not well matched with the characteristics of the motion compensation error. The DCT coding method is best suited for coding a random field that can be characterized well by a Markov Random Field (MRF) [1]. Motion compensation errors have much of their energy clustered around edges in the scene, corresponding to misalignment of edges. Such characteristics are not well matched to a MRF model, and DCT coding is very inefficient for coding such errors.

The key observation that has lead to our current investigation is that the locations and shapes of the errors due to misalignment of edges are highly predictable. Specifically, prediction errors due to edge-misalignment are collocated with edges in the predicted frame, appearing as narrow stripes that are oriented in parallel with reference edges. The error spread in the orthogonal direction of the edge depends on the actual misalignment, whereas the magnitude of the error depends on the gradient of the edge. Furthermore, when motion estimation is accomplished by assuming a block-wise translational model and motion vectors are obtained by minimizing the prediction error within each block independently, misalignment of edges usually lead to edge discontinuities at block boundaries in the predicted frame.

As an illustration of the above observations, Figures 4(a)-4(d) show two typical frames (portion) in the sequence “football” and the prediction result using half-pel accuracy block motion estimation. The alignment of error clusters around the edges in the predicted frame and the correlation between prediction error locations and edge discontinuities are obvious. Because the decoder has access to the predicted frame after receiving the motion parameters, the above information is available in both the encoder and the decoder. The conventional approach for coding the residual ignores the information carried in the edges of the predicted image. In this work, we explore how to avoid such information leakage in video coding. Towards this goal we have developed an approach for estimating the edge-misalignment information. We also developed an algorithm for detecting edge discontinuities across block boundary and for predicting misalignment based on detected discontinuities. These algorithms have been incorporated in a block-based coder, which first corrects errors due to edge-misalignment in a residual block, before applying DCT coding. The correlation between edge-discontinuity and edge-misalignment is exploited in the coding of the misalignment parameters. In the remaining sections, we first describe our algorithms for estimating edge-misalignment and for detecting edge-discontinuities. We then describe the coding scheme and show some preliminary simulation results.

*This work was performed when the first author was on sabbatical leave at Princeton University. This work was supported in part by the National Science Foundation under Grant MIP-9616716.
2 Edge-Misalignment Estimation

**Misalignment Function Parameterization** Consider a block $B$ which has $K$ edges. Let $e_{k,l}$ represent the location of the $l$-th edge point in the $k$-th edge. Suppose the correct position for this edge is $e_{k,l}^*$, then the misalignment at this point is defined as $m_{k,l} = e_{k,l} - e_{k,l}^*$. Clearly, this misalignment in general can have an arbitrary direction. However, for a more compact representation, we only specify its projection along the gradient direction. Let $g_{k,l}$ denote the direction vector of the gradient at $e_{k,l}$, we parameterize the misalignment at this point by $m_{k,l}g_{k,l}$, as shown in Figure 1. Because the gradient information is known, the misalignment information for a given edge is entirely described by the one-dimensional scalar function, $m_{k,l}, l = 1, 2, \ldots, L_k$, where $L_k$ is the length of $k$-th edge. The misalignment function is in general a smooth function. This is because the actual motion of an edge is usually smooth, and when it is being approximated by a single translation, the misalignment in adjacent points on the same edge should follow a smooth curve. For example, if the real edge is a straight line and has undergone a shift and rotation, the misalignment function along this edge can be completely characterized by a linear function.

**Misalignment Correction** To correct misalignment along all edges in $B$, we describe the expected contribution of the misalignment at every edge $e_{k,l}$ to any pixel $p$ by a weighting function $w_{k,l}(p)$. Because the original block motion vector (MV) is estimated with edge present, the background region (the block minus the edge region) may be better estimated with a different MV after correcting misalignment at edges. Therefore, we also allow a mismatch MV, $m_0$, with weight function, $w_0(p)$, to be assigned to the background. The weighting functions satisfy $\sum_{k,l} w_{k,l}(p) + w_0(p) = 1$. Let $f_r(p)$ represent the reference frame color function, and $v$ the block MV (for non-translational model, $v$ will be replaced by $v(p)$). The misalignment-corrected value at $p$ is given by

$$\hat{f}(p) = \sum_{k,l} w_{k,l}(p)f_r(p + v + m_{k,l}g_{k,l}) + w_0(p)f_r(p + v + m_0), p \in B.$$  

**Misalignment Estimation** To estimate the misalignment functions for all edges in block $B$, we minimize the error

$$E(m_{k,l}, \forall k,l, m_0) = \sum_{p \in B} |f(p) - \hat{f}(p)|^p.$$  

Ideally, we should find the best set of $m_{k,l}$ for all $k,l$ and $m_0$ simultaneously that minimizes the above error. To reduce the computation, we search individual $m_{k,l}$ separately by minimizing the error associated with the $(k,l)$-th edge point:

$$E_{k,l}(m_{k,l}) = \sum_p |w_{k,l}(p)|^p|f_2(p) - f_1(p + v + m_{k,l}g_{k,l})|^p.$$  

Note that if the weights are non-overlapping, i.e.,

$$w_{k,l}(p)w_{k',l'}(p) = 0, k \neq k', l \neq l',$$

then minimizing individual errors is equivalent to minimizing the total error. In our simulation we used $p = 1$. The minimum for $m_{k,l}$ is found by using an exhaustive search within a search range of $-3$ pixels, at a search step of half pixel. We did not implement the estimation of $m_0$. This algorithm is only applied to edges which has an initial error (the sum of the errors associated with all edges on this edge) greater than a threshold.

Figure 5 illustrates a typical error block from a real video sequence. The block has three edges. One of the edge has undergone a rotation with respect to the estimated translation vector, creating a butterfly pattern (cf. Fig. 5(c)). Fig. 5(d) shows the extracted edges from the predicted block. Fig. 5(e) shows the weight function for the second edge. The estimated misalignment function for this edge is a linearly increasing function, as expected. The other two edges have very small errors associated with them and therefore are not subject to misalignment estimation and correction. The block after applying misalignment correction and the new error block are shown in Fig. 5(f). We can see that the misalignment of the second edge has been successfully corrected. The remaining error is not due to edge-misalignment.

Going back to the example given in Fig. 4, Fig. 4(e) shows the image obtained after misalignment correction, and Fig. 4(f) the new residual image. It can be seen that most narrow stripe patterns in the original residual image due to edge-misalignment are gone. Most of the remaining errors are due to uncovered regions and changes in reflection and shadow patterns. There are still visible thin lines in the error image, which are caused by the slight errors in the estimated misalignment values.
3 Discontinuity Detection and Misalignment Prediction

Detection of Edge Discontinuities Across Block Boundaries For every edge in a block boundary, e, we try to determine whether it continuously extends to its adjacent block, has discontinuity with its corresponding edge in the adjacent block, or does not belong to either category (other). In the last case, the edge that e belongs may have terminated at e, or changed its direction at the boundary. Figure 2 illustrates our detection scheme, where e' is the expected extension of e in the adjacent block (determined based on the gradient direction g at e), and p is the actual extension, which is the point that has the minimum difference from e over a one-pixel wide slit. If the minimal error is less than a threshold, and the edge directions at e and p are similar (by examining the inner product of the gradient directions at e and p), e is considered continuous if p = e, or discontinuous if p ≠ e. Otherwise, it is labeled as “other”.

Prediction of Edge Misalignment In Fig. 2, the discontinuity between e and p may be caused by the error in the position of either e or p or both. If the position p at the adjacent block is correct, then e should be moved to p', the extension of p to this block. Therefore, the misalignment should be the projection of p' - e onto g. When applying this prediction algorithm in a coder, which processes the blocks in a raster order, we assume all the neighboring blocks on the top and left of a current block have already been corrected. For any edge in the top or left boundary of the current block, if it is found to be discontinuous with its neighboring pixels, we use the above method to predict its misalignment. For an edge on the right or bottom boundary, we first use the above method to find its misalignment with its corresponding point in the right or bottom blocks. But because we do not know which one is more correct, the best we can do is to assume that they meet in the middle. Therefore, the predicted misalignment is half of the projected value.

4 Coding Scheme and Preliminary Results

We have incorporated the above misalignment estimation and correction scheme in a block-based coder. For each 16 × 16 block where the initial motion compensation error exceeds a threshold, we perform the following steps: i) Detect and trace edges in this block; ii) Estimate the edge misalignment function for each edge; iii) Quantize and code the misalignment function for each edge; iv) Obtain a new prediction block based on quantized edge misalignment functions; v) Code the new error block using DCT and run-length coding.

Coding of Misalignment Function As shown previously, for a straight edge that undergoes a rotation, the misalignment function along this edge can be approximated well by a linear function. We have also observed, when the estimated misalignment function cannot be approximated well by a linear function, it often corresponds to cases where misalignment correction leads to insignificant error reduction. In addition, to maintain the edge continuity across block boundaries, it is important to specify precisely the misalignment values at block boundaries. Based on these considerations, we code the misalignment values at the two end points of an edge, and linearly interpolate the intermediate values. The end points that are on block boundaries are further subjected to prediction based on detected discontinuities along the block boundaries (see Sec. 3). That is, for end points that are discontinuous across the boundary, we predict the misalignment using the method of Sec. 3, and code the prediction error. For other cases (continuous, others, not-on-border), the predicted misalignment is zero.

The misalignment function along an edge is coded only if the error reduction after misalignment correction exceeds a threshold. To optimize the rate-distortion (RD) performance, ideally, the threshold should be the product of the bit rate required for coding the misalignment function along this edge and the slope of RD curve of the conventional DCT coder at this rate. For simplicity, we use a fixed threshold which is the product of the average rate for coding a misalignment function and the RD slope of the conventional DCT coder at the lowest rate possible (with QP=31). The side information about whether an edge is subjected to misalignment correction is coded based on the detected state of edge discontinuity (continuous, discontinuous, others, or not-on-border), using the conditional probability that an edge suffers from misalignment under different states.

Figure 3 shows the RD curves of the proposed coder and the conventional coder for the CIF sequences “football” (4 frames) and “foreman” (15 frames). For both coders, the bit rates are estimated based on entropies of the variables to be coded, and only count the bits
for coding the residuals after motion compensation using block motion vectors. Only the luminance component is considered. The solid line represents the error reduction by coding the misalignment function. The dash-line is obtained by DCT coding the residual after misalignment correction using different QPs. The dotted line is the RD curve obtained by applying DCT coding to the motion compensation residual directly. The rate and distortion are measured only over blocks where misalignment correction has been applied. We can see that for these blocks, the proposed method can achieve a significant gain (0.5 to 1 dB). When the rate and distortion are averaged over all blocks, the coding gain by the proposed method is insignificant (less than 0.25 dB for “foreman” and almost zero for “football”). This is because, in both “football” and “foreman”, there are many blocks where errors are due to uncovered regions and changes in reflection and shadowing patterns on the moving objects. Misalignment correction was applied to less than 15% and 6% of the total number of blocks, in these two sequences. Note however, that even when the overall PSNR is similar, the coded image with misalignment correction has less blocking and ringing artifacts. The decoded images are not shown here because of page limitation.

5 Discussion

Edge-misalignment is only one source of error in motion compensation. Another significant source is the changes in reflection and shadow patterns due to object motions and/or lighting conditions, and the changes due to uncovered regions. The errors so generated usually are smoothly varying and can be modeled well by the MRF and consequently be coded efficiently using DCT. Yet, there are still errors that do not fit either of the above two categories. These are usually caused by restrictions in motion estimation (e.g. insufficient search range) that lead to matching of blocks which are structurally dissimilar but with similar color ranges. In order to code motion compensation residuals efficiently, one should separate errors with different characteristics and use the most appropriate representation for each type. The study here is a first step towards this direction. To improve the overall coding efficiency, more efficient schemes for coding errors that cannot be compensated well by misalignment correction need to be developed.

The coder presented here first corrects misalignment and then codes the remaining error. Another approach to handle errors which are clustered around edges is by using a transform basis well matched with such errors. The matching pursuit approach of [2] can be considered in this category, which uses a set of separable Gabor functions (essentially horizontal and vertical blobs with different length and width) as the transform basis, and finds the best set of basis functions to use for any given block.

References


Figure 4: An example from "football": (a) The current frame; (b) The previous frame; (c) The predicted current frame using half-pel accuracy block motion vectors; (d) The absolute error image between (a) and (c); (e) The new predicted image after correcting edge-misalignment; (f) The new error image.

Figure 5: A typical motion compensation error block: (a) the original block; (b) the predicted block based on the block MV; (c) the error between (a) and (b); (d) the edge map of the predicted block; (e) weight function for the second edge; (f) the predicted block after misalignment correction; (g) the new error block;