Low Bit-Rate Video Coding Using Iterative Affine Motion Estimation and Quadtree Segmentation [1]

O. Reshef and D. Malah

Department of Electrical Engineering
Technion - Israel Institute for Technology
Technion city, Haifa 32000, Israel
Email: malah@ee.technion.ac.il

Abstract

In this work we present a low bit-rate hybrid coder which achieves a significant rate reduction, as compared to the standard H.261 coder, through adaptive motion segmentation, improved motion compensation, and efficient representation of motion-model parameters. In the proposed coder, each frame is adaptively segmented into non-fixed-size blocks using a *quadtree* approach by which a block is split if the estimated motion parameters for that block do not provide adequate motion compensation. Motion estimation is performed by an Iterative Least Squares algorithm using an *affine* motion model for large blocks and the common *translation* motion model for the minimum size blocks. An efficient representation of the motion-model parameters is obtained via *'corner vectors'* which are shown to provide reduced sensitivity to quantization errors. Simulation results show a particular advantage of the proposed coder at bit rates of 64Kbps and below.

1 Introduction

Standard video coders like H.261 and MPEG [1] exploit temporal, spatial, and statistical redundancies present in image-sequences in order to obtain effective compression. One of the key strategies for achieving low bit rates is to improve the prediction of the current frame from the previous frame without causing a sizable increase in side information. Standard coders use a fixed segmentation into blocks of identical size and estimate translation motion (2 displacement parameters) for each block using block matching [2].

To improve the performance of those coders, recent works [3, 4, 5] applied *quadtree* motion segmentation. A given block is split when its prediction fails to match some quality criterion. This way, the amount of side information is reduced. Other reported works [6, 7, 8] use higher order motion models to improve the prediction but keep using a fixed motion segmentation.

In this work we propose a low bit-rate coder which improves the performance of the standard H.261 coder by combining adaptive motion segmentation, improved motion estimation, and efficient representation of motion information. An affine motion model is used to improve the prediction over the simple translation model. The increased model order increases the motion information per block. Hence, to reduce the total coded motion information in each frame, we perform a quadtree motion segmentation, allowing large blocks in regions with homogeneous motion. To obtain improved motion compensation it is important to have a good estimate of the motion parameters. To this effect we apply an iterative least-squares (ILS) estimation technique, which provides improved compensation even for the simple translation model (which is used for the minimum-size block of 16x16). For efficient coding of the affine motion parameters, we propose to represent these parameters via the resulting displacements of 3 corners of the block. This representation is shown to reduce the sensitivity of the motion information to quantization and helps in reducing the coding rate of the side information.

[1] This work was supported by the Samuel Neaman Institute for Advanced Studies in Science and Technology of the Technion.

2 Iterative least-squares algorithm for motion estimation

The Iterative Least Squares algorithm (ILS) is an iterative algorithm which has been applied for estimating the parameters of motion models which are more complex than just translation motion [9, 10, 11]. The basic assumptions are: (i) All the pixels in a region undergo the same motion according to a prescribed motion model. (ii) The changes in pixel values between two successive frame are only due to motion.

Below we provide a concise derivation of the algorithm for a *general* motion model T. Ignoring for the moment the problem of motion segmentation, the motion of any region B in the current frame $I(t)$, in relation to the previous frame $I(t-1)$ can be described by:

$$\forall \vec{r} \in B : I(\vec{r}, t) = I((T^{\Delta k} \circ T^{k-1})\vec{r}, t-1), \tag{1}$$

where $\vec{r}$ denotes the coordinates of a pixel, T^{k-1} is the coordinate mapping according to the motion that was estimated at iteration $k-1$, and $T^{\Delta k}$ is the coordinate mapping *refinement* calculated at iteration k. '$\circ$' denotes composition of coordinate mappings. According to T^{k-1}, the pixel $\vec{r}$ in the current frame has its origin in a pixel located at $\vec{r'}$ in the previous frame, where $\vec{r'} = T^{k-1}\vec{r}$. Using this notation, (1) becomes:

$$\forall \vec{r} \in B : I(\vec{r}, t) = I(T^{\Delta k}\vec{r'}, t-1). \tag{2}$$

Ideally, as stated in (2), the refinement of the motion parameters, $T^{\Delta k}$, should satisfy this equation for all the pixels in B. Formally adding and subtracting $\vec{r'}$ from the argument of the right hand side of (2), and then expanding it into a Taylor series about $\vec{r'}$, assuming that the motion is small, and retaining terms only up to first order, we get:

$$\forall \vec{r} \in B : I(\vec{r}, t) = I(\vec{r'}, t-1) + r''_x I_x(\vec{r'}, t-1) + r''_y I_y(\vec{r'}, t-1), \tag{3}$$

where r''_x and r''_y are respectively the x and y coordinates of $\vec{r''} = T^{\Delta k}\vec{r'} - \vec{r'}$ and

$$I_x(\vec{r'}, t-1) = \left.\frac{\partial I}{\partial x}\right|_{\vec{r'}, t-1}, \quad I_y(\vec{r'}, t-1) = \left.\frac{\partial I}{\partial y}\right|_{\vec{r'}, t-1}.$$

This way, the mapping refinement $T^{\Delta k}$ is not an argument in (3). Now the following least squares (LS) cost function (the squared-prediction-error) is defined:

$$\varepsilon^2 = \sum_{\vec{r} \in B} \left(I(\vec{r}, t) - P(\vec{r}, t)\right)^2. \tag{4}$$

where $P(t)$ is the prediction of $I(t)$, given by the right hand side of (3). We assume that $T^{\Delta k}$ is defined by L parameters $\{z_i\}_{i=1}^{L}$. The parameters which minimize ε^2 in (4) can be found by solving the following system of equations:

$$\frac{\partial \varepsilon^2}{\partial z_i} = 0, \quad i = 1, \ldots, L. \tag{5}$$

By choosing a suitable mapping we can obtain a system of *linear* equations , while applying the LS criterion directly to (2) would result in a *non-linear* set of equations.

The result of iteration k is a coordinate mapping, $T^{\Delta k}$, which is used to refine the previous mapping, T^{k-1}, via $T^k = T^{\Delta k} \circ T^{k-1}$. The initial mapping, T^0, is calculated by the block matching algorithm [2], with the intention to reduce the difference between the motion compensated prediction block and the original block, such that the approximation in (3) is justified.

2.1 Translation motion estimation with the ILS algorithm

In order to clarify the general description of the ILS algorithm presented above, we will derive now the explicit ILS algorithm for estimating *translation* motion parameters.

The parameters (a, d) of a translation motion of a region B, induce a coordinate mapping T_d, that is applied to a coordinate vector $(x, y)^t$ in the following way:

$$T_d(x, y)^t = (x + a, y + d)^t, \tag{6}$$

where superscript t denotes 'transpose'.
By substituting (6) in (1) we can explicitly write the model for translation motion of region B as

$$\forall (x,y) \in B \ : \ I(x,y,t) = I(x + a^{k-1} + a^{\Delta k}, y + d^{k-1} + d^{\Delta k}, t-1), \tag{7}$$

where (a^{k-1}, d^{k-1}) are the motion parameters calculated in iteration $k-1$ and $(a^{\Delta k}, d^{\Delta k})$ are the parameters of the refinement calculated in iteration k. First order expansion of the right hand side of (7) about $(x + a^{k-1}, y + d^{k-1})$ gives:

$$\forall (x,y) \in B \ : \ I(x,y,t) = I(x + a^{k-1}, y + d^{k-1}, t-1) + a^{\Delta k} I_x + d^{\Delta k} I_y, \tag{8}$$

where, to simplify notation, $I_x \equiv I_x(x + a^{k-1}, y + d^{k-1}, t-1)$, $I_y \equiv I_y(x + a^{k-1}, y + d^{k-1}, t-1)$. Using the LS cost function defined in (4), and applying (5) with $z_1 \equiv a^{\Delta k}$ and $z_2 \equiv a^{\Delta k}$ gives a system of linear equations whose solution provides the refining parameters for iteration k:

$$\begin{pmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{pmatrix} \begin{pmatrix} a^{\Delta k} \\ d^{\Delta k} \end{pmatrix} = \begin{pmatrix} \sum I_x I_d \\ \sum I_y I_d \end{pmatrix}, \tag{9}$$

where, for simplification, $I_d \equiv I(x,y,t) - I(x + a^{k-1}, y + d^{k-1}, t-1)$ and all the summations are done over all the pixels in B. The final result of iteration k is the motion parameters that refine the motion parameters calculated in iteration $k-1$: $a^k = a^{k-1} + a^{\Delta k}$, $d^k = d^{k-1} + d^{\Delta k}$. The initial motion parameters, (a^0, d^0), are estimated by block matching.

2.2 Affine motion estimation with the ILS algorithm

As explained earlier, the ILS algorithm can be used for estimating the parameters of various types of motion models. One particular useful model is the *affine* motion model that is characterized by 6 parameters as shown below. Moon and Kim [11] evaluated various motion models and concluded that the affine motion model gives better results, in terms of estimation accuracy, convergence and noise sensitivity, than a higher order model.
The affine motion parameters induce an affine coordinate mapping $T_{\mathbf{A},\mathbf{b}}$ which is defined by the 2×2 matrix $\mathbf{A}$ and the two elements vector $\mathbf{b}$:

$$T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{b} = \begin{bmatrix} b & c \\ e & f \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ d \end{bmatrix} = \begin{bmatrix} a + bx + cy \\ d + ex + fy \end{bmatrix}. \tag{10}$$

The affine motion model can describe *translation*, *rotation*, *shear* and *scaling* or their combination. It can describe a mapping of a square to any parallelogram or a mapping of a triangle to any triangle. A composition of two affine mapping is an affine mapping itself. At each least-squares iteration, the motion refinement parameters are calculated by solving the following system of 6 linear equations, where a notation similar to that in section 2.1 is used.

$$\begin{bmatrix} \sum I_x^2 & \sum x I_x^2 & \sum y I_x^2 & \sum I_x I_y & \sum x I_x I_y & \sum y I_x I_y \\ \sum x I_x^2 & \sum x^2 I_x^2 & \sum xy I_x^2 & \sum x I_x I_y & \sum x^2 I_x I_y & \sum xy I_x I_y \\ \sum y I_x^2 & \sum xy I_x^2 & \sum y^2 I_x^2 & \sum y I_x I_y & \sum xy I_x I_y & \sum y^2 I_x I_y \\ \sum I_x I_y & \sum x I_x I_y & \sum y I_x I_y & \sum I_y^2 & \sum x I_y^2 & \sum y I_y^2 \\ \sum x I_x I_y & \sum x^2 I_x I_y & \sum xy I_x I_y & \sum x I_y^2 & \sum x^2 I_y^2 & \sum xy I_y^2 \\ \sum y I_x I_y & \sum xy I_x I_y & \sum y^2 I_x I_y & \sum y I_y^2 & \sum xy I_y^2 & \sum y^2 I_y^2 \end{bmatrix} \begin{bmatrix} a^{\Delta k} \\ b^{\Delta k} \\ c^{\Delta k} \\ d^{\Delta k} \\ e^{\Delta k} \\ f^{\Delta k} \end{bmatrix} = \begin{bmatrix} \sum I_x I_d \\ \sum x I_x I_d \\ \sum y I_x I_d \\ \sum I_y I_d \\ \sum x I_y I_d \\ \sum y I_y I_d \end{bmatrix} \tag{11}$$

3 Representation of motion model parameters via corner vectors

In order to encode the real valued parameters of the affine motion model they must be quantized. When we use the translation motion model of a region, the two parameters (a, d) form a *motion vector* whose components have a dimension of displacement (in pixels). Quantizing a and d is naturally done by choosing a quantization step according to a desired sub-pixel resolution. The affine mapping of a region is describe by

the 6 parameters (a, b, c, d, e, f) as in (10). Only a and d have a dimension of pixel displacement. Therefore, we have to choose a quantization step for each parameter that matches its type, the region size and the region location. We simplify this problem here by using the concept of *corner vectors*. Instead of describing the mapping with 6 parameters, each basically having a different physical effect, we use 3 corresponding motion vectors. The regions in the current frame are selected to have shape of a rectangular block and the corner vectors are the motion vectors of 3 corners of the region. The corner vectors were used in other works for the actual estimation of the motion parameters (e.g., [7]). We use them here only for the purpose of representing the affine motion parameters in order to reduce their sensitivity to quantization errors, as explained below.

The following relations exist between the corners of a block B and the corners of the parallelogram B' resulting from application of the affine mapping (see Fig. 1):

$$(x'_i, y'_i) = (a + bx_i + cy_i, d + ex_i + fy_i) \ , \ i = 1, 2, 3. \tag{12}$$

The corner vectors, $\{\vec{d}_i\}_{i=1}^3$, are then given by: $\vec{d}_i = (x_i - x'_i, y_i - y'_i) \ , \ i = 1, 2, 3$. The corner vectors describe, by definition, translation motion and thus have the dimension of pixel displacement. Recall that each pixel in the current frame has a corresponding pixel in the previous frame that is used as its prediction. It can be shown (not shown here because of lack of space) that if the affine mapping parameters, (a, b, c, d, e, f), are quantized, the induced quantization error in the location of the prediction pixel depends on the *absolute* location of the pixel in the current frame, i.e., its location relative to the origin of the *frame*. However, using the corner vector representation, $\{\vec{d}_i\}_{i=1}^3$, and quantizing their components, gives rise to a location error that depends only on the *relative* position of the pixel in the block; i.e., its position relative to the origin of the *block*.

4 Quadtree motion segmentation

Using affine motion estimation gives better performance (in terms of bit-rate needed for encoding the *difference* frames) than translation motion, but increases the amount of side information. For sequences which require relatively high bit rates for encoding the difference frames with reasonable quality, the reduction in bit-rate due to better motion compensation by using the affine motion model is greater than the added overhead in side information. However, for low bit-rate sequences, like typical videophone sequences, the increase in the side information is too large. We have, therefore, to reduce the amount of side information for such sequences, yet keep the quality of the prediction as high as possible. We can solve these contradicting requirements by employing a *quadtree* motion segmentation approach.

The quadtree is a recursive data structure for describing homogeneous block regions. In this work the quadtree is used for segmenting the current frame into coherently moving regions. The segmentation is built in conjunction with the motion estimation using the following algorithm:

1. Initialization: The current frame is divided into several large blocks. The motion parameters of each block are estimated and the corresponding prediction block is built. The maximum allowed number of corner vectors, N_v, is set.

2. Selection: The block with the largest Squared-Prediction-Error (SPE), as defined in (4), is selected as a candidate for splitting.

3. Test: The candidate block is split into four sub-blocks and the motion parameters of each are estimated. The smallest allowed block size is 16×16. The affine motion model is used for blocks larger than the minimum size and a translation motion model is used for the minimum size blocks. The prediction of each of the four sub-blocks is constructed and the sum of the SPEs of the four sub-blocks is compared to the SPE obtained by using the entire block, calculated above. The split is retained if the SPE is reduced.

4. Loop: The algorithm repeats steps 2 and 3 until the total number of corner vectors obtained reaches the given maximum of N_v vectors. A block whose motion is described by an affine model needs 3 corner vectors while the smallest size block, which uses a translation model, needs only one vector.

5 Simulation results

Table 1 shows simulation results in coding the first 100 frames (*luminance* only) of the test sequence "Miss America' (CIF format). The average rates given exclude the first frame (intra-coded). The quantization step was kept constant to keep (as found in simulations) a nearly constant visual quality. The proposed algorithm, which combines quadtree motion segmentation, ILS motion estimation, and corner-vector representation, was incorporated in a H.261-type encoder (labeled 'quadtree+ILS'). Two iterations (following the initial block matching) were found to be sufficient. The maximum number of corner vectors was set to $N_v = 100$ and the corner vectors were quantized with a step of 1/4 pixel. Adaptive arithmetic coding was used to encode the corner-vector data. For comparison purposes, a modified H.261 encoder using an enhanced (half-pixel resolution, as in MPEG) block-matching, with blocks of 16×16 pixels, was implemented too (labeled 'fixed+bmhs'). It is seen from Table 1 that the proposed coder ('quadtree+ILS') achieves more than a 3:1 rate reduction as compared to both the standard and the modified H.261 coders. Fig. 2 shows the total number of bits for each frame in the coded sequence, by each of the three examined coders. Fig. 3 provides a demonstration of processed results for one of the frames (# 43) of "Miss America" sequence. The reduction in prediction error achieved by the proposed coder is clearly seen from the figure.

encoder	quantization step	average # of bits/frame	average PSNR in dB
standard (H.261)	10	5413	37.21
modified ('fixed+bmhs')	10	4736	37.58
proposed ('quadtree+ILS')	11	1493	37.42

Table 1: Simulation results obtained for 100 frames of "Miss America" sequence (CIF format, Luminance)

6 Conclusion

The proposed hybrid coder combines improved motion estimation (via ILS and an affine motion model), adaptive motion segmentation (via a quadtree) and an efficient corner-vector representation. Compared to the standard H.261 it is found to achieve more than a 3:1 rate reduction at and below 64Kbps, for typical videophone sequences (e.g., "Miss America" sequence) with similar quality. Further work is needed to reduce its computational complexity which at present could run up to 5 times the complexity of the standard coder.

References

[1] C-T. Chen. Video compression: Standards and applications. *Journal of Visual Communication and Image Representation*, 4(2):103–111, June 1993.

[2] J.R. Jain and A.K. Jain. Displacement measurement and its applications in interframe coding. *IEEE Transactions on Communication*, COM-29(12):1799–1808, 1981.

[3] M.H. Chan, Y.B. Yu, and A.G. Constantinides. Variable size block matching motion compensation with applications to video coding. *IEE Proceedings*, 137, Pt I(4):205–212, August 1990.

[4] F. Dufaux, I. Moccagatt, B. Rouchouze, T. Ebrahimi, and M. Kunt. Motion-compensated generic coding of video based on a multiresolution data structure. *Optical Engineering*, 32(7):1559–1570, July 1993.

[5] J.W. Kim and S.U. Lee. Hierarchical variable block size motion estimation technique for motion sequence coding. *Optical Engineering*, 33(8):2553–2561, August 1994.

[6] C-S. Fuh and P. Maragos. Affine models for image matching and motion detection. *ICASSP'91*, pp. 2409–2412, Toronto, Canada.

[7] V. Seferidis and M. Ghanbari. General approach to block-matching motion estimation. *Optical Engineering*, 32(7):1464–1474, July 1993.

[8] H. Sanson. Motion affine models identification and application to television image coding. *SPIE*, 1605, Visual Communications and Image Processing: Visual Communications:570–581, 1991.

[9] M. Hotter and R. Thoma. Image segmentation based on object oriented mapping parameter estimation. *Signal Processing*, 15:315–334, 1988.

[10] S.F. Wu and J. Kittler. A differential method for simultanous estimation of rotation, change of scale and translation. *Signal Processing: Image Communication*, (2):69–80, 1990.

[11] J-H. Moon and J-K. Kim. On the accuracy and convergence of 2-D motion models using minimum MSE motion estimation. *Siganl Processing; Image Communication*, (6):319–333, 1994.

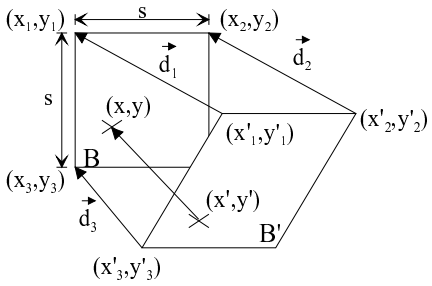

Figure 1: Corner-vector representation

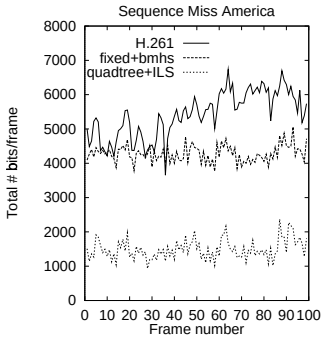

Figure 2: Frame-rate vs. frame no.

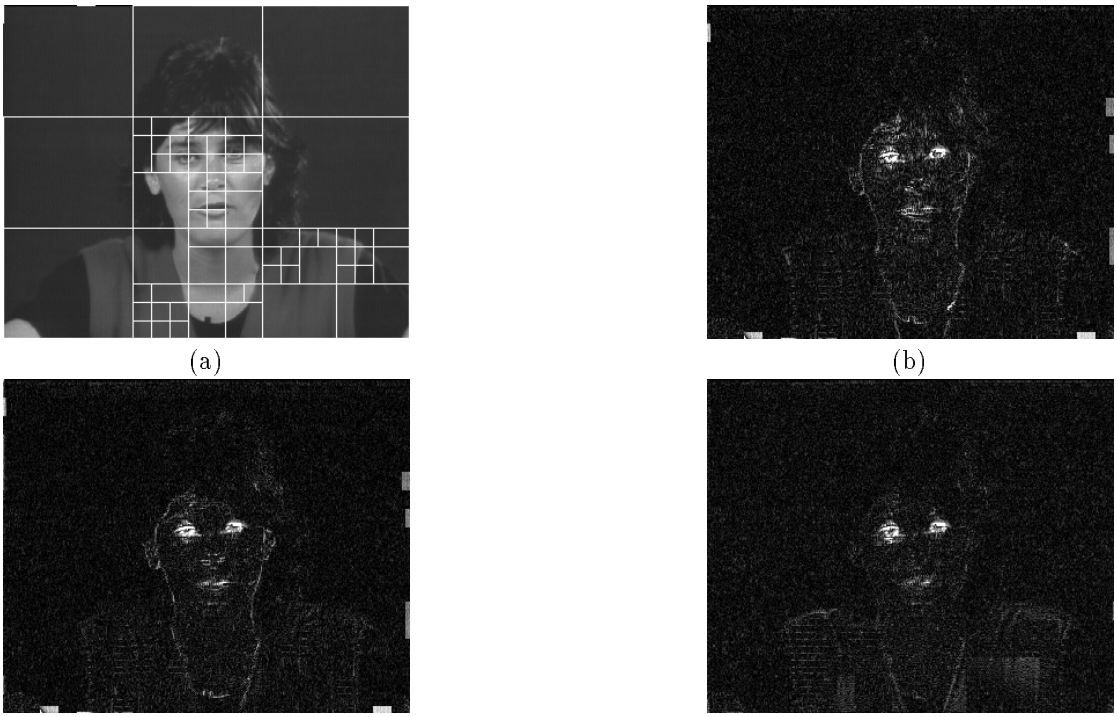

Figure 3: Processing results for frame #43 of "Miss America" sequence: (a) Segmentation ('quadtree+ILS'), and Prediction-error for (b) H.261, (c) Modified H.261 ('bmhs'), (d) Proposed ('quadtree+ILS').