Inter-Frame Coding using Motion-Adaptive Sliced Blocks and Elastic Motion Model Techniques: A Survey

[1]Eskinder Anteneh Ayele and [2]S B Dhok

[1,2]Department of Electronics & Communication Engineering (Center for VLSI & Nanotechnology), Visvesvaraya National Institute of Technology, Nagpur-440010, India.
E-mail: [1]eskinderanteneh@yahoo.co.uk [2]sbdhok@ece.vnit.ac.in

***Abstract*— In this paper, we present an experimental survey on transformed block-based motion compensation using higher order elastic motion models and motion/geometry adaptive block partitioning which are advanced motion compensation techniques that are good candidates for incorporation into next generations of video coders. However, it is very important that these methods should keep reasonable complexity burden. The paper presents an efficient motion compensation scheme that integrates both geometry/motion-adaptive partitioning and transformed block-based motion model (elastic motion model) as additional room to the standard motion estimation and compensation procedure. From our review and simulation results, making use of geometric partitioning incorporation with a non-rigid image registration or elastic motion model can give a better performance.**

***Keywords*—Motion compensation, Video coder, Elastic motion model, Adaptive block partitioning.**

1. INTRODUCTION

Inter-frame coding explores temporal redundancy between frames to save coding bits. By using motion compensated prediction, the best matching position of the current block is found within the reference picture so that only prediction difference needs to be coded [1]. However, the findings of true motion do not always translate to the finest prediction. So, the objective is to find the best prediction that optimizes a rate-distortion trade-off. Motion compensation (MC) through block matching algorithm (BMA) has established itself as the most successful in the current production of video codec standards, exceeding all other candidates as stated in [2]. Mostly the implementation of BMA depends on two factors: the motion model and the block size and/or shape. In video sequences, motion occurs mainly for two reasons. The first is global motion (camera motion) and the second reason is due to the intrinsic motion of the object(s) in the scene. Hence, independent moving objects in combination with camera motion lead to a complicated motion vector field [3]. In addition, the computational complexity of block-based motion estimation is the direct consequence of the expensive 2D block-matching process [4]. Due to its simplicity and effectiveness, a translational motion model technique has been the most apparent choice so far. Yet, this model has a number of indispensable shortcomings which should be addressed.

The predominant limitation is the blockwise-constant motion model where all the pixels in a typical block are allotted the same motion vector (MV). Second, BMA is not capable of detecting camera zoom, rotation and complex motions like deformation of objects. The complex motion in blocks can be approximated using a piecewise narrow translational model with smaller sub-blocks. However, the restriction here is that blocks smaller than 4×4 are impractical from a rate distortion optimization point of view. In the past two decades under the block matching framework, quite a lot of strategies have been investigated for efficient prediction of complex MV fields [5]-[11]. These research efforts including techniques such as; generalized block-matching [6] and transformed block-based motion compensation [8], basically replace the conventional translational motion model with more dynamic and higher-order motion models using affine [7], perspective [6], bilinear [8] or polynomial [5] transformation functions. Recently, an elastic motion model [11] with larger blocks (also known in the literature [12] as super macro-blocks) has proven to be the most efficient in the context of the state-of-the-art H.264 standard [13]. This new motion model proposed in [11] has outperformed existing approaches including popular motion models such as affine, as a result of its capability to precisely capture complex motion fields with a small number of motion parameters even when the block is much larger than conventional ones [3].

In earlier video coding standards for example H.261 [14], fixed square blocks of size 16×16 were implemented. Later in H.263 [15], each block was possibly permissible to be split into 8×8 square sub-blocks whenever it leads to an improved rate-distortion performance. This choice was influenced by the better performance of variable size block matching presented in [16] and [17] or locally adaptive multigrid block matching [2] over the fixed ones in H.261. The most recent standards, in particular, H.264 and HEVC, have adopted an advanced tree and quad-tree structured block partitioning

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

methods respectively. Though the sub-blocks are limited to be of fixed shapes and sizes, the combination of squares and rectangles provide a good piecewise approximation of complex motion fields in video frames [3].

Even though square and symmetric partition modes are well known since they exist in H.264/AVC, the asymmetric motion partitioning modes are included and introduced in [18] for the HEVC standard design to improve the coding efficiency for the irregular object boundaries. Specifically, by adding 4 possible partition modes for inter prediction mode, efficient representation of irregular patterns using flexible motion partitions can be realized.

However, in video sequences, it is expected that object(s) or parts of an object undergo autonomous motion. Hence, it is logical to assume that macro-block segmentation is influenced by this phenomenon. In other words, the partitioning should represent a major motion field discontinuity in a block. Being unable to take advantage of this, the quad-tree segmentation of fixed symmetric or asymmetric often causes additional residual data to be coded in the region of boundaries of objects or wherever motion field discontinuities occur.

Motion-adaptive sliced blocks [19]-[24] try to address this problem by dividing blocks using an arbitrary line slice for independent motion compensation. Hence, the new subblocks which have more homogeneous motions be able to give an enhanced prediction and increasing rate-distortion output as compared to pre-defined squares and rectangles. However, the judgment of the best line segment remains a cumbersome task. A full search technique to locate the best partition has been proposed in [19]. In [3], the line segment is established using any two pixels in a block boundary. However, the technique increases the computations since the number of possible partitions is very large and resultant sub-blocks are needed to be motion compensated for every likely partition. In order to minimize the computational complexity, a fast wedge based motion compensation technique that decides the best partition from a total candidate partitions (77 iterations per block) is proposed [20]. This gives a significant computation reduction. However, compared to full search presented in [19], it is still quite costly and time consuming.

This paper has dual purposes. First, it gives the survey/review of the ways of determining the best geometry-adaptive partition with a reduced amount of computational burden presented in [3]. Where, it proposes a quick technique whereby the near-optimal partitions can be found using only some 16 iterations per block as compared to [20]. The partition has been determined by detecting the dominant texture edge in a block and then applying a chronological two-step search for the radius and angle of the line segment. It is shown that the partition obtained by [3] had no significant loss in compression efficiency when compared with the more computationally expensive full search. Second, it presents the investigations how geometry information provides additional compression gain when used with higher-order motion models such as the elastic model proposed in [11] through experimental simulations. Both geometry/motion-adaptive block partitioning and the elastic motion model try to achieve a common objective as a goal– better prediction with fewer bits, with different ways by exploiting the characteristics of video sequences [3].

A remarkable part of this review paper is to investigate and give verification for a unique way of improving the accuracy of the position of the line segment presented in [3] when compared with prior proposed techniques [22]-[24]. The enhanced fractional pixel accuracy together with a customized coding mechanism gives better results which will be revealed subsequently. This paper presents more in-depth review, analysis, and conducting experimental tests to verify the efficiency of fast motion adaptive sliced blocks and its potential relevance when jointly used with higher-order elastic motion models.

The remains of the paper are structured in six sections as follows. Section 2 presents a short review of motion estimation techniques so as to remind the reader. Section 3 describes the fast geometry adaptive block partitioning scheme and complexity-reduced geometry search techniques. The Geometry-assisted elastic motion model and motion compensation methods are explained in section 4. Section 5 explains the experimental results. Finally, we conclude the paper in section 6.

2. Short Review of Motion Estimation Techniques

A number of various motion estimation (ME) algorithm techniques have been proposed in the literature. Detailed reviews are given in [25]-[28]. They also address the problem in an image sequence analysis perspective. As stated before, the purpose of motion estimation in the field of coding is significantly different. More specifically, the determination of the motion is not an intrinsic goal; indeed motion estimation techniques aim at minimizing the bandwidth corresponding both to the prediction error information and to the motion overhead. Therefore, in the remainder of this section, some of the classical motion estimation algorithms are discussed and compared from an image sequence coding point of view. Motion estimation techniques can be divided into four main groups:

- Gradient techniques [2],
- Pixel recursive techniques [29],
- Block matching techniques [2],
- Frequency-domain techniques [30].

Gradient techniques have been developed for video sequence analysis purposes. They solve the optical flow and results in a dense motion field. Both pel-recursive and block matching techniques have been developed in the framework of image sequence coding. Pixel recursive techniques can be regard as a subset of gradient techniques. However as they constitute an important contribution in the field of coding, we consider them as a separate group. Block matching techniques are based on the minimization of a disparity measure. They are the most widely used in coding applications. Finally, frequency-domain techniques are derived from the relationship between transformed coefficients (e.g., Fourier transform) of shifted frames.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

However, they lack a widespread use, especially in the field of image sequence coding.

3. FAST GEOMETRY/ MOTION-ADAPTIVE PARTITIONING

From the characteristic of video sequences, the combination of camera motion and the motion due to objects in the scene creates intricate MV field even at a macro-block level [3]. In a traditional block-based video coding (H.261 and H.263), a single MV is assigned to a block. When there is different type of motion present in a block, variable or tree-structured block-matching attempts to slice that block into smaller sub-blocks (eg. H.264 and HEVC) [1]. So, the inherent goal in [3] is to find blocks and/or sub-blocks that have more than one type of independent motion. If there is more variety of motion present in a block, it is very liable that those individual parts of a block be the property of separate entities. Yet, the consequential sub-blocks always do not reflect this separation and instead they model a coarse estimate of the associated motion field. Therefore, if a block is considered necessary to be segmented anchored in a motion field discontinuity, the assumption of fixed squares or rectangle blocks has to be predetermined as shown in Figure1. This adjustment will facilitate further gain in inter-frame coding. Thus, motion/geometry adaptive block partitioning is a realistic evolution in video compression research.

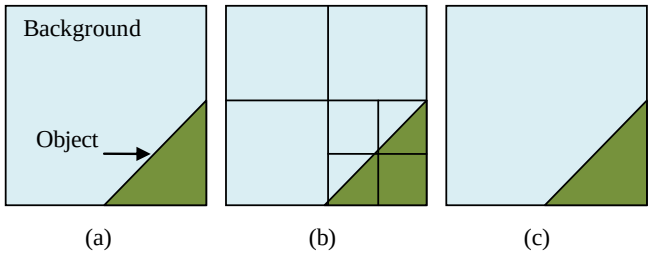


Fig. 1. A demonstration of geometry-adaptive partitioning. *(a)* A typical video frame with background and a foreground object, (b) Possible tree-structured partitioning using H.264, and (c) A possible partition using the proposed method [3].

3.1 Slice Line Selection

The slice line is determined by a rate-distortion optimization technique considering the amount of bits for motion compensation residuals and motion vectors [19]. Considering the slice line determination method for macroblock division, the set of line segments which divide a macroblock into two sliced blocks is defined as: $\Lambda=\{l_i;\ 1 \le i \le L\}$. When selecting the slice line, the line segments l_i belonging to Λ are selected in turn and the macroblock is divided into two regions using l_i. Then, for each region (where the region number $k \in \{1, 2\}$), the motion vector $m_k(l_i)$ is estimated for a reference picture using (1) and (2).

$$J_k(m,l_i) = D_v\left(s_k, r_k(m)/l_i\right) + \lambda_v R(m-p) \quad (1)$$

$$m_k(l_i) = \underset{m}{\operatorname{argmin}}\ J_k(m,l_i) \quad (2)$$

Where, the function $J_k(m, l_i)$ denotes the cost function for the region k when the macroblock is divided by the line segment l_i and the motion vector for the region k is m. The function $D_v(\cdot)$ denotes the sum of the error between the region to be coded in the original image s_k and the corresponding region of the reference image $r_k(m)$, when referring to with the motion vector m. p is the prediction value for the motion vector and the function $R(\cdot)$ denotes the amount of bits for the difference in motion vectors $m-p$. The motion vector prediction value p is obtained as the median value of the motion vectors used in the preceding encoded blocks neighboring the block being encoded. λ_v is the Lagrangian multiplier in the motion estimation stage [31].

Next, an evaluation value is calculated for the whole macroblock using the motion vectors $m_k(l_i)$ calculated for two regions and the line segment l which gives the lowest evaluation value is selected as the slice line using (3).

$$l = \underset{l_i \in \Lambda}{\operatorname{argmin}} \sum_k J_k\left(m_k(l_i), l_i\right) \quad (3)$$

3.2 Fast and Complexity-Reduced Geometry Partition Search

Even though motion-adaptive block partitioning is a reasonable innovation, the main challenge still remains to find out the optimal partition or line segment for slicing a block. Let a line segment l divide a block I into sub-blocks I_i for i = 1, 2. A line segment can be characterized by the gradient or slope information m and y-intercept or the x and y-intercept. It can also be represented using its intersection angle with y-axis and its distance from the coordinate system [3][21]. Using this illustration, the equation for the line segment, $l(r,\theta)$ can be given as [3]

$$x\cos\theta + y\sin\theta = r \quad (4)$$

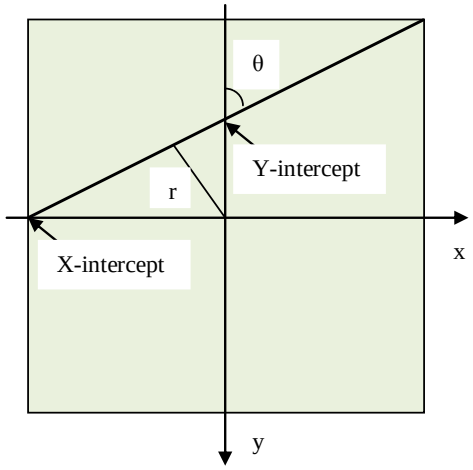


Fig. 2. Partition line segment within a block [3].

Where, r is the radius or distance from the block centre and θ is the angle with respect to the y-axis as shown in Figure 2.

In a rate-distortion optimized video coder, the selection of line segment is performed by choosing the line that minimizes a Lagrangian cost function $J(l)$ given by (1).

Preferably, the line segment should lie exactly on the background and foreground boundary or at the boundary of different objects or elements of the same object assuming they

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

move separately of each other. However, due to the limitations of optical sensors and lighting effects etc, the motion border often does not lie on the object boundary and hence it is difficult to recognize [3]. Nevertheless, we can initialize it to be on the leading edge of a block as presented in [20]. To achieve this, edge detection is useful on the block. Thus, Sobel edge detection produces very satisfactory results in detecting the dominant edge in a block [32]. Next, the Standard Hough Transform (SHT) is applied to find the line parameters of the detected edges. Subsequently, peak values are found in the parameter space and possible lines are rating in order. We use the top-ranked line as the last part of this process to initialize the search process. In some cases, where the edge detection algorithm proceeds no major edge in a block, we empirically choose initial values of r = 0 and $\theta = 0$. The geometry-adaptive partitioning search can be optionally skipped in these cases as well.

The first line segment found in the previous step is exclusively based on texture information. The next step is to search for the best line segment that minimizes (1), i.e. to separate the block according to a motion field discontinuity rather than only texture information. This is a complicated task as the true MV for every pixel is unknown at this step. The MV for each block is currently a coarse estimation and varies according to the size and shape of the block. The cost function in (1) relies on the radius *r* and angle θ of the line segment. Jointly optimizing these two parameters can be demanding and protracted [3].

Often complex optimization problems are reduced to separable problems for the purpose of increasing speed. So, the joint optimization problem is divided into two separate optimization problems (of radius *r* and angle θ).

a) The first step is to vary the radius *r* within a predefined range with small increments. The angle θ is kept fixed at this stage. Experiments show that varying *r* with a single pixel increment leads to inferior performance [22]-[24]. To attain extra precise increments of *r*, we vary the *x*-intercept and *y*-intercept of the line simultaneously with a single pixel increment. Varying the intercepts concurrently ensures that the line segment moves parallel to its original position. Hence, the angle θ remains unchanged.

b) Secondly, a single-pixel increment/decrement of the *x* and *y*-intercepts produces a sub-pixel of the radius *r*. This improved accuracy for *r* produces superior results to those reported in [22], as will be revealed in Section 5.

For coding purposes, a dictionary of possible partitions is a priori defined for

$$r : r \in \left[0, \frac{\sqrt{2}MB_{size}}{2}\right), \qquad \text{where } r \subset \{0, \Delta r, 2.\Delta r, ...\},$$

and θ: $\{if\ \mathrm{r} = 0 \text{ then } \theta \in [0,180), \text{else } \theta \in [0,360)\}$,

where $\theta \subset \{0, \Delta\theta, 2.\Delta\theta, ...\}$, being Δr and $\Delta\theta$ the selected sampling steps. The sampling indices for r and θ are coded to represent the partition edge. We note that for r=0, angles 0 and 90 are redundant, so they are removed.

Let *a* represent the distance of the *x* or *y*-intercept from the block center. It can be varied in a defined range [$a - \Delta a$, $a + \Delta a$] with increments of δa. For each value of *a* (at each iteration), the corresponding sub-blocks are motion compensated to minimize (1). Once the best *a* (and the corresponding *r*) is found for a particular block, the angle θ is altered within the range [$\theta - \Delta\theta$, $\theta + \Delta\theta$] with $\delta\theta$ increments. The best θ is found in this fashion for the previously selected *r*. This process is illustrated in Figure 3.

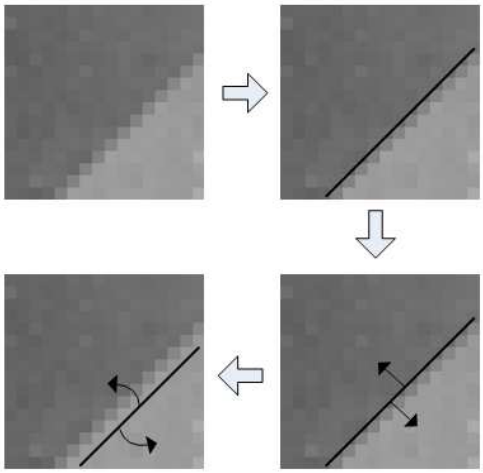


Fig. 3. A demonstration of the fast partition search [3].

In [3], the motion-adaptive partitioning is coded as a new inter-coding mode and the mode decision process is customized to allow the first choice between standard inter coding modes and the new mode. Besides, the translational motion model is employed in the MC stage. The fast geometry adaptive block partitioning algorithm can be recapitulated as follows [3]:

1. *For all macro-block*
   - *1.1. For each inter mode*
     - *1.1.1. Calculate rate and distortion*
   - *1.2. For geometry mode*
     - *1.2.1. Find the leading edge using Sobel edge detection and the Hough transform*
     - *1.2.2. Find the best r by varying a*
     - *1.2.3. Find the best θ*
     - *1.2.4. Calculate rate and distortion*
2. *Select the best coding mode using* λ_{mode}
3. *If the mode is quad, split and repeat the 1st and 2nd steps for each sub-blocks*

Regarding encoding issues, the distance of the *x* or *y*-intercept (represented by *a*) is coded with 1-pixel accuracy. For example, 0 is represented using 1 bit (1), 1 is represented using 3 bits (010), -1 is represented using 3 bits (011) etc. It is known that when the absolute value of $\theta \leq 45°$, the length of the *x*-intercept is smaller than the length of the *y*-intercept. That's why; the *x*-intercept is coded because it requires fewer bits. In the same manner, when $\theta > 45°$, the *y*-intercept is coded. The decoder comprehends whether the *x* or *y*-intercept

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

is coded based on the value of the angle θ. The angle θ is quantized by rounding to the nearest 5 degrees and coded using the same code. Figure 4 shows an example of how coding modes were selected in H.264 and the proposed method [19].

The coding mode of each macroblock is stated by showing the partition shape for that macroblock. In order to distinguish intra-macroblocks easily, they are shown with altered brightness levels. It can be seen that, compared to the results with H.264, through the proposed method in [19], the division of blocks in the boundary area between the hat and the face, and the background are corresponding to the shape of the moving object.

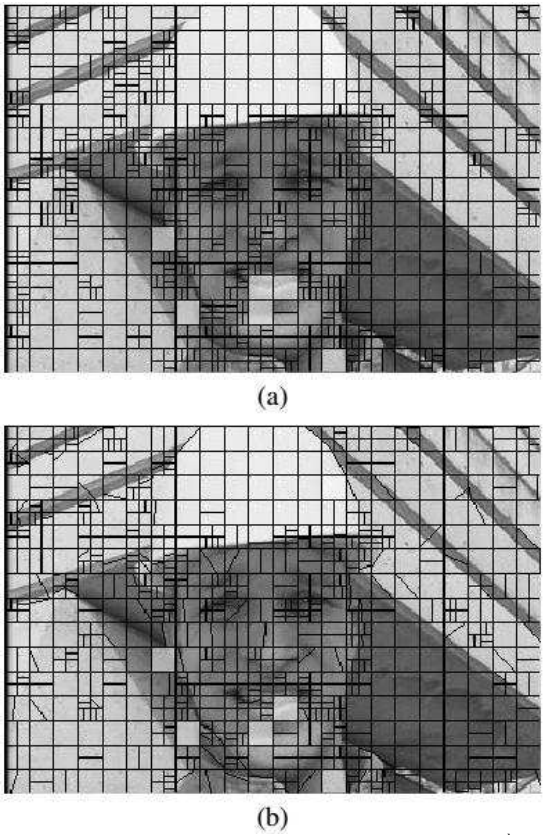


Fig. 4. Divided region shapes for the test sequence"Foreman" (16th frame, quantization parameter = 23). (a) H.264 (b) Proposed Method [19].

In order to reduce the search complexity related with geometry partitioning in HEVC which includes a more flexible rectangular block partition algorithm and H.264/MPEG-4 AVC, several simplified geometry block partition schemes have also been described in [22], [24] and [33] based on H.264/MPEG-4 AVC. In [22], the search was performed on a subset of candidate partition lines instead of the full set. The lines were predefined with seed values and reduced searching ranges for the two parameters that describe the partition line. The scheme described in [24] employed a two-step independent search to simplify the search process. In [33], a simplified geometry partitioning approach was introduced that restricted the set of candidate partition lines to a subset known as most valuable partitions that was obtained by statistical analysis.

With the intension of searching complexity reduction for geometry partitioning in video coding, [34] proposed a texture difference based partition line selection technique by skipping geometry partitioning for uniform MBs and a background-compensated prediction method to improve the R-D performance of the geometry partitioning prediction by reducing its MV overhead. Here by combining the texture difference approach with wedge-based geometry partitioning, they obtain a texture difference wedge based geometry partitioning (T-WGP) scheme. Similarly, they design a texture difference object-boundary-based geometry partitioning (T-OBGP) approach in a similar manner as for the T-WGP approach, where the object boundary in each reference MB is obtained by the Canny edge detector [32]. The T-WGP and T-OBGP approaches were both implemented on the 16×16 MB level of the H.264/MPEG-4 AVC coding design. Additionally, by integrating the proposed approach in [34] into object-boundary based geometry partitioning, they achieved about 10% bit-rate savings relative to the FS approach while keeping the complexity at about the same level as their proposed complexity reduced wedge-based geometry partitioning. The proposed approaches were a further extension of their previous work [35] which was applied only for the wedge based geometry partitioning. Without additional ME for the partition line search, these approaches can significantly reduce the computational complexity of the prior rate–distortion optimization (RDO) based search schemes.

Furthermore, [36] presented techniques for enhancing the efficiency and dropping the complexity of geometry partitioning schemes particularly for systems in which high-quality depth maps are available. In some video applications, such as multiview, free viewpoint, and 3-D videos a high-quality depth maps may be available. In these systems, the edges in the depth maps usually represent actual object boundaries. So, [36] incorporated depth map usage into the described approaches to generate a more accurate partitioning. For computational complexity reduction, they introduced a texture-difference-based approach for determining the position of the partition line. The texture-difference-based approach can be enhanced as a texture-plus depth- difference based approach by integrating the depth information into the partition line search criterion and the skipping strategy.

4. Geometry/Motion-Adaptive Partitioned Elastic Motion Compensation

4.1 Review of Transformed Block-Based Motion Compensation Methods

Block matching algorithm (BMA) adopts a motion model that describes the motion of image objects by the translational motion of blocks. Some problems of the BMA, like blocking artifacts, are caused by the insufficiency of the motion model to represent the real situations. To overcome this

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

shortcoming, a lot of coding methods that use more sophisticated motion models have been proposed [37]. In this section, we give a brief review of the motion compensation methods examined in [7]. Given the small bit allotment for motion information in very low bit-rate coding, motion compensation using BMA fails to sustain an adequate level of prediction errors. The reason is that the motion model through spatial transformation, assumed in block matching cannot estimate the motion in the real world precisely with a small number of parameters (i.e. MVs). To design an efficient motion compensation method for very low bit-rate video coding, [7] addressed the issues of adopting more sophisticated spatial transformations than block matching and developing a motion estimation algorithm that is suitable for these spatial transformations.

The principles of transformed block-based MC can be regarded as a method that applies spatial transformations on the blocks of the frame. Through simplification of this transformation, new motion compensation methods that adopt different motion models can be developed. Recently, many such ideas have been proposed independently. The spatial transformations ideas can be categorized into methods based on affine transformation [7] and [10], bilinear transformation [17] and perspective transformation [6], which are all spatial transformations that are used frequently in the field of computer graphics.

The performance of the MC methods, which utilize the spatial transformations and block-based motion estimation algorithms, is evaluated theoretically and experimentally in [7] using the following criteria: prediction error, the amount of motion information, and computation cost. Experimental results show that in the proposed method, the amount of motion information required to achieve a fixed level of the prediction error is reduced to one-fourth that of block matching.

4.1.1 Transformation functions

MC techniques can be defined as methods that divide frames into local regions (blocks or patches) and approximate for all region a set of motion parameters. The procedure that synthesizes the predicted image of the n[th] frame $\hat{I}_n(x,y)$ from the decoded image of the previous frame $\tilde{I}_{n-1}(x',y')$ can be regarded as an image warping or texture mapping process. This process can be written as

$$\hat{I}_n(x,y) = \tilde{I}_{n-1}\big(f(x,y), g(x,y)\big) \tag{5}$$

Where, the geometric relationship between $\hat{I}_n(x,y)$ and $\tilde{I}_{n-1}(x',y')$ is defined by the *transformation functions* $x' = f(x,y)$ *and* $y' = g(x,y)$. *when (x',y')* is not a sampling point of the image, the intensity value $\tilde{I}_{n-1}(x',y')$ is obtained by interpolation.

We adopt bilinear interpolation as the interpolation function. By this function, the intensity value at point (x',y') is obtained by

$$\begin{aligned}\tilde{I}_{n-1}(x',y') = (1-\alpha)\big((1-\beta)\tilde{I}_{n-1}(X,Y) + \beta\tilde{I}_{n-1}(X+1,Y)\big) \\ + \alpha\big((1-\beta)\tilde{I}_{n-1}(X,Y+1) + \beta\tilde{I}_{n-1}(X+1,Y+1)\big)\end{aligned} \tag{6}$$

Where, *(X, Y)* is the integral part and *(α,β)* is the fractional part of the coordinate *(x',*y'). This method requires six multiplications for each pixel. Various fast algorithms to speed up the computation of (6) have been proposed in [38].

For example in BMA, the transformation functions for the pixels in the i[th]-block of the image are

$$\begin{aligned} f(x,y) &= x - \hat{u}_i \\ g(x,y) &= y - \hat{v}_i \end{aligned} \tag{7}$$

Where, $(\hat{u}_i, \hat{v}_i)$ - the estimated MV of the i[th] block.

The motion parameters are estimated by first extracting the motion vectors of the vertices of the patch and then computing the parameters of the patch from these motion vectors. Considering the above-mentioned conditions, we examine the following three functions.

a) Affine Transformation

The functions for the pixels incorporated in the i[th] patch or block of the frame are

$$\begin{aligned} f(x,y) &= a_{i1}x + a_{i2}y + a_{i3} \\ g(x,y) &= a_{i4}x + a_{i5}y + a_{i6} \end{aligned} \tag{8}$$

Where, a_{i1}~a_{i6} are the six motion parameters of the patch. Since there are six degrees of freedom, the parameters of this transformation can be found out from MVs of three vertices. Therefore, the patches of this transformation are triangles [10].

b) Bilinear Transformation

The transformation functions are

$$\begin{aligned} f(x,y) &= a_{i1}xy + a_{i2}x + a_{i3}y + a_{i4} \\ g(x,y) &= a_{i5}xy + a_{i6}x + a_{i7}y + a_{i8} \end{aligned} \tag{9}$$

Where, a_{i1}~a_{i8} denote the eight motion parameters. Since there are eight degrees of freedom, it requires a quadrilateral patch. However, only rectangular patches are allowable, otherwise, the computation of the motion parameters become complicated [17].

c) Perspective Transformation

The transformation functions are

$$\begin{aligned} f(x,y) &= \frac{a_{i1}x + a_{i2}y + a_{i3}}{a_{i7}x + a_{i8}y + 1} \\ g(x,y) &= \frac{a_{i4}x + a_{i5}y + a_{i6}}{a_{i7}x + a_{i8}y + 1} \end{aligned} \tag{10}$$

Where, a_{i1}~a_{i8} denote the parameters of the transformation. As in bilinear transformation, the patches are quadrilaterals. The drawback of this transformation is the computational cost of the scanline algorithm: it requires two divisions for each pixel. Despite this disadvantage, this transformation describes the motion of a rigid planar patch when a perspective projection is assumed [7].

4.2 Elastic Motion Model

Non-rigid image registration or Elastic motion model techniques have been broadly used in object tracking, image stabilization, medical imaging and motion analysis

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

applications [39]. It is the task of finding a correspondence function mapping coordinates from a reference frame to coordinates of homologous points in a test frame [3][11]. In fact, the basic image registration technique can be applied and functional to the block matching framework without any major modification. Assume two blocks $I(x_i,y_i)$ and $I'(x'_i,y'_i)$ are associated with the following coordinate transformation of the form:

$$x'_i = x_i + \sum_{k=1}^{P/2} m_k \phi_k(x_i, y_i)$$
$$y'_i = y_i + \sum_{k=P/2+1}^{P} m_k \phi_k(x_i, y_i) \quad (11)$$

Where, P is the number of motion parameters, m_k are the motion parameters and $\phi_k(\,)$ are basis functions that describe arbitrarily adaptable mappings between the coordinates of "*I"* and "*I'"*. Examples of basis functions that have been used in [39] include harmonic functions, B-splines, polynomials, radial basis functions and wavelets.

In this paper, a set of discrete cosines can be used from [3] and [11] to capture elastic or non-rigid motions. This selection was based on the requirement to have the majority compacted parameterization of the non-translational motion. Discrete cosines have the capability to symbolize smooth motion fields with the smallest number of coefficients. The choice of discrete cosines is also enthused by its common popularity and hardware implementation in various image and video processing applications. Hence, for the coordinate transform of the M x N blocks to be registered, the basis functions are given by

$$\phi_k(x_i, y_i) = \phi_{k+P/2}(x_i, y_i) = \cos\left(\frac{(2x_i+1)\pi u}{2M}\right)\cos\left(\frac{(2y_i+1)\pi v}{2N}\right) \quad (12)$$

for $k = su + v + 1$, $u,v = 0,...,s\text{-}1$ and $s = \sqrt{\frac{P}{2}}$

Having the defined motion model, the subsequent step is to compute the motion parameters that will enable the best prediction of a block from its reference video sequence. The standard gradient-based image registration techniques are well-known and have been used broadly in image and video processing as well as in machine vision applications. These methods typically attempt to minimization of the sum of squared difference (SSD) between "*I"* and "*I'"* [40].
The SSD can be written as

$$E = \sum_{i=1}^{T}\left[I'(x'_i, y'_i) - I(x_i, y_i)\right]^2 = \sum_{i=1}^{T} e_i^2 \quad (13)$$

Here, T denotes the number of pixels in the area where the two frames overlap and e_i is the difference in intensity values of pixel i. A commonly used technique for minimizing E is a Gauss–Newton gradient descent nonlinear optimization algorithm that is given in [40]. In this technique, a first-order Taylor approximation of the SSD is used to linearize the nonlinear expression in (13). Therefore, after updating the motion parameters by Δm, the expression becomes

$$E' = \sum_{i=1}^{T}\left[\left(I' + \frac{\partial I'}{\partial m}\Delta m\right) - I\right]^2 = \sum_{i=1}^{T}\left[\frac{\partial I'}{\partial m}\Delta m + e_i\right]^2 \quad (14)$$

The partial derivative of E with respect to the parameter updates is given by

$$\frac{\partial E'}{\partial \Delta m} = 2\sum_{i=1}^{T}\frac{\partial I'}{\partial m}\left[\frac{\partial I'}{\partial m}\Delta m + e_i\right] \quad (15)$$

If we let this partial derivative equal zero, the error after updating will be minimized. Therefore, the parameter updates can be given by

$$\sum_{i=1}^{T}\left[\frac{\partial I'}{\partial m}\right]^2 \Delta \mathrm{m} = -\sum_{i=1}^{T}\frac{\partial I'}{\partial m} e_i \quad (16)$$

Since transforms typically have multiple motion parameters, this equation can be written using matrix notation as

$$H\Delta m = b \quad (17)$$

Where, **H** is called as the Hessian matrix and **b** is a weighted gradient vector. The elements of **H** and **b** are given by

$$H_k = \sum_{i=1}^{T}\frac{\partial I'}{\partial m_k}\frac{\partial I'}{\partial m_k} \quad \text{and} \quad \mathrm{b_k} = -\sum_{i=1}^{T}\frac{\partial \mathrm{I'}}{\partial \mathrm{m_k}} e_i$$

and can be calculated using the chain rule as follows:

$$\frac{\partial I'}{\partial m_k} = \frac{\partial I'}{\partial x'_i}\frac{\partial x'_i}{\partial m_k} + \frac{\partial I'}{\partial y'_i}\frac{\partial y'_i}{\partial m_k} \quad (18)$$

In this chain rule equation, the terms $\partial I'/\partial x'_i$ and $\partial I'/\partial \mathrm{y'_i}$ are simply the horizontal and vertical gradients of I'. For elastic motion model, the terms $\frac{\partial x'_i}{\partial m_k} = \phi_k(x_i, y_i)$ and $\frac{\partial y'_i}{\partial m_k} = \phi_k(x_i, y_i)$ are equal to the basis functions of the warping function. The motion parameters are then updated iteratively as follows:

$$m^{t+1} = m^t + \Delta m = m^t + H^{-1}b \quad (19)$$

Where, t denotes the iteration number and x'_i, y'_i, I', **H**, and **b** are recalculated at each iteration using the updated motion parameters. In this method, the parameters are iteratively updated until a minimum of E is found. Hence, the estimation of the parameter updates which are essential to minimize SSD is calculated. These updates are then added to the existing motion parameters and hence a warped version of I' is computed with the new warping function. This warped version of I' is used in the subsequent iteration and the process maintains until some threshold or stopping criterion is reached (maximum number of iterations or minimum change in SSD). Investigational results show that 10 to 16 iterations are adequate to compute the elastic motion parameters for the typical blocks in a frame [3][11].

It is significant that the selection of an elastic model here is enthused by its superior flexibility in terms of the number of motion parameters that can be easily calculated without altering the gradient-descent algorithm [3]. Other higher order models such as affine, perspective or polynomial

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

models with 6 parameters, 8 parameters, and 12 parameters respectively are limited to a fixed set of motion parameters. Conversely, non-rigid image registration appends greater flexibility in choosing the number of parameters reliable with the size of a block and the necessity of the application. Besides, affine and perspective models are basically global motion estimation methods so they are not adequate to capture complex motions or deformations that are typically seen on videos.

In the process of eradicating the 'blockwise-constant motion' assumption, the use of an elastic motion model is an important step with respect to the classical translation motion model whereby all the pixels in a block are allotted the same MV [3]. Though the assumption is straightforward, it can be costly in cases where the underlying motion is complex. In these conditions, elastic motion compensation technique does better by capturing the camera effects, rotation as well as deformation of objects etc. The model presented in [3] can capture complex motions in larger blocks as well, which then enables better rate-distortion optimization with minimum computational cost.

4.3 Elastic Motion Compensation with Geometric Partitions

In video coding, inter-frame coding involves finding the best prediction for a block of pixels in the current frame from a reference frame which has formerly been transmitted to the decoder. Where, the MVs are used to describe the translation of the best prediction block in the reference frame. This section combines the motion/geometry-adaptive block slicing presented in section 3.2 with the elastic motion model described in section 4.2, to improve the compression efficiency of inter-frame video coding [3]. As stated prior, both motion/geometry-adaptive block slicing and elastic motion models give enhanced predictions by using different characteristics in video frames. Therefore, it is expected that they can gain efficiency from each other. As shown in [11], an elastic motion model is well performed with larger macro-blocks with 2-levels of partitioning. So, we use the same block size of 32×32 in this scheme with 2 levels of partitioning as [3]. At the initial level of partitioning, blocks of 32×16, 16×32 or 16×16 pixels are produced and the second level of partitioning further slices the 16×16 blocks into 16×8, 8×16 and 8×8 blocks, respectively.

For a rate-distortion optimization implementation of the H.264/AVC encoder, the mode choice is performed by selecting the mode that minimizes a Lagrangian cost function, J_{mode} given by [11]:

$$J_{mode} = D_{mode} + \lambda_{mode} R_{mode} \quad (20)$$

Where, D_{mode} is the SSD between the original and reconstructed macroblocks, R_{mode} is the bit-rate required to transmit the MVs, transform coefficients of the prediction residual and macroblock type information and λ_{mode} is a Lagrangian multiplier. The choice of MV for a particular partition is also carried out using a Lagrangian cost function given by [11]:

$$J_{motion} = D_{motion} + \lambda_{motion} R_{motion} \quad (21)$$

Where, D_{motion} is the sum of the absolute differences between the current partition and the partition in the reference frame indicated by the candidate MV, R_{motion} is the bit-rate required to transmit the MVs and λ_{motion} is a Lagrangian multiplier. The detail on rate-distortion optimization in video compression is given in [41].

During elastic motion compensation algorithm, the mode decision procedure is customized to allow preferences between H.264 inter prediction modes and the geometric-adaptive partitioning mode as well as a choice between a translational and elastic motion transformation model. The common inter predictions are performed as usual with the only exception being the use of super macro-blocks. Besides, elastic motion compensation is performed for each block or sub-blocks under each mode. Subsequently, the geometry adaptive block slicing is carried out (as described in section 3) as an additional inter-coding mode. Finally, to take care of any complex motion or deformation, this predicted block is further motion compensated using an elastic motion transformation model. The best coding mode and motion model can be selected using the Lagrangian rate-distortion optimization, given in equations (20) and (21) [31]. The enhanced motion compensation algorithm can be summarized as follows [3] [11]:

1. *For each 32×32 macro-block*
   - *1.1. For every inter mode*
     - *1.1.1 For each block*
       - *1.1.1.1 Calculate rate and distortion using translational MVs*
       - *1.1.1.2 Calculate rate and distortion using elastic MVs*
       - *1.1.1.3 Select the best motion parameters using λ_{motion}*
   - *1.2. For geometry mode*
     - *1.2.1 Select the best line segment*
     - *1.2.2 Calculate rate and distortion using translational MVs*
     - *1.2.3 Calculate rate and distortion using elastic MVs*
     - *1.2.4 Select the best motion model using λ_{motion}*
     - *1.2.5 If cost for elastic motion is less than translational motion use rate and distortion for elastic motion compensation for this partition*
2. *Select the best coding mode (minimum cost) using λ_{mode}*
3. *If the mode is quad, split and repeat steps 1 and 2 for each sub-blocks*

The motion parameters are computed for a warping function using discrete cosine basis functions as described in section 4.2. To calculate these parameters, a partition from the reference frame that corresponds to the integer pixel translational MVs is used. This reference partition is then registered to the current partition using Gauss-Newton gradient-descent algorithm as explained in [40]. The motion parameters found from the registration process are then quantized and used to determine a warped version of the

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

reference partition which is used as the motion compensated prediction.

As to the encoding issues, translational MVs are coded in a manner similar to H.264/AVC with quarter-pixel accuracy. When a block is coded using this method, two sets of MVs are coded; translational MVs to indicate the integer pixel location of the prediction block and elastic MVs to define the superior prediction. In this case, translational MVs are coded with single pixel accuracy and the absolute values of the elastic MVs are coded with 1/4 and 1/8 pixel accuracy to code elastic parameters without much disparity in performance. However, 1/8 pixel accuracy usually yields the best rate-distortion tradeoff due to their high sensitivity.

A demonstration of block partitioning and motion modeling is given in Figure 5 [11]. It shows that in cases where the underlying motion of a region is complex, conventional H.264/AVC applies a piecewise-translational coding approach by dividing the region into multiple smaller blocks with independent MVs. Conversely, the proposed scheme is able to represent the same region using one large block efficiently [3]. The elastic motion model generates a smooth motion field which results in better prediction with reducedbit-rate.

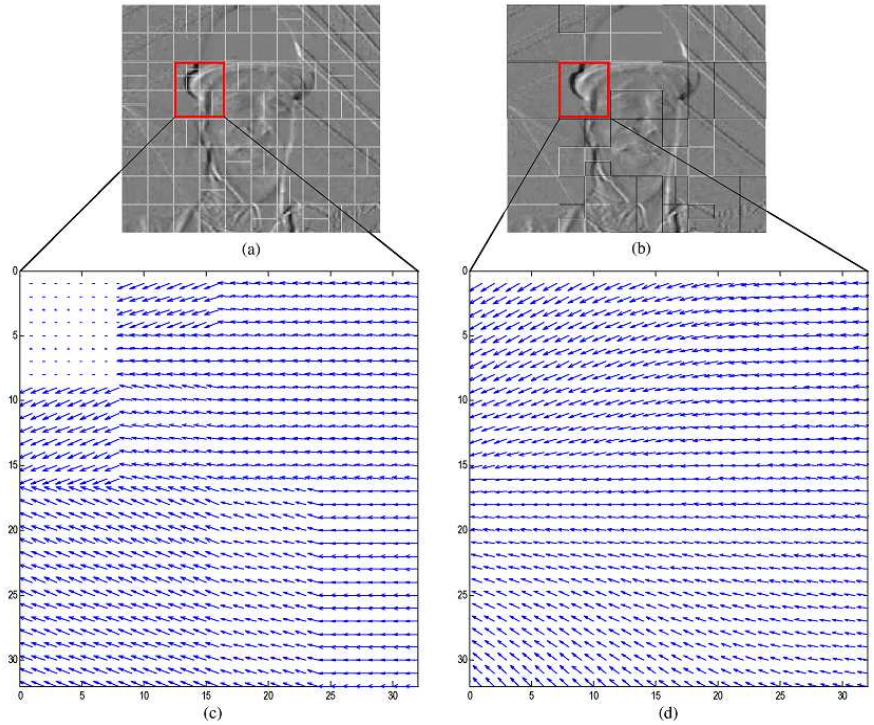


Fig. 5. Demonstration of block partitioning and motion modeling using the proposed algorithm in [10] (a) H.264, (b) Elastic motion model, (c) 2-D MV field of a selected 32 × 32 region in (a), (d) Smooth 2-D MV field for the same region generated in (b).

To evaluate the performance of the enhanced motion compensation algorithm described in the previous section, the algorithm was incorporated into a simplified video coder and 150 frames of the test sequence Foreman were coded. The specific details of the experiments performed are as follows [3]:

The first frame of the sequence was coded as I frame and all other frames were coded as P frames. Only 1 reference frame was allowed. The output bit-rate of the coder was the sum of the rate required for entropy coded motion parameters, transform coefficients, and macroblock type parameters. For ease of implementation, 4x4 DCTs were used to transform the prediction residuals, H.264 VLCs were used to entropy code the motion and macroblock-type parameters and H.263 VLCs were used to code the transform coefficients. Elastic registration was performed using P = 8 parameters and the motion parameters were quantized using the following equation [42].

$$m'_k = round\left(\frac{m_k}{0.25}\right) \tag{22}$$

The quantized motion parameters were entropy coded independently using the same H.264/AVC VLCs as the translational motion parameters. The transform coefficients were quantized using the H.264/AVC quantization formula and the Lagrangian multipliers for the rate-distortion optimizations were calculated using the equations [42]

$$\lambda_{\text{mode}} = 0.85\left(2^{QP/3}\right)$$
$$\lambda_{motion} = \sqrt{\lambda_{\text{mode}}} \tag{23}$$

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Figure 6 illustrates an example of the macroblock mode choices made for the two algorithms. It shows the borders of the partitions selected for frame 31 of the Foreman sequence superimposed on the difference between the original frames 31 and 30 when QP = 37. In Fig. 6b the partitions selected to use the elastic motion parameters are shown with a black border. It is shown that in cases where the underlying motion is complex, it can be more efficient to estimate an elastic motion field using 8-parameters relatively than represent the motion with smaller partitions each with its own translational MVs.

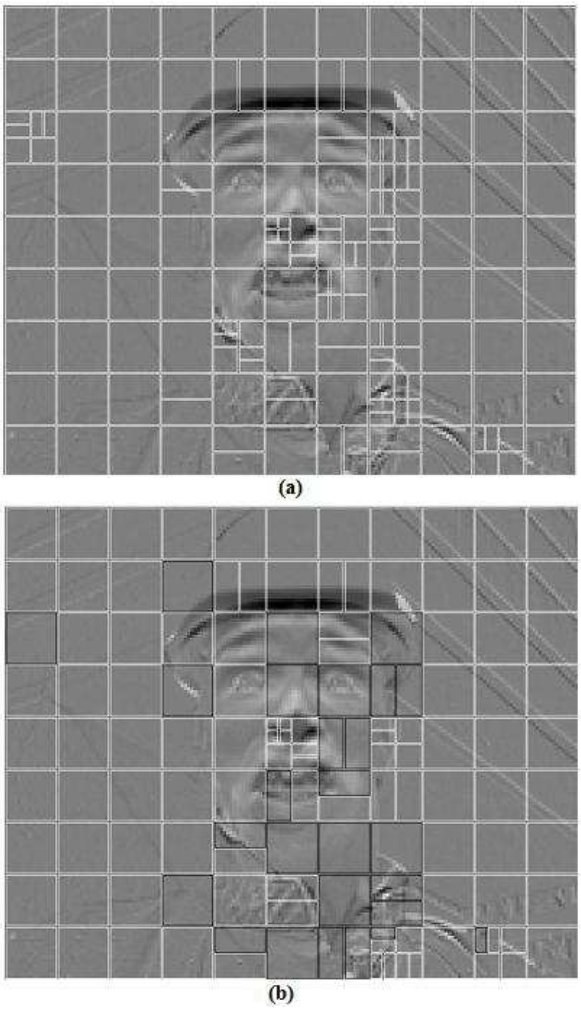


Fig.6. Selected macroblock modes for (a) H.264 motion compensation (b) Elastic motion compensation - blocks with black borders were selected to use elastic motion parameters.

4.4 Computational Complexity

The better prediction technique through motion/geometry-adaptive partitioning presented here comes with some computational load. The papers, [3] and [11] use the *Big-O* notation to convey and evaluate computational complexity. The computational complexity of the Sobel operation, which is performed only once for each block, is equal to that of a linear convolution operation which is given as $O(N^2)$ where N is the size of the block. This complexity can be reduced to $O(N\ logN)$ by using Fast Fourier Transform (FFT) based convolution. However, the computational complexity of the Sobel operation and the successive Hough transform is greatly less than that required for the repeated ME performed in the exhaustive search in [19], which requires 1024 iterations for a 16×16 block.

The computational cost of elastic motion model is firstly incurred by the inverse compositional gradient descent optimization [40] used to compute the elastic motion parameters [3][11]. The complexity of pre-computation and subsequent iterations are $O(P^2T)$ and $O(PT+ P^3)$ respectively, where P is the number of elastic motion parameters and T is the total number of pixels in the block. In this paper, $P = 8$ and $T \leq 32\times32$ as [3]. Usually, it takes about 10-16 iterations to get the best motion parameters. It has also been shown in [11] that reduction in the number of iterations per block significantly increases the speed of the process with only a small degradation in rate-distortion performance.

The next computational burden is because of the repeated MEs carried out during the geometry partition search. The complexity of each iteration is $O(T^2)$ where $T \leq 32\times32$. In contrast, H.264/AVC ME costs $O(T^2)$ where $T \leq 16\times16$. However, the computational time because of this extra complexity could be afforded for offline processing and storage applications with a relaxed time limit [3]. So far, there is some issue for real-time applications with very limited resources. However, since the load of the complexity is on the encoder part, the proposed algorithm [3] puts a comparatively small load on the decoder side. The overall complexity can be reduced in different ways. At the outset, the complexity of BMA can be lowered using fast block search techniques [2]. Next, the gradient descent optimization complexity can be decreased using less iteration per block, introducing early termination criteria to exit the iterative loop or using quick gradient descent optimization using reduced bit-depth [11].

5. EVALUATIONS THROUGH SIMULATION RESULTS

In this paper, the performance of the proposed scheme [3] is evaluated by considering different video conferencing and streaming applications. The characteristics of the sequences used and the experimental conditions are listed in Table 1. Here, we consider only the luminance component to simplify the experimentations. For video conferencing and streaming applications, the first frame is coded as I-frame and subsequent frames are coded as P.

In the experimental simulation, we investigate the effect of geometry-adaptive partitioning and elastic motion model in conjunction with super macro-blocks. In other words, we examine how geometry information can influence the performance of elastic motion compensation. We compare the performance of the algorithm to that of the following algorithms [3]:

a. H.264 - H.264/AVC implementation (Macro-block size of 16×16 with 2 levels of partitioning). The motion search

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

range is limited to ±8 and ±4 pixels for the first and second levels of full search motion estimation, respectively. This algorithm was selected as the modern video coding standard [13].

b. *Geom* - The fast geometry adaptive block partitioning scheme proposed in section 2. This scheme was simulated using $\Delta a = 3$, $\delta a = 1$, $\Delta\theta = 20$ and $\delta\theta = 5$. This results in 16 iterations per block regardless of the block size.

c. *Elastic + Geom* - The geometry-assisted elastic motion compensation as presented in section 4.

d. *Elastic* - H.264 enhanced with elastic motion model and super macro-blocks presented in [11].

TABLE 1. LIST OF SEQUENCES AND EXPERIMENTAL CONDITIONS

Sequence	Resolution	Duration	Characteristics	Experimental Conditions
Carphone	QCIF	10s	Fast camera and content motion with landscape passing	▪ *Search range :* ±8 & ±4 pixels for the 1st & 2nd levels ▪ *Search precision:* 1/8 pixel (for H.264) ▪ Elastic MVs are coded with 1/8 precision ▪ *Deblocking filter:* OFF ▪ *Quantization parameter:* Constant (37)
Forman	QCIF	10s	Fast camera and content motion with pan at the end	
Miss America	QCIF	5s	Still camera and content motion	
Mother & Daughter	QCIF	10s	Still camera on human subjects	
Forman	CIF	10s	Fast camera and content motion with pan at the end	
Horse	CIF	10s	Fast camera and fast complex motion	
Mobile Calendar	CIF	10s	Camera zoom and panning with complex content motion	
Paris	CIF	10s	Still camera on human subjects: Video-conferencing content	

TABLE 2. FIXED BIT-RATE RD-RESULTS FOR QP=36

		H.264 based	Geom	Elastic	Elastic + Geom	Gain
	QCIF 10 Hz					
48 kbps	Forman	29.29	29.54	29.94	**30.18**	0.89
	Mother & Daughter	35.04	35.33	35.55	**35.73**	0.69
64 kbps	Forman	31.67	31.91	32.00	**32.16**	0.49
	Mother & Daughter	36.89	37.15	37.24	**37.33**	0.44
96 kbps	Forman	33.91	34.24	34.35	**34.57**	0.66
	QCIF 15 Hz					
64 kbps	Forman	29.75	29.98	30.82	**31.15**	1.40
	Mother & Daughter	35.48	35.74	**35.98**	35.97	0.49
96 kbps	Forman	32.96	33.24	33.60	**33.75**	0.79
	Mother & Daughter	37.95	38.25	**38.54**	38.37	0.42
128 kbps	Forman	34.76	35.10	35.37	**35.44**	0.68
	Mother & Daughter	39.54	39.80	**39.96**	39.80	0.26
	CIF 10 Hz					
512 kbps	Forman	33.73	34.02	34.35	**34.58**	0.85
	Horse	25.02	25.31	25.40	**25.66**	0.64
768 kbps	Forman	35.97	36.26	36.46	**36.70**	0.73
	Horse	27.35	27.51	27.67	**28.00**	0.65
1 Mbps	Forman	37.69	37.93	38.08	**38.20**	0.51
	Horse	29.24	29.48	29.72	**29.98**	0.74
						Average gain = 0.60 dB

The results are given in terms of PSNR (dB). The best performing scheme is shown as **bold**

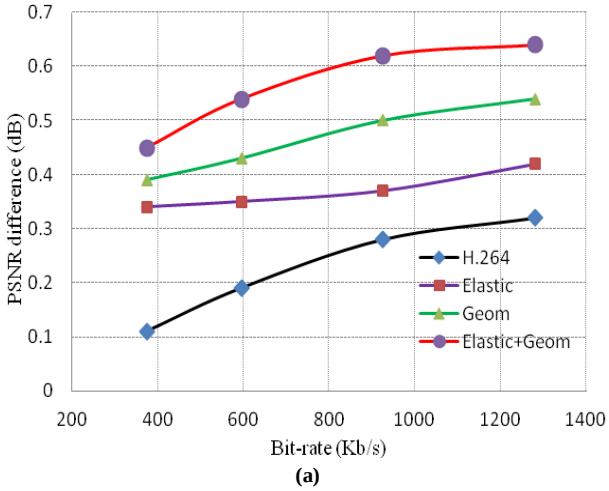
https://sites.google.com/site/ijcsis/
ISSN 1947-5500

5.1 Rate-Distortion Comparison

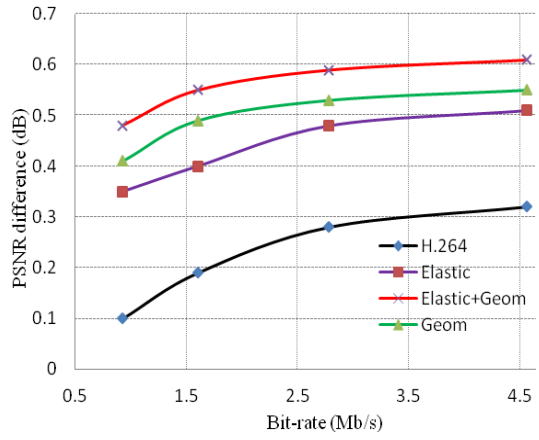
Table 2 shows the simulation results for the test sequences under different test conditions (bitrates) selected to represent a range of different applications (typical interactive video conferencing and streaming). Video conferencing applications generally support low to medium bit rates and picture resolutions—quarter common intermediate format (QCIF) and common intermediate format (CIF) [3]. Video streaming supports the same resolutions but with better bit rates. The best performing technique is shown in bold and the gain of the geometry-assisted elastic motion compensation scheme over H.264/AVC is given in the rightmost column. It can be observed that a motion-adaptive sliced block with elastic motion model scheme considerably outperforms the H.264 standard by the gain of 0.42–0.85 dB in most of the cases.

When motion-adaptive partitioning is combined with elastic motion compensation, we get an overall gain of 0.6 dB in PSNR, which signify a good bitrate savings. Therefore, it is evident that the introduction of geometry information certainly helps the elastic motion model to get an extra accuracy of prediction, and hence plummeting the residual bits to be coded. So, motion-adaptive partitioning can be considered and applied as an elective addition to elastic motion compensation when the inter-frame coding can afford the extra computational time.

With the selected rate-distortion curves, the overall Bjøntegaard Delta (BD) PSNR difference [43] for the simplified coder with and without the enhanced motion compensation algorithm are also presented in Fig.7. The Bjøntegaard delta bitrate is computed using piece-wise cubic interpolation with the tested quantization parameter (QP) values of 22, 27, 32 and 36. Fig. 7a and b illustrate the PSNR improvements achieved by the 'Elastic+Geom', 'Geom', 'Elastic', and H.264 approaches on the Forman and Horse CIF test sequences respectively. These curves show that a significant improvement in compression efficiency can be obtained by using the elastic motion parameters. It is evident here that the 'Elastic+Geom' scheme outperforms the H.264 over a wide range of bitrates. This enhancement is obtained due to a mutual benefit of super macro blocks and an elastic motion model. The motion model is able to characterize complex motions in typical video frames even when the block size is large. The use of larger blocks makes it efficient as well from a rate-distortion optimization point of view.



(a)



(b)

Fig. 7. Selected BD PSNR results for geometry-assisted elastic motion compensation. (a) Forman CIF (b) Horse CIF

The fast geometry adaptive block partitioning scheme attains a significant gain. However, sometimes, there is no significant gain at lower rates. This is for the reason that geometry information is believed to be inefficient at lower bit rates for rate-distortion optimization using typical blocks. It is also revealed that neither 'Geom' nor H.264/AVC performs very well with super macro-blocks when only translational MVs and 2-levels of partitioning are used. When larger blocks are introduced, we observe gains of up to 0.45-0.64 dB at lower rates. At higher rates, the gain diminishes or even poorer when compared to the H.264/AVC algorithm as shown in Fig.7b. This is due to the fact that block-wise constant translational MVs are not sufficient enough to capture the motion in larger blocks, even with the presence of geometry partitions. Less signaling and MV bits are required for super macro-blocks or larger blocks but, at medium and upper bit rates, this advantage is surpassed by the requirement to transmit far more residual bits.

Partition maps of a typical frame corresponding to each technique are also shown in Figure 8. These demonstrate how geometric information combines with elastic motion

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

compensation technique provides a more efficient prediction of the underlying motion field among the two successive frames. The use of geometry partitioning and an elastic motion model causes extra motion bits per block. However, larger blocks or super macro-blocks result in fewer numbers of blocks and therefore, reduce the cumulative overhead of motion bits. As a result, we obtain comparable or better prediction at the cost of much less motion and overhead bits.

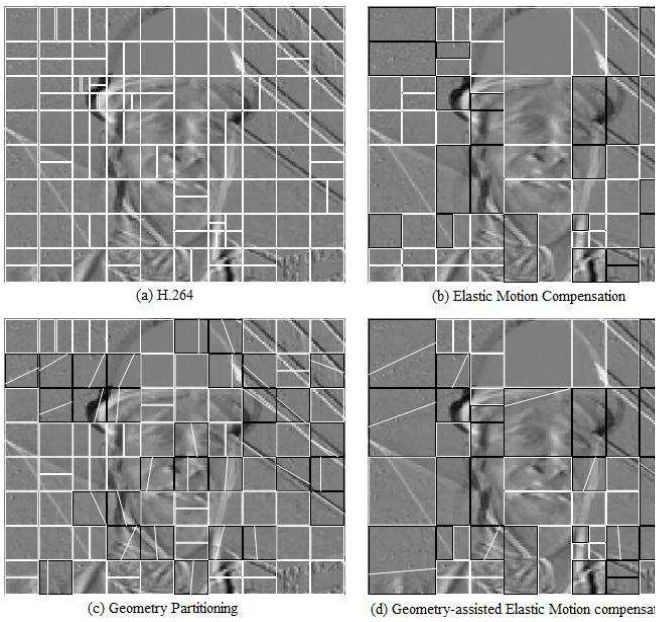


Fig. 8. Partition maps of a foreman frame using different schemes [3].

For possible coding modes for geometric partitions, considering the DC value or directional prediction with the same direction as the partition edge, the mode is tested in concurrence with other H.264/AVC Intra coding modes. BD PSNR coding results over 10 frames of Foreman sequence can be found in Figure 9. As shown by the results, Intra geometry adaptive modes can better exploit a strong geometric content of pictures, with observed compression gains of intra-data. Hence, one of the outcomes of such a mode is examined in intra-predicted data when using geometry-adapted coding.

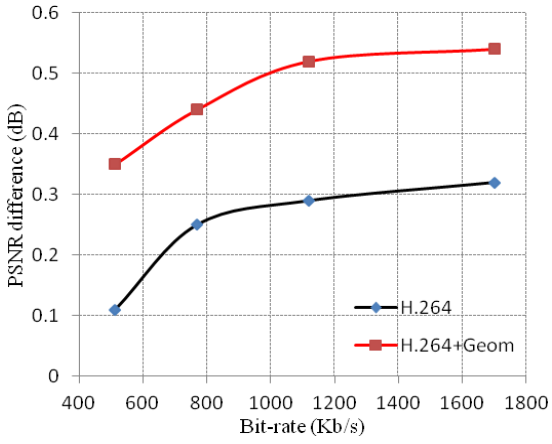


Fig. 9. BD PSNR Comparison Foreman CIF @ 15 Hz, 10 Intra frames.

Finally, Figure 10 shows the visual impact of geometry-adaptive partitions for QP=36 in Foreman sequence. H.264/AVC with geometric extensions has a sharper look with higher detail around contours. Moreover, higher detail on the neck is observed as well as Foreman upper lip is better reconstructed when geometry-adaptive modes are used.

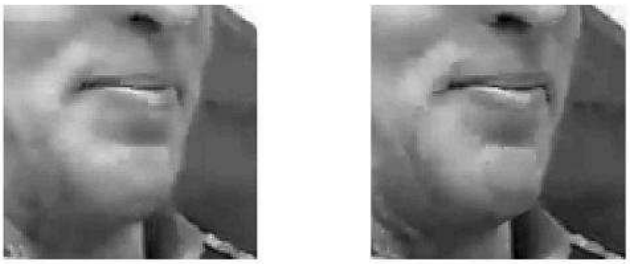


Fig. 10. Foreman sequence: 7th reconstructed picture visual detail. LEFT: H.264/AVC result, RIGHT: H.264/AVC + Geometry-adaptive mode result.

5.2 Encoding Time

It is shown that the H.264/AVC enhanced with elastic motion model and super macro-blocks scheme requires just about 1.5 times the computational time of the H.264 algorithm [11]. Similarly, the fast geometry adaptive block partitioning and geometry-assisted elastic motion compensation schemes increase the computational complexity by factors of 4.5 and 2.5 respectively [3]. This encoding time increase is mostly because of the ME phase for every likely partition. The reduction of computational complexity in the motion/geometry-assisted elastic motion compensation scheme is basically due to the use of larger blocks and hence a lesser number of blocks to be motion-compensated when compared to geometry adaptive block partitioning. This paper used motion ranges of ±8 and ±4 pixels for the 2-levels of exhaustive search technique for translational motion estimation experimentation to realize its complete potential as [3]. But, larger motion ranges can be used if the required computational time permits. However, fast block search approaches can significantly reduce this search time.

6. CONCLUSION

In this paper, we have presented a survey on inter-frame coding technique for video compression through geometry adaptive partitioning and an elastic motion model. Moreover, we have conducted experimentation in order to show the efficiency of using geometry information in conjunction with an elastic motion model and larger macro-blocks particularly presented in [3][11][22][23]. It is apparent that elastic motion compensation with geometry adaptive partitioning can significantly improve an inter-frame coding technique. These methods have extra potential to the baseline profile and higher profiles of H.264/AVC as well for better savings in bit rates. Therefore, both of these techniques are quite capable and they can be good candidates for future standards and specific applications (where the encoding time is not crucial).

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Yet, the computational time still remains a major concern for motion adaptive sliced blocks even with the fastest search algorithm techniques proposed in [3]. This will limit the combined use of these techniques in applications where the computational time is vital. In these conditions, elastic motion compensation alone is a reasonable option. But, off-line applications with no strict computational time requirements can be advantageous from the further savings in bitrate offered by motion adaptive sliced blocks. We also present several visual results to demonstrate the partition map achieved via different schemes. These examples are displayed by means of super- imposing the difference between successive frames in the sequence to provide an existence of the motion between the two frames.

REFERENCES

[1] E.A. Ayele and S. B. Dhok, "Review of proposed high efficiency video coding (HEVC) standard, "International Journal of Computer Applications, vol. 59(15), pp.1-9, 2012.

[2] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: a review and a new contribution," Proceedings of the IEEE, vol. 83, pp. 858-876, 1999.

[3] A. A. Muhit, M. R. Pickering, M. R. Frater, and J. F. Arnold, "Video coding using fast geometry-adaptive partitioning and an elastic motion model," Journal of Visual Communication and Image Representation, vol. 20(1), pp. 31-41, 2012.

[4] A. Paul, K. Bharanitharan and JiaJi Wu, "Algorithm and Architecture for Adaptive Motion Estimation in Video Processing," IETE Technical Review, Vol. 30, Issue 1, pp. 24-30, Jan-Feb 2013.

[5] M. Karczewicz, J. Nieweglowski, and P. Haavisto, "Video coding using motion compensation with polynomial motion vector fields," Signal Processing: Image Communication, vol. 10, pp. 63-91, 1997.

[6] V. Seferidis and M. Ghanbari, "General approach to block-matching motion estimation," Optical Engineering, vol. 32, pp. 1464-1474, 1993.

[7] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations," IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, pp. 339-367, 1994.

[8] H. Li and R. Forchheimer, "A transformed block-based motion compensation technique," IEEE Trans. on Communications, vol. 43, pp. 1673-1676, 1995.

[9] K. Zhang, M. Bober, and J. Kittler, "Image sequence coding using multiple-level segmentation and affine motion estimation," IEEE Journal on Selected Areas in Communication, vol. 15, pp. 1704-1713, 1997.

[10] R. C. Kordasiewicz, M. D. Gallant, and S. Shirani, "Affine motion prediction based on translational motion vectors," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, pp. 1388-1394, 2007.

[11] A. A. Muhit, M. R. Pickering, M. R. Frater, and J. F. Arnold, "Video coding using elastic motion model and larger blocks," IEEE Transactions on Circuits and Systems for Video Technology, vol. 20, pp. 661-672, 2010.

[12] S. Ma and C.-C. J. Kuo, "High-definition video coding with super-macroblocks," SPIE 6508, Visual Communications and Image Processing, 650816, 2007.

[13] ISO/IEC 14496-10 & ITU-T Recommendation, "H.264 Advanced Video Coding," 2003.

[14] ITU-T, ITU-T Recommendation H.261, "Video Codec for Audiovisual Services at $p\times64$ kbits," 1994.

[15] ITU-T, ITU-T Recommendation H.263, "Video Coding for Low Bit-rate Communication Version 1," 1995.

[16] M. H. Chan, Y. B. Yu, and A. G. Constantinides, "Variable size block matching motion compensation with application to video coding," IEE Proceedings I – Communications, speech and vision, vol. 137, pp. 205-212, 1990.

[17] G. J. Sullivan and R. L. Baker, "Rate distortion optimized motion compensation for video compression using fixed or variable size blocks," In Proc. of GLOBECOM, pp. 85-90, 1991.

[18] Il-K. Kim, S. Lee, M.-S. Cheon, T. Lee, and J.H. Park, "Coding efficiency improvement of HEVC using asymmetric motion partitioning," IEEE international Symposium on Broadband Multimedia systems and broadcasting, Seoul, 2012, pp. 1 – 4.

[19] S. Kondo and H. Sasai, "A motion compensation technique using sliced blocks in hybrid video coding," In Proc., Italy, ICIP, 2005, pp. II 305-8.

[20] E. M. Hung, R. L. D. Queiroz, and D. Mukherjee, "On macroblock partition for motion compensation," In Proc., ICIP, 2006, pp. 1697-1700.

[21] O. D. Escoda, P. Yin, C. Dai, and X. Li, "Geometry-adaptive block partitioning for video coding," IEEE, ICASSP, 2007, pp. 657-660.

[22] A. A. Muhit, M. R. Pickering, and M. R. Frater, "A Fast Approach for Geometry-Adaptive Block Partitioning," In Proc. 27th Conf. on Picture Coding Symposium, 2009a, pp. 413-416.

[23] A. A. Muhit, M. R. Pickering, and M. R. Frater, "Motion compensation using geometry and an elastic motion model," In Proc. 16th IEEE Intl. Conf. on Image Processing, ICIP, 2009b, pp. 621-624.

[24] R. U. Ferreira, E. M. Hung, R. L. Le Queiroz, and D. Mukherjee, "Efficiency improvements for a geometric-partition-based video coder," In Proc. 16th IEEE Intl. Conf. on Image Processing, ICIP, 2009, pp. 1009-1012.

[25] A. Singh, "Optical Flow Computation: A Unified Perspective," Los Alamitos, CA: IEEE Computer Soc. Press, 1991.

[26] M. I. Sezan and R. L. Lagendijk, "Motion Analysis and Image Sequence Processing," Boston, MA: Kluwer, 1993.

[27] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," Int. J. Computer Vision, vol. 12, pp. 43-77, 1994.

[28] G. Tziritas and C. Labit, "Motion Analysis for Image Sequence Coding," Amsterdam: Elsevier, 1994.

[29] D. R. Walker and K. R. Rao, "Improved pel-recursive motion compensation," IEEE Trans. Cornman., COM-32, pp. 1128-1134, 1984.

[30] J. D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information," Int. J. Computer Vision, vol. 5, pp. 77-104, 1990.

[31] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," IEEE Trans. Circuits and Systems for Video Technology, vol. 13, pp. 688-703, 2003.

[32] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," Prentice Hall, 2002.

[33] L. Guo, P. Yin, Y. Zheng, X. Lu, Q. Xu, and J. Sole, "Simplified geometry-adaptive block partitioning for video coding," in *Proc. IEEE ICIP*, Sep. 2010, pp. 965–968.

[34] Q. Wang, M.-T. Sun, G. J. Sullivan, and J. Li, "Complexity-Reduced Geometry Partition Search and High Efficiency Prediction for Video Coding," IEEE international Symposium on circuits and systems, Seoul, Korea (South), 2012, pp. 133 – 136.

[35] Q. Wang, J. Li, G. J. Sullivan, and M.-T. Sun, "Reduced-Complexity Search for Video Coding Geometry Partitions Using Texture and Depth Data," in *Proc. Int. Conf. Vis. Commun. Imag. Process. (VCIP)*, Nov. 2011, pp. 1-4.

[36] Q.Wang, X. Ji, M.-T. Sun, G. J. Sullivan, J. Li, and Q. Dai, "Complexity Reduction and Performance Improvement for Geometry Partitioning in Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, pp. 338-352, 2013.

[37] R. Forchheimer and T. Kronander, "Image coding-from waveform to animation," IEEE Tram.Acoust. Speech, Signal Processing, vol. 38, pp. 2008-2023, 1990.

[38] G. Wolberg, "Digital image warping," Los Alamitos, CA: IEEE Computer Society Press, 1990.

[39] J. Kybic and M. Unser, "Fast Parametric Elastic Image Registration," IEEE Trans. Image Proc., vol. 12, pp. 1427- 1442, 2003.

[40] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," International Journal of Computer Vision, vol. 56, pp. 221–255, 2004.

[41] G. Sullivan and T. Wiegand, "Rate-Distortion Optimization for Video Compression," IEEE Signal Processing Magazine, pp. 74-90, 1998.

[42] M. R. Pickering, M. R. Frater, and J. F. Arnold, "Enhanced Motion Compensation using Elastic Image Registration," ICIP, 2006, pp. 1061-1064.

[43] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves,' Technical Report VCEG-M33, ITU-T SG16/Q6, Austin, Texas, USA, 2001.

https://sites.google.com/site/ijcsis/
ISSN 1947-5500

Eskinder Anteneh Ayele received his B-Tech degree in Electrical engineering from DEC, Ethiopia and M-Tech in Control and Instrumentation from IIT, Madras, India in 2001 and 2005 respectively. He was a senior lecturer and HoD in Electronics engineering department, DEC, Ethiopia from 2006 to 2011. Currently, he is a PhD scholar in Centre for VLSI and Nanotechnology, VNIT, Nagpur, India. His research interests include Signal and Image Processing, Electronic Instrumentation, and Control Systems.

Email: eskinderanteneh@yahoo.co.uk

Dr. S. B. Dhok is Associate Professor in Centre for VLSI and Nanotechnology at Visvesvaraya National Institute of Technology, Nagpur (India). He received his PhD in Electronics Engineering from VNIT Nagpur, India. He is a member of IEEE society. He has published many research papers in national and international journals and conferences. His area of interest includes Signal Processing, Image Processing, Data Compression, Wireless Sensor Networks and VLSI Design.

Email: sbdhok@ece.vnit.ac.in

https://sites.google.com/site/ijcsis/
ISSN 1947-5500