New adaptive interpolation schemes for efficient mesh-based motion estimation

H. Mahdavi-Nasab and Shohreh Kasaei

Abstract: Motion estimation and compensation is an essential part of existing video coding systems. The *mesh-based motion estimation* (MME) produces smoother motion field, better subjective quality (free from blocking artifacts), and higher *peak signal-to-noise ratio* (PSNR) in many cases, especially at low bitrate video communications, compared to the conventional *block matching algorithm* (BMA). However, the iterative refinement process of MME is computationally much costly and makes the method impractical in real- (or near real-) time systems. Also, eliminating the iterative refinement step deteriorates the motion estimation result. In this paper, we propose motion adaptive interpolation schemes for non-iterative MME, which use BMA to compute the *motion vectors* (MVs) of mesh nodes. The proposed algorithm aims at compromising the MME and BMA by modifying the *interpolation patterns* (IPPs) of the MME in an adaptive manner, based on the MVs of mesh nodes. Experimental results show notable rate-distortion improvement over both BMA and conventional non-iterative MME, with acceptable visual quality and system complexity, especially when applied to sequences with medium to high motion activities.

Keywords: Motion estimation, block matching, mesh-based analysis, low bitrate video coding, interpolation.

1 Introduction

Motion estimation (ME) and compensation has played a great role to make low bitrate video coding possible. Among the ME algorithms, the block matching algorithm (BMA) has been the most widely used approach. The BMA is accepted in existing video standards (MPEG-X, H.26X), due to its structural simplicity and low computational load [1]-[3]. However, the BMA assumes a single translational movement (parallel to the image plane) for each block, which limits its efficiency. In addition, adjacent blocks with different motions lead to sharp rectangular isolated patches (or blocking artifacts), which are especially more annoying at low bitrates.

A promising and more recent approach called mesh-based ME (MME, also called warping, or control grid interpolation), is able to realize non-translational movements (such as rotation, zooming, and moving away or towards the camera) with a smooth motion field (due to its more complex motion models) [1], [4]-[8]. Fig. 1 illustrates a comparative example of the MME and BMA. The main problems of the MME are its high computational load, inability to present motion discontinuities, difficulty in handling large motions (while preserving mesh structure), and error propagation.

Iranian Journal of Electrical & Electronic Engineering, 2005.
Paper first received 20th November 2004 and in revised from 11 July 2005.
H. Mahdavi-Nasab is a PhD candidate at the department of Electrical Engineering, Science & Research Branch, Islamic Azad University, Tehran, Iran.
Shohreh Kasaei is with the department of Computer Engineering, Sharif University of Technology, Tehran, Iran. Email: skasaei@sharif.edu.

In the backward mode of this approach, a mesh or grid of polygons (usually triangles or quadrangles) is overlaid on the current frame, and the corresponding points of the vertices of mesh patches (elements) are found in the previous (reference) frame by some ME technique, such as the BMA. To obtain the motion vectors (MVs) of the interior pixels of mesh patches, the MVs of the vertices are linearly interpolated (see Fig. 1). In the next (and much more costly) step, the MVs of the vertices are refined, by perturbing their values and computing the resulting PSNRs of the jointed patches. Due to the inter-dependence of the MVs of mesh nodes in determining the PSNRs of the patches (which implies

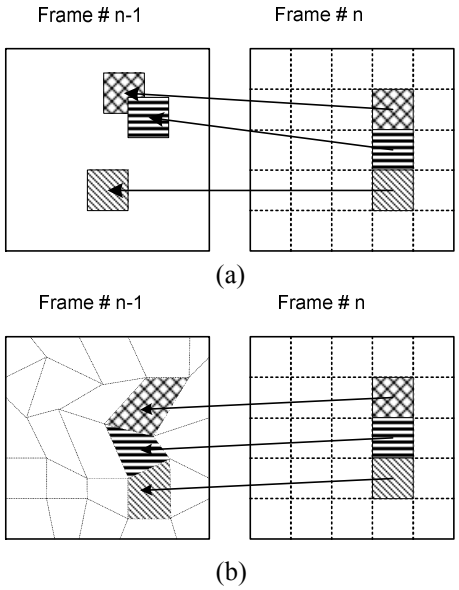

Fig. 1 Backward motion estimation: (a) Block-based, (b) Mesh-based.

a large set of possibilities; and iterations) and the need for interpolation of all pixels (at each iteration), the refinement step introduces a high computational load to the MME. This makes the method very difficult or even impossible to implement [9]. The pioneering researchers of the MME algorithm have tried to overcome this problem, but the problem is still mainly remained. Some of the strategies proposed in this regard are hexagonal matching [4], limited perturbations [10], and sub-optimal search [11]. Note that in the BMA, on the contrary, the MV of each block is determined independent from others.

On the other hand, in forward MME, the mesh is established on the reference (previous) frame, and the corresponding nodes are found in the current frame (see Fig. 2). The forward approach is deemed to be a suitable scheme for tracking objects and content-based mesh generation processes. However, for each new frame a new deformed mesh should be considered, which makes forward MME more difficult to implement. On the contrary, the advantages of the backward approach are its ease of use (for its fixed shape for each new frame), lower computation load, and ability to be integrated in standard coders which use block-based motion models (if a regular quadrilateral mesh is applied).

In addition to the computational load, another main problem of the MME is its assumption of motion smoothness and continuity. This is implied in the affine or bilinear interpolation process used for generating the interior MVs of mesh patches. This leads to the inability of MME to present multiple motions (of different objects, or an object against its background), when encountered inside the mesh elements. To overcome this problem, one solution might be the adaptation of the mesh with the scene(hierarchically or content-based). The result would be an irregular mesh with more nodes in areas of higher activities. In the hierarchical MME, the mesh is decomposed regularly to more patches. While in the content-based approach, either the mesh nodes are shifted toward the object borders (active meshes) [6]-[8], or they are deleted or added in occluded regions (occlusion adaptive meshes) [11], [12]. Clearly, this solution leads to irregular meshes and requires more sophisticated mesh structures, extra bitrates for tracking the mesh, more computation loads, and incompatibility with existing video coding systems, especially for content-based meshes. Notable improvements in visual subjective and objective quality are reported in some cases [6], [8], [12]; which justify the additional load.

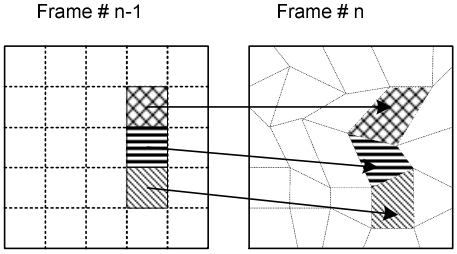

Fig. 2 Forward mesh-based motion estimation.

A simpler and more practical alternative (to overcome the inability of the MME to represent motion discontinuities) is to preserve the regular mesh and adapt the interpolation patterns (kernels) to the motion and texture of image sequences. Here, we refer shortly to two approaches of this kind.

In one approach, *Hsu et al.* [13] proposed to select the interpolation pattern (IPP) from a fixed set of 33 patterns (one default bilinear pattern and 32 others). The other patterns are defined based on quadrilateral elements divided by straight lines with different origins and angles; which approximate different possible motion boundaries. The interpolation is accomplished linearly taking two or only one node (remained in each side of the divided line) into account. If the prediction error of a patch (after motion compensation with bilinear interpolation of MME) is greater than a threshold value, other 32 IPPs are tested and then with respect to a rate-distortion criterion the best pattern is chosen. In the second step, using the chosen IPPs, the MVs of mesh nodes are iteratively refined.

In another approach, *Nosratinia* [9] suggested to compute an optimum IPP for each sequence based on an introduced generalized orthogonality condition. The optimum pattern is used in a fast (non-iterative) MME framework with the MVs of mesh nodes being found by an exhaustive BMA approach. The patterns depend on the statistics of sequences and are computed using the first 50 frames of each sequence. It is observed that the optimal IPP varies significantly from a sequence to another and is often very different from the conventional bilinear pattern. It has been shown experimentally that (contrary to the conventional MME), these new patterns mostly lead to better results than the BMA in terms of PSNR without any refinement process even when used in their simplified parametric form.

In this paper, we extend the adaptive interpolation ideas in a fast MME framework by designing new adaptive schemes. Especially we introduce motion adaptive interpolation algorithms, which are capable of enhancing the motion prediction accuracy in most cases with no extra bitrate and noticeable complexity. The power of these methods lies in their ability to handle large MVs and preventing error propagation. Here, to reduce the computational cost, we use backward prediction with regular quadrilateral or triangular meshes with integer-pixel accuracy. The tests are run on some typical QCIF[1] standard sequences to emphasize on low bitrate applications. The experimental results show the efficiency of the proposed algorithms.

2 Some basic preliminaries

A. Mesh structure

Fig. 3-a shows the typical structure of a regular quadrilateral mesh (often used in the MME algorithms). Consider a frame with M×N tiles. For BMA, there

[1] *Quarter Common Intermediate Format* (QCIF)

would be M×N MVs per frame. For MME, this becomes (M+1)×(N+1) for the (M+1)×(N+1) mesh nodes. This causes about %10 higher bitrates when applying on CIF[1] (288×352) videos with 16×16 blocks, and about %20 higher bitrates for QCIF (144×176) videos; and even more difficult to compensate for lower bitrate applications. Besides, clearly the nodes on the frame borders, which are 2(M+N) nodes, are not suitable candidates for the BMA motion computation. In [5], it is suggested to ignore these border nodes to save some bitrate. In cases without any camera pan or object movements near the borders of the frame this idea works well but otherwise some useful information will be lost (resulting up to about 1dB decrease in PSNR in some cases). Also, in the *hierarchical mesh-based matching algorithm* (HMMA) method proposed in [10], the right and bottom border nodes are dropped.

For a better compromise between the BMA (as the ME method for mesh nodes) and the MME, we suggest the mesh shown in Fig 3-b. Here, the centers of the BMA blocks are considered as mesh nodes. The bitrate is equal to the BMA and the motion prediction is more efficient for the outermost nodes. The MVs inside the non-complete patches connected to the frame borders are computed by the same MME IPPs as in the interior patches; with taking the two (or one) remained nodes into account. Actually, the *hierarchical block-based matching algorithm* (HBMA) method suggested in [10] applies such a mesh placement and (naturally) produces better results than the other MME version (i.e., HMMA).

Adding diagonals to the proposed mesh quadrilaterals generates the triangular meshes that we use in this paper (see Fig. 4). The direction of the diagonals does not lead to a deterministic significant result in different cases. Although equilateral triangles have been reported to be somewhat more efficient [4], again the consistency of the proposed triangular mesh with the BMA justifies this selection.

B. Interpolation patterns

In the conventional MME, the MVs of interior pixels of mesh patches are interpolated using affine or bilinear patterns (for triangular or quadrilateral meshes), respectively. These patterns produce a uniform change of MVs just from the vicinity of mesh nodes. As proposed in [9], many other IPPs can be considered in a

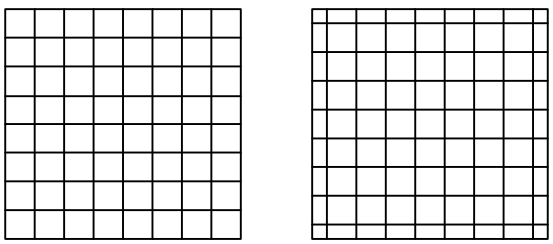

Fig. 3 Quadrilateral mesh structures, (a) conventional, (b) proposed; mesh nodes are the centers of blocks in (a).

[1] *Common Intermediate Format* (CIF)

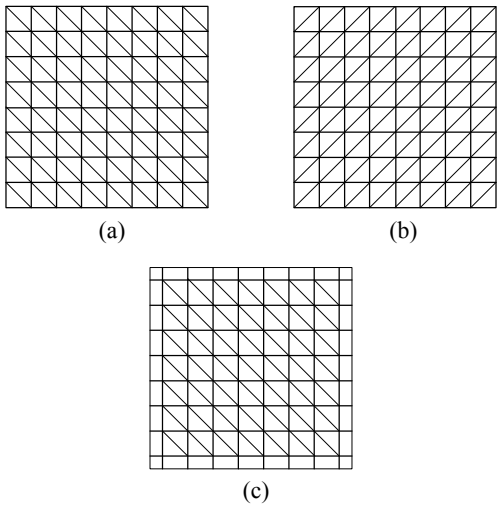

Fig. 4 Various right-angled triangular meshes: (a) and (b) conventional *right* and *left triangular*, (c) proposed *right triangular* mesh.

Spectrum from quite linear to BMA resembling discontinuous forms. Here, we introduce semi-linear or *medium* (MED), *near-block-matching* (NBM), and *block-matching* (BM) IPPs which are used in our proposed adaptive MME algorithms.

Considering Fig. 5, the MV of pixel *s*, d(s), inside the patch can be computed as:

$$d(s)=\sum_{i,j=1}^{2} h_{k,ij}(s).d_{ij} \qquad (1)$$

where $h_{k,ij}$ and d_{ij} are the 2-D IPPs, and the MVs of patch nodes, respectively.

Our suggestion for non-linear IPPs is based on the modified sigmoid function defined as:

$$h_k(x)=\frac{1}{1+\exp[k(x-0.5)]}\left(1+\frac{0.1}{(k-5)^2}-\frac{0.2x}{(k-5)^2}\right)$$

$$x\in(0,1)\ ,\ k\geq 6$$

$$h_k(0)=1\ ;\ h_k(1)=0\ . \qquad (2)$$

These are shown in Fig. 6 for k = 10, 20, an d 200, respectively, compared to the bilinear pattern in its 1-D form, i.e.:

$$h_l(x)=1-x\ ;\ \ x\in[0,1]\ . \qquad (3)$$

The 2-D patterns, h (x, y), used in (1) are:

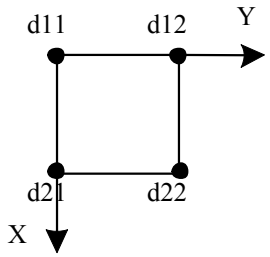

Fig. 5 Symbols used for motion vectors of a patch.

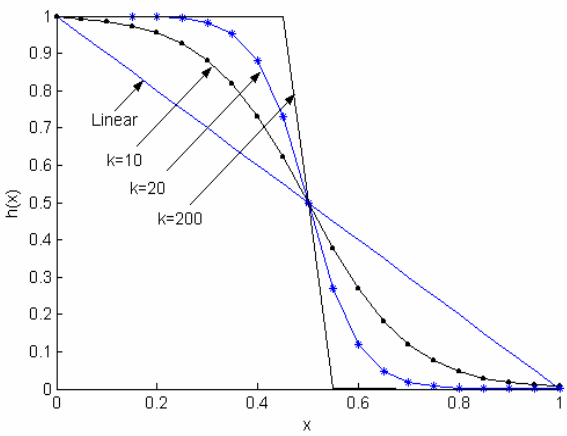

Fig. 6 Various 1-D interpolation patterns.

$$h_{k,11}(x,y) = h_k(x) \cdot h_k(y)$$

$$h_{k,12}(x,y) = h_k(x) \cdot (1 - h_k(y))$$

$$h_{k,21}(x,y) = (1 - h_k(x)) \cdot h_k(y)$$

$$h_{k,22}(x,y) = (1 - h_k(x)) \cdot (1 - h_k(y)) \qquad (4)$$

In which we have assumed dimensions of patches to be normalized to one. In fact, the IPPs introduced by (2) and (4) are similar to those defined in [9] as the parametric patterns, but with simpler closed forms.

We define the IPPs of the MED, NBM, and BM patterns as $h_k(x,y)$, and substitute k in (4) by 10, 20, and 200, respectively.

Note that the MED pattern is more *adhesive* than linear to the closer mesh node. For the NBM and BM, the *adhesion* is more and more. On the contrary, further pixels in relation to a mesh node get more and more freedom in accepting the MV of the node.

Clearly, using the BM IPP in the MME will convert it to the BMA. In other words, the BMA can be considered as a special case of the MME. In fact, moving from the bilinear interpolation toward the BM pattern is equal to proceeding from a conventional linear MME to a strict BMA (and applying patterns like the MED and NBM would be a compromise between these two).

Similarly, in addition to the known affine IPP for triangular meshes [1], [4], [7], [8], we have defined *Semi-affine* patterns, which are more similar to the BM pattern. A form of such patterns is introduced in the Appendix, in its matrix form.

3 Proposed adaptive interpolation mesh-based motion estimation algorithms

As mentioned above, the main problem of the conventional MME is its disability to deal with motion discontinuities. When computing the motion of interior pixels of mesh patches by the linear interpolation of the motions of mesh nodes, it assumes that the motion is continuous and changes smoothly. In addition, large motion differences between adjacent mesh nodes may collapse the geometrical structure of the mesh. As a matter of fact, image sequences with medium or high motion variances often cannot be handled efficiently by the MME method. Consequently, in this work, the IPPs are adjusted to the scene motion content to overcome these shortcomings.

As the motion features of an image sequence are not fixed (neither in time, nor in the spatial dimensions), seeking one optimum IPP (as done in [9]) results in a global answer which can be quite different from the local desired patterns. In other words, such a unique pattern may be optimum only in the mean sense, and not necessarily for every patch of a frame or all of the frames. Also, estimating such an optimum solution often requires a notable computational load and large number of training frames.

On the other hand, finding optimum IPPs for each mesh element of a frame is not a trivial task either; especially when the election is based on direct examining of some defined patterns. In addition to its high computational burden, the required overhead may compensate the achieved compression rate (especially in low bitrate applications).

On the other hand, decreasing the number of candidate patterns and patches to be examined (only patches with prediction errors more than a specified value are tested by other patterns), will reduce the computational load (in proportion to the restriction made), but clearly the estimation quality will also deteriorate.

To choose a suitable IPP for each mesh element from a restricted list (without computing the errors of each one), here we suggest making the decision based on an investigation of the MVs of nodes of each patch. Since these MVs are supposed to be already known in the coder and decoder, such a method will *not* require any overhead. We call this method *motion adaptive MME* (MAMME). In this work, we have designed and examined several MAMME algorithms among which two alternatives are stated below. Consider Fig. 5, for the nomination of the MVs of a mesh element. Fig.7 shows a block-diagram of the first proposed method (quadrilateral MAMME).

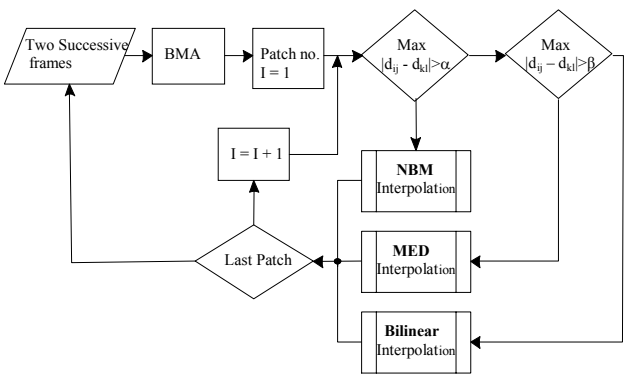

Fig. 7 Schematic block diagram of the proposed Q-MAMME method, $\alpha \geq \beta$; i,j,k,l ∈ {1,2}.

In Fig. 7 the processing of the patches is shown to be serial. However, as there is no dependency among them, the computation of the MVs of different patches can be performed a parallel.

A. Quadrilateral MAMME

To implement the proposed *quadrilateral MAMME* (Q-MAMME) algorithm do as follows.

1- Perform the BMA to estimate the MVs of mesh nodes ($\vec{d}=(d_x,d_y)$).

2- For each mesh element, $\forall i,j,k,l\in\{1,2\}$,

2.a) If $\{\max_{i,j,k,l}|dij_x-dkl_x|\geq\alpha$, or $\max_{i,j,k,l}|dij_y-dkl_y|\geq\alpha\}$,

Perform the NBM interpolation on the nodal MVs to determine MVs of inter-element pixels.

2.b) Else, if $\{\max_{i,j,k,l}|dij_x-dkl_x|\geq\beta$, or $\max_{i,j,k,l}|dij_y-dkl_y|\geq\beta\}$, ($\beta\leq\alpha$)

Perform the MED IPP.

2.c) Otherwise, apply bilinear interpolation.

B. Square to triangular MAMME

To implement the proposed *triangular MAMME* (T-MAMME) algorithm do as follows.

1- Perform the BMA to estimate the MVs of mesh nodes ($\vec{d}=(d_x,d_y)$).

2- For each quadrilateral mesh element,

2.a) If $\{|d11_x-d22_x|\geq\alpha$, or $|d11_y-d22_y|\geq\alpha\}$,

Break the quadrilateral to two triangles, by plotting the diagonal from top-right to down-left node (*left-triangular* element*)*, and,

2.a.1) If $\{|d11_x-d22_x|\geq\beta$, or $|d11_y-d22_y|\geq\beta\}$, ($\alpha\leq\beta$)

Perform *Semi-affine* interpolation on the nodal MVs to estimate the MVs of the internal pixels of the element.

2.a.2) Otherwise, perform affine interpolation to estimate the interior pixels motions.

2.b) Else, if $\{|d12_x-d21_x|\geq\alpha$, or $|d12_y-d21_y|\geq\alpha\}$,

Break the quadrilateral to two triangles, by plotting the diagonal from top-left to down-right node (*right-triangular* element), and,

2.b.1) If $\{|d12_x-d21_x|\geq\beta$, or $|d12_y-d21_y|\geq\beta\}$,

Perform *Semi-affine* interpolation.

2.b.2) Otherwise, perform affine interpolation.

2.c) Else, if $\{\max_{i,j,k,l}|dij_x-dkl_x|\geq\lambda$, or $\max_{i,j,k,l}|dij_y-dkl_y|\geq\lambda\}$, ($\forall i,j,k,l\in\{1,2\}$)

Perform the MED pattern interpolation.

2.d) Otherwise, for all other mesh elements, perform bilinear interpolation to determine the interior pixel motions.

Among these two, the first approach (which is also the more straightforward algorithm) leads to better results in most cases. In step 2.a of this algorithm the NBM pattern can be replaced by the BM. The reason to introduce the second method is to show various possibilities in motion adaptive algorithms, and also to compare the triangular and quadrilateral MME.

In the conventional MME with BMA as the estimator of its nodal motions, the main possible sources of errors are: I) the existing motion boundaries in mesh elements, and II) the BMA errors in determining the nodal MVs, which can be caused by occlusion (especially uncovered background in backward tracking), limited search range, or noise.

As for the first case, it is natural to assume that sufficiently large differences between the MVs of an element is caused by either a moving object against a stationary background, or by two objects with different MVs. And when the differences are relatively larger, the mentioned assumption would be closer to reality. Thus, in such cases, a linear interpolation is chosen. Here, we change the IPP to increase its ability to handle motion discontinuity, by making it more similar to a strict BM type pattern. Applying the NBM (or BM) and MED patterns in the Q-MAMME and also the MED and *Semi-affine* patterns in the T-MAMME, in cases where the differences among the MVs of element nodes are significant, realizes this idea.

Considering the second case, proceeding from the bilinear to the BM IPP (or dividing the patches), prevents from error propagation; which is a serious problem in the MME. (in general, errors produced by any means tend to propagate through the interpolation process.) Increasing the *adhesion* of the pattern to the centers of the BMA blocks prevents the incorrect MVs to affect the correctly estimated regions. Regarding the sources of the BMA errors, it seems reasonable to assume that their results have a uniform distribution. Thus, there is a small probability for the incorrect MVs to be the same or similar to the MVs of their adjacent nodes; which justifies the proposed algorithms.

C. Dual interpolation MME

In addition to the introduced Q-MAMME and T-MAMME methods, we have designed another adaptive interpolation MME algorithm, called *dual interpolation MME* (DMME), for comparison purposes. In this method, the estimation errors of two different IPPs are directly computed and compared for each frame (a

unique pattern is used for the motion estimation and compensation of all mesh elements of a frame). Then, a code (zero or one) indicating the pattern with lower error is transmitted to the decoder, along with the motion field of each frame. The selected patterns are the bicubic (close to bilinear) and the NBM.

Clearly, the main drawbacks of this method, compared to the MAMME schemes, are its required overhead bitrate and higher computational cost. The overhead is insignificant (only one bit per frame) and the reason for using only two IPPs is to avoid increasing the computational cost.

D. Computational cost

Computationally, the ME process is one of the most costly stages of the existing video coding systems (even when using the simplest method, namely BMA). Clearly, the computational burden of iterative MME is much more than the BMA. As shown in [9], considering some typical assumptions (*e.g.,* 3 or 4 iterations and 16×16 blocks), the computational cost of the conventional iterative MME is in the order of more than 100 times of the BMA. This makes the implementation of the MME to be practically unreasonable. However, without the iterative refinement step, the computational cost of the MME in the coding system is equal to the BMA. Note that in the fast MME, the motion field is completely determined by the MVs of mesh nodes, which are also the only transmitted MVs to the decoder. To compute the dense motion field, the interpolation of the MVs for inter-element pixels is required. Also, as the interpolated MVs are not necessarily integer-valued, the intensity of the pointes within the pixels in the reference frame may need to be interpolated. In the worst case, this requires an additional 8 multiplications, and 7 additions for each pixel. The computational cost of the multiplications depends on the used word length. Assuming 8-bit word lengths and QCIF videos, the additional computational cost for our MAMME methods is only about 40% of the full search BMA (for 7 or 8 pixels search range).

The real computational complexity is much less than the above estimations. For instance, if the MVs of the nodes of an element happen to be equal (which occurs in most cases in low bitrate applications), the MVs of the element pixels would all be equal, and no multiplications or additions are required. Also, considering the proposed IPPs, with more ones and zeros in their multiplying coefficients (e.g., NBM and especially BM IPP), many of these operations will be eliminated.

4 Experimental results

To give the experimental results and the performance analysis we first define the used parameters and resources.

A. Parameters and resources

The results presented here are obtained by using a number of QCIF (144×176) MPEG standard sequences at 25 fps, with emphasis on human *face* and *head and shoulder* videos. In the presented tables, frames 51~83 of *Mother-and-Daughter* sequence (where the mother's hand enters the scene) and also frames 41~73 of *Suzie* sequence are selected as sequences with medium and high motion content, respectively. Also, low motion videos such as *Akiyo* sequence, and initial frames of *Suzie* sequence are used. It might be interesting to note the quite different motion statistics in one sequence (*e.g., Suzie*), requiring different attitudes. In addition to 16×16 blocks, used in the experiments listed in Tables 1, 2 and 3, we have shown the results of applying 8×8 blocks (odd frames), for higher bitrate applications in Table 4.

In all these experiments the exhaustive BMA is used for motion estimation of mesh nodes, with ±7 pixels search range, and an integer-pel precision. The MME algorithms are all one step versions with no refinements (fast MME). The QMME and TMME methods use quadrilateral and *right-triangular* meshes, with bilinear and affine IPPs, respectively. In the Q-MAMME algorithm, for 16×16 blocks, α and β are chosen to be 6 and 3, respectively. In the T-MAMME algorithm, α, β, and λ are set to 4, 6, and 3, respectively. When the blocks are 8×8, α and β in Q-MAMME are equal to 4 and 2 (in this case the NBM is replaced by the BM in step 2.a), and α, β, and λ in the T-MAMME are chosen to be 3, 5, and 3, respectively. For simplicity the defined patterns are computed up to at most two significant digits.

In Fig. 8, we have compared the performance of the discussed ME methods when applied on *Suzie* sequence, with 16×16 blocks (as a good example, containing different motion features in its different parts). Fig. 9 also compares the performance of the ME methods, when applied to the first 40 frames of *Foreman*, with 8×8 blocks, (*i.e.,* in higher bitrates). Fig. 10, provides a subjective demonstration of the compensation quality of the Q-MAMME method when compared to the BMA and the QMME, by applying them to frames 11 and 13 of *Foreman* sequence.

B. Performance analysis

As shown in Fig. 8, the proposed MAMME and QMME methods are clearly superior to the BMA in terms of PSNR (in some cases near 2 dB) in the initial low-motion frames of *Suzie* sequence. In the high-motion parts (from frame 41), where the QMME fails, the proposed adaptive methods, especially the Q-MAMME, still remain superior (up to 1dB in many frames), compensating the problems of the QMME. Fig. 9 shows similar results when running the QMME algorithm on *Foreman* sequence, at a higher bitrate. While Q-MAMME performs always better than other schemes, the BMA and the fast MME methods often commute their places in the PSNR diagram.

The visual superiority of the proposed Q-MAMME method to the BMA and QMME, is evident in Fig. 10.

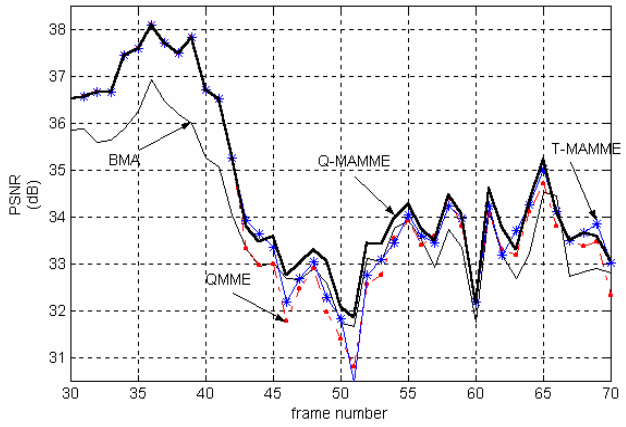

Figure 8 Comparing the quality of various motion estimation methods applying on *Suzie* (with 16×16 blocks)

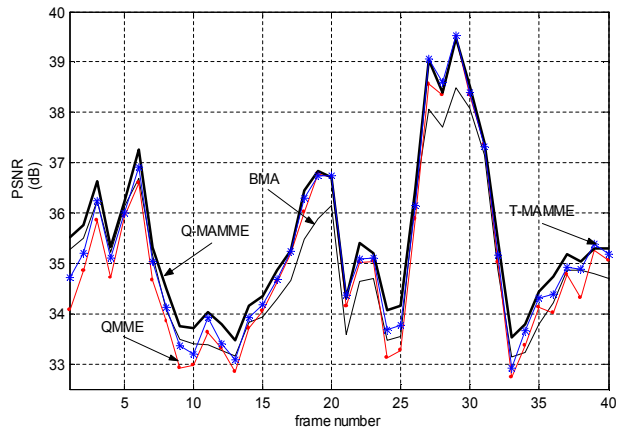

Fig. 9 Comparing the quality of various motion estimation methods applying on *Foreman* (with 8×8 blocks)

No blocking artifacts are present (which are easily seen in the BMA compensated frame, Fig. 10-c), and the residual error is clearly reduced (see Figure 10-f, g, h). In addition, comparing the numerical results of the aforementioned tables and figures, some interesting results can be deduced. We have summarized these as follows:

1- In lower bitrates (16×16 blocks), nearly always the QMME algorithm performs better than (or at least equal to) the BMA. In Table 1, the main exception is the high motion part of *Suzie* (frames 41~83), and to a less extent the medium motion *Mother-and-Daughter* (frames 51~93). When the motion content decreases, by reducing the distance between frames, the QMME performs even better. In Table 2, the only exception is *Suzie* (frames 41~83), and in Table 3 (with the least distance between frames, and thus the least motion content), the QMME is superior to the BMA in all cases. It seems to have a law that, "*the less the motion content, the higher the quality of the fast MME in comparison to the BMA, in terms of PSNR*". Figure 8 confirms this statement clearly.

2- In higher bitrates, on the contrary, nearly always the QMME performs worse than the BMA. In *Suzie* (41~83), with 8×8 blocks, the BMA is more than 1.2 dB better than the QMME, in terms of PSNR (Table 4). The only exceptions are the very low motion videos, *Akiyo* and the first part of *Suzie*. Note that increasing the dimensions of blocks (from 8×8 in Table 4, to 16×16 in other cases), makes the mesh geometrical structure more immune to large motion differences (between the mesh elements nodes), which tends to deform the mesh elements (see Fig. 1-b). With ±7 pixels search range (thus possible differences of at most 14 pixels between the MVs of an element nodes), 8×8 quadrilateral elements are more likely to degenerate, or completely collapse (when the difference among the MVs is greater than 8), than 16×16 elements. This can explain this observation, which leads to another law, "*fast MME performs better in lower bitrates*". For higher bitrates, or higher motion contents, the MAMME methods provide an efficient answer. Fig. 8 shows an example for these cases.

3- As expected, the Q-MAMME and the T-MAMME methods perform better than (or at least equal to) the MME (QMME or TMME), in all cases. Between these two, the Q-MAMME performs better (as is the case for the QMME in relation to the TMME). It may be because of its equilateral polygons [4], and its quite compatibility with the geometrical structure of the BMA. Contrary to the MME, which in some cases performs inferior to the BMA in terms of PSNR, the Q-MAMME always performs better, adding an advantage to the better visual quality of the MME family of algorithms (see Figure 10 as an example). The superiority of the Q-MAMME to the conventional MME is especially observed in cases with large MVs, relative to block dimensions. In *Mother-and-Daughter* (51~83), with 8×8 blocks, this superiority reaches to near 1 dB in average, and in *Suzie* it is about 1.5 dB (see Table 4). When the motion content is negligible, the MAMME reduces to the MME, and therefore the superiority is also negligible. Note that in these cases the MME performs better than the BMA (Figure 8 also provides a good example). As the bitrate and the computational cost of the QMME and Q-MAMME are quite the same (with the difference only in the IPPs) the higher efficiency of the Q-MAMME is justified.

4- Finally, as the transmitted MVs of all the compared ME algorithms are alike (the BMA MVs of mesh nodes), the bitrate of motion is the same for all these methods, except for the DMME which has an overhead equal to one bit per frame. The higher quality (in the sense of mean-squared error) of our proposed schemes also is expected to lead to lower bitrates for residual errors and therefore we expect an overall improvement in the rate-distortion function. The results of DMME are also better than the QMME and BMA in most cases, and the bitrate overhead is quite insignificant. This shows a promising approach with the computational load as the only drawback. The coder has to compute the errors of two different IPPs to choose the

Table 1 Mean PSNRs (in dB) of various ME algorithms for some standard QCIF sequences with 16×16 blocks (frames 1, 4, 7, …).

SEQ. / ALG.	Foreman (1~43)	Carphone (1~43)	M&D (51~93)	Akiyo (1~43)	Suzie (1~43)	Suzie (41~83)
BMA	28.94	31.44	33.18	38.69	34.82	27.34
QMME	28.93	31.65	33.10	39.44	35.81	26.63
TMME	28.83	31.64	33.11	39.40	35.74	26.60
DMME	29.20	32.08	33.43	39.49	35.79	27.34
T-MAMME	29.17	31.97	33.30	39.47	35.86	26.86
Q-MAMME	29.33	32.16	33.51	39.47	35.86	27.44

Table 2 Mean PSNRs (in dB) of various ME algorithms for some standard QCIF sequences with 16×16 blocks (frames 1, 3, 5, …).

SEQ. / ALG.	Foreman (1~33)	Carphone (1~33)	M&D (51~83)	Akiyo (1~33)	Suzie (1~33)	Suzie (41~73)
BMA	30.57	31.80	34.59	41.42	36.11	29.27
QMME	30.91	32.03	34.75	42.01	36.95	28.80
TMME	30.86	31.92	34.77	41.97	36.85	28.81
DMME	31.01	32.38	35.00	42.03	36.90	29.41
T-MAMME	31.00	32.50	34.91	42.01	36.95	29.03
Q-MAMME	31.12	32.63	35.04	42.01	36.95	29.52

Table 3 Mean PSNRs (in dB) of various ME algorithms for some standard QCIF sequences with 16×16 blocks (frames 1, 2, 3, …).

SEQ. / ALG.	Foreman (1~33)	Carphone (1~33)	M&D (51~83)	Akiyo (1~33)	Suzie (1~33)	Suzie (41~73)
BMA	33.31	32.81	38.09	44.85	38.23	33.21
QMME	33.86	33.10	38.30	45.03	38.61	33.32
TMME	33.78	32.97	38.30	45.01	38.57	33.26
DMME	33.92	33.40	38.53	45.02	38.58	33.63
T-MAMME	33.96	33.39	38.41	45.03	38.61	33.54
Q-MAMME	34.02	33.54	38.53	45.03	38.61	33.73

Table 4 Mean PSNRs (in dB) of various ME algorithms for some standard QCIF sequences with 8×8 blocks (frames 1, 3, 5, …).

SEQ. / ALG.	Foreman (1~33)	Carphone (1~33)	M&D (51~83)	Akiyo (1~33)	Suzie (1~33)	Suzie (41~73)
BMA	32.92	33.32	36.14	42.47	36.95	31.83
QMME	32.60	33.20	35.58	42.85	37.64	30.56
TMME	32.53	33.02	35.47	42.74	37.60	30.46
DMME	33.10	33.55	36.25	42.90	37.62	31.58
T-MAMME	32.79	33.50	36.03	42.86	37.66	30.80
Q-MAMME	33.34	33.78	36.51	42.88	37.66	32.02

more proper one. It is interesting to note the superiority of the Q-MAMME against the DMME in most cases. But increasing the number of IPPs in the DLMME may reverse the situation.

5 Conclusion

In this paper, we introduced fast adaptive MME algorithms that overcome the shortcomings of the MME (such as, representing motion discontinuities and error propagation), by adapting the IPP to the motion features of the scene. The proposed Q-MAMME and T-MAMME schemes do not require any additional overhead bitrate, since the choice of IPP for each mesh element depends only on the MVs of mesh nodes which are assumed to be available in decoder. Also, as the iterative refinement steps of the conventional MME are eliminated, the computational tasks of the coder are decreased to a great extent.

The presented results show that the proposed MAMME performs better than both the fast MME and BMA. The MAMME can be seen as an effort to make the MME more compatible with its initial estimator (which is the BMA in our study) and contains the advantages of both MME and BMA. The designed mesh structure and the defined IPPs, are supposed to realize this idea. Especially, the superiority of the MAMME methods to the conventional fast MME is quite significant in cases of moderate to high motion activities, where the MME actually fails.

Sub-pixel accuracy ME when applied to low motion sequences generally results in considerable improvement in PSNR; while requiring higher bitrates. Our preliminary experiments with half-pel accuracy ME methods show somewhat different results, especially when considering various up-sampling interpolation filters [3], [14]. This necessitates more complex adaptive schemes, and is an aspect of our future work on the subject. Also, hierarchical methods are considered in the next stages of this study.

6 References

[1] A.M. Tekalp, *Digital video processing* , Prentice-Hall, 1995.

[2] T. Sikora, "The MPEG4 video standard verification model", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 19-31, Feb. 1997.

[3] T. Wiegand, G. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard", *IEEE Trans. Circuits Syst. Video Technol*., vol. 13,pp. 560-576, July 2003.

[4] Y. Nakaya and H. Harashima, "Motion compensation based on spatial transformations", *IEEE Trans. Circuits Syst. Video Technol*., vol. 4, pp. 339-356, June 1994.

[5] C. L. Huang and C. Y. Hsu, "A new motion compensation method for image sequence coding using hierarchical grid interpolation", *IEEE Trans. Circuits Syst. Video Technol*., vol. 4, pp. 72-85, Feb. 1994.

[6] Y. Wang and O. Lee, "Active mesh-A feature seeking and tracking image sequence representation scheme", *IEEE Trans. Circuits Syst. Video Technol*., vol. 3, pp. 610-624, Sept. 1994.

[7] Y. Wang and O. Lee, "Use of two-dimensional deformable mesh structures for video coding, Part I-The synthsis problem: Mesh based function

approximation and mapping", *IEEE Trans. Circuits Syst. Video Technol*., vol. 6, pp. 636-646, Dec. 1996.

[8] Y. Wang and O. Lee, "Use of two-dimensional deformable mesh structures for video coding, Part II-The analysis problem and a region-based coder employing an active mesh representation", *IEEE Trans. Circuits Syst. Video Technol*., vol. 6, pp. 647-659, Dec. 1996.

[9] A. Nosratinia, "New patterns for fast mesh-based motion estimation", *IEEE Trans. Circuits Syst. Video Technol*., vol. 11, pp. 40-51, Jan. 2001

[10] Y. Wang and J. Osterman, "Evaluation of mesh-based motion estimation in h.263 like coders", *IEEE Trans. Circuits Syst. Video Technol*., vol. 8, pp. 243-252, June 1998.

[11] P. Beek, A. Tekalp, N. Zhuang, I. Celasun and M. Xia, "Hierarchical 2-D mesh representation, tracking, and compression for object-based video", *IEEE Trans. Circuits Syst. Video Technol*., vol. 9, pp. 353-369, March 1999.

[12] Y. Altunbasak and A. Tekalp, "Occlusion-adaptive content-based 2-D mesh design and tracking for object-based coding", *IEEE Trans. Image Processing*, vol. 6, pp. 1270-1280, Sept. 1997.

[13] P. Hsu, K. J. R. Liu and T. Chen, "A low bitrate video codec based on two-dimensional mesh motion compensation with adaptive interpolation", *IEEE Trans. Circuits Syst. Video Technol*., vol. 11, pp. 111-117, Jan. 2001.

[14] T. Wedi and H.G. Musmann, "Motion and aliasing compensated prediction for hybrid video coding", *IEEE Trans. Circuits Syst. Video Technol*., vol. 13, pp. 577-586, July 2003.

Appendix

Semiaffine Interpolation Pattern

The elements of the triangular meshes also can be considered to be quadrilaterals that are divided in two parts by their diagonals, and thus have two different IPPs (one for the right angle, isolated vertices, and the other for the two common vertices of the two triangles, see Fig. 11).

Similar to the MED and NBM patterns defined for quadrilateral meshes, the SAM and CAM matrices can be defined for a more block-matching resembling

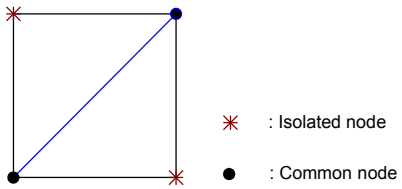

Fig. 11 Isolated and common nodes of a triangular element, made by dividing a quadrangle by its diagonal.

pattern (than the affine pattern) in the triangular case, as shown below.

For the top-left node, considering the diagonal to be from the top-right to down-left (*left triangular*), the matrix form of the multiplying pattern (for an 8×8 block) is:

$$SAM = \begin{bmatrix} 1.000 & 1.000 & 0.857 & 0.714 & 0.286 & 0.143 & 0.000 & 0.000 \\ 1.000 & 0.857 & 0.714 & 0.286 & 0.143 & 0.000 & 0.000 & 0.000 \\ 0.857 & 0.714 & 0.286 & 0.143 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.714 & 0.286 & 0.143 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.286 & 0.143 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.143 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$$

And for the top-right node of the common nodes of the two triangles created this way, we have the pattern:

$$CAM = \begin{bmatrix} 0.000 & 0.000 & 0.143 & 0.286 & 0.714 & 0.857 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.143 & 0.286 & 0.714 & 0.857 & 1.000 & 1.000 \\ 0.000 & 0.000 & 0.143 & 0.286 & 0.714 & 0.857 & 0.857 & 0.857 \\ 0.000 & 0.000 & 0.143 & 0.286 & 0.714 & 0.714 & 0.714 & 0.714 \\ 0.000 & 0.000 & 0.143 & 0.286 & 0.286 & 0.286 & 0.286 & 0.286 \\ 0.000 & 0.000 & 0.143 & 0.143 & 0.143 & 0.143 & 0.143 & 0.143 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{bmatrix}$$

For the common down-left, and the single down-right nodes, 180-degrees rotated version of the CAM and SAM are used, respectively.

If the mesh is a *right triangular* (with quadrilateral diagonals connecting top-left node to down-right one), the SAM and CAM patterns are rotated by 90-degrees clockwise.

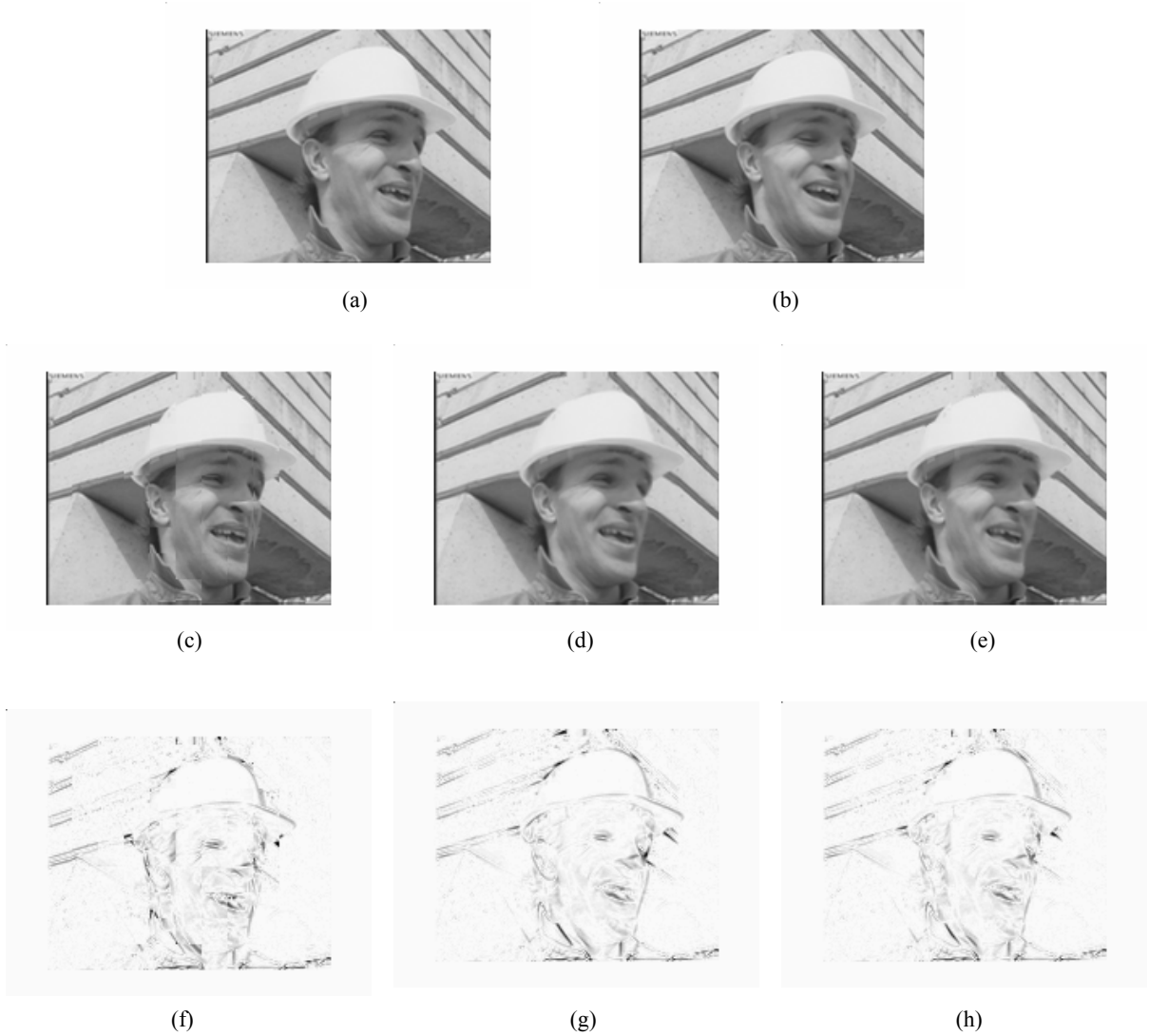

Fig. 10 Performance comparison of various ME methods on *Foreman,* compensating frame 13, based on reference frame 11, with 16×16 blocks. (a) Original frame 11, (b) original frame 13, compensated frames applying (c) BMA, (d) QMME, (e) Q-MAMME, and residual errors of (f) BMA, (g) QMME, and (h) Q-MAMME. The PSNRs are 29.7, 29.85, 30.37 dB respectively.