Hierarchical Motion Estimation With Content-Based Meshes

Ghassan Al-Regib, *Member, IEEE*, Yucel Altunbasak, *Senior Member, IEEE*, and Russell M. Mersereau, *Fellow, IEEE*

***Abstract*—Two-dimensional mesh-based models provide a good alternative to motion estimation and compensation. The estimation of the best node-point motion vectors is a challenging task. To this effect, Nakaya and Harashima proposed a hexagonal matching procedure. Toklu *et al.* improved the hexagonal search algorithm in terms of both motion-estimation accuracy and computational complexity by employing a hierarchy of regular meshes. Recognizing the limitations of regular meshes, Van Beek *et al.* extended Toklu *et al.*'s work by utilizing content-based meshes. Here, we provide an alternative hierarchical motion-estimation method with content-based meshes where hierarchical representations are employed for both the images and the irregular meshes in order to provide further improvements in computational complexity as well as motion accuracy. The comparison results are provided with real video sequences.**

***Index Terms*—Hierarchical motion estimation, model-based coding, two-dimensional mesh, two-dimensional mesh-based motion representation, video compression.**

I. Introduction

TWO-DIMENSIONAL (2-D) dense motion-estimation methods can be classified as block-based, optical-flow equation-based, pel-recursive, and Bayesian methods [1]. Although, block-based dense motion compensation yields high peak signal-to-noise ratio (PSNR), the corresponding dense motion field usually contains several outliers (because of the aperture problem and lack of overall smoothness constraints). Optical-flow equation-based methods yield smoother dense motion fields, but they do not perform well in terms of PSNR. Joint Bayesian motion-estimation and segmentation methods have the ability to provide both piecewise smooth motion fields and excellent PSNR performance. However, they are generally time consuming and parameter dependent.

2-D mesh-based models provide a good alternative to motion estimation and compensation, as reported by several researchers [2]–[15]. They are simple enough for fast implementations (especially with the help of graphics cards), but powerful enough for describing the motion content accurately.

In 2-D mesh-based methods, motion compensation within each mesh element (patch) is accomplished by a spatial transformation (affine, bilinear, etc.), whose parameters can be computed from node-point motion vectors. Estimation of motion vectors around each node independently (e.g., by block-matching or deformable block matching) is usually not desirable because: 1) motion vectors do not represent the entire 2-D dense motion field and 2) motion vectors may cross each other (especially around small patches), destroying the connectivity of the mesh. To this effect, search-based solutions to node-point motion estimation with triangular and/or quadrilateral meshes and spatial transformations have been proposed [2], [3], [7]–[11]. In particular, Nakaya and Harashima [2] proposed a hexagonal matching procedure where the motion vector at a node point is estimated by iterative local minimization of the prediction error. Toklu *et al.* [16] extended this method by employing a hierarchy of regular meshes such that motion estimation with a coarse mesh provides initialization for the next (finer) level of the mesh. However, both Nakaya and Harashima and Toklu *et al.* utilized regular meshes. Regular meshes are obtained by dividing the image area into equal size triangular or rectangular elements, hence, they may not have the ability to reflect the scene content, i.e., a single mesh element may contain multiple motions. To this effect, hierarchical quad-tree meshes are proposed in which patches that yield high motion-compensation error are successively subdivided. A more fundamental approach to overcome the problem of mesh elements with more than one motion is to employ a content-based mesh, which is not limited to a pseudoregular hierarchical structure. Content-based meshes aim to match boundaries of patches with important scene features [3], [4]. In [4], Altunbasak and Tekalp proposed a computationally efficient algorithm for content-based 2-D triangular mesh design. Van Beek *et al.* [17] extended Toklu *et al.*'s work by employing coarse-to-fine content-based meshes in motion estimation. However, although both Toklu *et al.*'s and Van Beek *et al.*'s work employ coarse-to-fine meshes, the image/frame size is kept the same, i.e., the number of nodes at each level of the mesh hierarchy changes while all meshes are *still* laid on the *original* image since no hierarchical image representation is employed. Here, we provide an alternative hierarchical motion-estimation method with content-based meshes that is superior to the aforementioned method in terms of both performance and complexity. A comparison among various 2-D mesh-based motion-estimation methods is summarized in Table I.

Section II briefly describes the hierarchical method proposed by Van Beek *et al.* [17], while the proposed method is described in Section III. Section IV describes the coarse-to-fine and fine-to-coarse mesh-generation algorithms employed in the two implementations of the proposed method, respectively. The results are discussed in Section V.

II. Background

The hierarchical content-based motion-estimation method proposed by Van Beek *et al.* [17] is especially relevant within the context of this paper, hence, it will be briefly described. This method inherits its hierarchical nature from the process

Manuscript received January 30, 2002; revised March 16, 2003. This paper was recommended by Associate Editor S. U. Lee.

The authors are with the Center for Signal and Image Processing, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: gregib@ece.gatech.edu; yucel@ece.gatech.edu; rmm@ece.gatech.edu).

Digital Object Identifier 10.1109/TCSVT.2003.816520

TABLE I
COMPARISON OF VARIOUS 2-D MESH-BASED MOTION-ESTIMATION ALGORITHMS

Method	Motion Est.	Mesh-Type	Mesh Hierarchy	Image Size/Hierarchy
Nakaya *et al.* [2]	Search	Regular	None	N/A
Wang *et al.* [3]	Optimization	Content-based	None	N/A
Altunbasak *et al.* [4]	Closed-form	Content-based	None	N/A
Toklu *et al.* [16]	Search	Regular	Coarse-to-fine	Constant size
Van Beek *et al.* [17]	Search	Content-based	Coarse-to-fine	Constant size
Proposed	Search	Content-based	Hierarchical	Var. size/Gaus. Pyr.

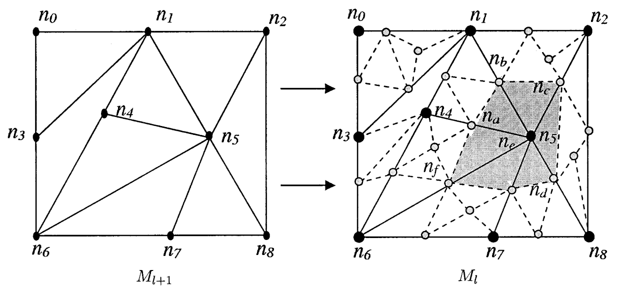

Fig. 1. Generating a fine 2-D mesh from a coarse one while the image is kept spatially at the same size.

of generating a fine mesh from a coarse mesh. The finer mesh represents the scene motion more accurately since new nodes are introduced. This is illustrated in Fig. 1, where the mesh on the left, i.e., M_{l+1}, represents the coarse mesh, while the one on the right, i.e., M_l, denotes the fine mesh. In M_{l+1}, there are nine node points labeled as $n_0, \ldots, n_8$ that are retained in M_l with more node points introduced by the mesh refinement algorithm. These new node points are denoted as gray-color dots and the new edges appear as dashed lines. The motion vectors calculated by the generalized polygonal search procedure [2] at the node points of the coarse mesh M_{l+1} are used in initializing newly added node points in the fine mesh M_l. In the remainder of this paper, this method is referred to as method I.

III. THEORY

Although the above method outperforms the single-level tracking in terms of motion vector accuracy and computational complexity, it *still* incurs high computational load. To further reduce the computation time, the proposed method constructs a hierarchical/pyramid representation for both the images and corresponding meshes. Accordingly, the image at each level will be a blurred and downsampled version of the image in the previous level. Similarly, the 2-D mesh at each level will be a coarse version of the one in the previous level. As a result, the computation time is significantly reduced since the patches and search ranges are small in size at the coarsest level of the pyramid. Another advantage arises from the fact that motion-estimation algorithms are less prone at becoming trapped at the local minima when applied to coarse images (e.g., blurred and downsampled images).

In this paper, two different implementations of this method are proposed. They differ in the level of hierarchy at which the mesh is designed, as well as the method of constructing mesh levels. These two implementations are discussed in the Sections III-A and B, respectively.

A. Implementation I

As mentioned earlier, the proposed method constructs a hierarchical representation for both the image and 2-D mesh. This is valid in both implementations. However, in the first implementation, the 2-D mesh is designed at the coarsest level. The 2-D mesh in the next finer level is then constructed from the coarser mesh by a coarse-to-fine mesh-generation algorithm, which is described in Section IV-A.

Fig. 2 illustrates this implementation for a specific example of two-level hierarchy for simplicity, though it can easily be generalized for any number of levels. An outline of this implementation is as follows.

```
1. Generate a coarse image $I_1$ by blurring and downsampling the original image $I_0$.
2. Design a mesh $M_1$ for the coarse image $I_1$.
3. Compute the motion vectors at the node points in $M_1$.
4. Generate a fine mesh $M_0$ by refining $M_1$ using the coarse-to-fine mesh-generation algorithm.
```

After this step, the motion vectors for the node points in M_0 needs to be initialized by the node-point motion vectors estimated at level 1. Assume that we are interested in finding the motion vectors for a node point with coordinates (x, y) in I_0. This can be achieved as follows.

```
1. Find the corresponding coordinates of the node point in $I_1$, which are equal to $(x/2, y/2)$.
2. Locate the patch $P_i$ in $I_1$ where the point $(x/2, y/2)$ lies in.
3. Calculate the affine parameters (for $P_i$) from the motion vectors of the vertices of the patch $P_i$.
4. Compute the motion vector for the point $(x/2, y/2)$ from the affine parameters estimated in step 3. Scale this motion vector by a factor of two to calculate the motion vector for the node point $(x, y)$ in $M_0$.
```

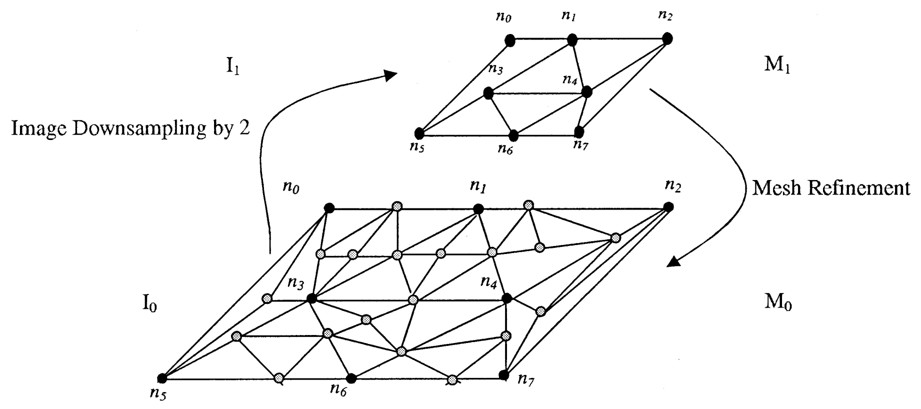

Fig. 2. Illustration of implementation I for a two-level hierarchy.

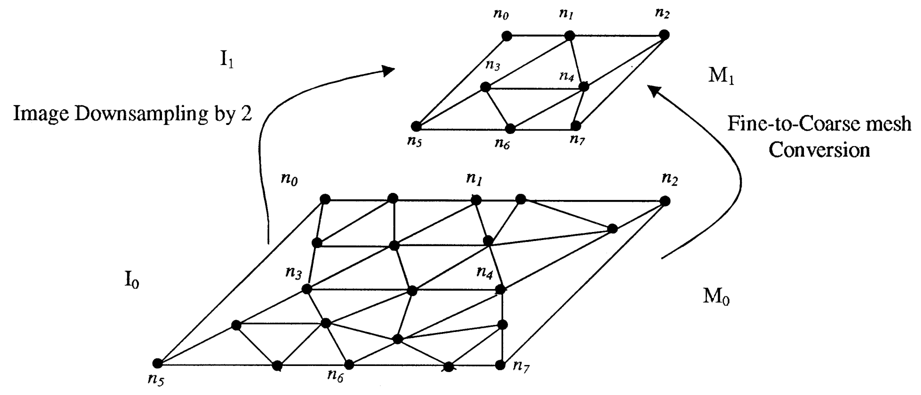

Fig. 3. Illustration of implementation II for a two-level hierarchy.

B. Implementation II

While the first implementation designs the 2-D mesh at the coarsest level, this implementation designs the mesh at the finest level. Similar to the first implementation, the tracking algorithm is still applied on the coarsest level. However, while a coarse-to-fine *mesh-generation* algorithm is needed for the first implementation, here, a fine-to-coarse *mesh-reduction* algorithm is employed.

The process of getting a coarse mesh from a fine one is modified from progressive-mesh methods, which are commonly employed in 3-D computer graphics [18]. This technique will be discussed in details in Section IV-B. Fig. 3 illustrates this implementation for a specific example of two-level hierarchy. The outline of this implementation is as follows.

```
1. Generate a coarse image $I_1$ by blurring
and downsampling the original image $I_0$.
2. Design a 2-D mesh $M_0$ for the original
image $I_0$.
3. Construct a coarse mesh $M_1$ from $M_0$ by
applying the fine-to-coarse mesh-reduction
algorithm.
```

This is indicated by the right-hand arrow that heads up the hierarchy in Fig. 3, while in the first implementation, this arrow heads down the hierarchy, as in Fig. 2. After M_0 and M_1 is reconstructed, the rest of implementation II is the same as that of implementation I.

IV. Mesh Construction

As mentioned earlier, the two implementations of the proposed method differ in the level of hierarchy at which the mesh is designed, as well as the method of constructing mesh levels. While the first implementation employs a mesh-refining algorithm, the second one uses a fine-to-coarse mesh-generation algorithm. These two algorithms are described here.

A. Coarse-to-Fine Mesh Construction

Here, we modify the 2-D content-based triangular mesh design previously proposed by Altunbasak and Tekalp [4]. The algorithm retains the selected node points from the coarser mesh level, and further selects nonuniformly spaced node points at each level. An outline of the algorithm is as follows.

```
1. Generate an $L$-level Gaussian image
pyramid by blurring and downsampling the
original image pair. Let $l = 0,\ldots,L-1$
denote levels of the hierarchy, where
$l = 0$ represents the original images. Set
$l = L-1$.
2. Label all pixels as “unmarked.”
3. Scale the locations of all node points
selected at the level $l+1$ by a factor of
```

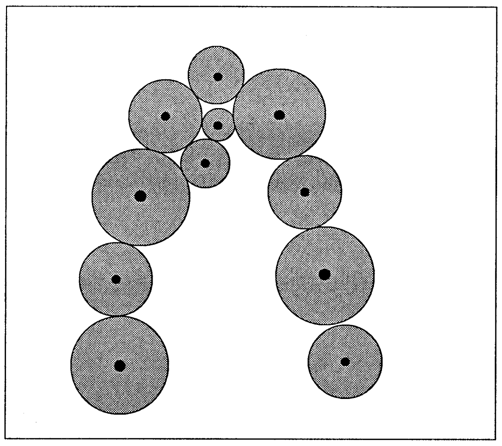

Fig. 4. Partitioning the image into patches such that the predefined function of the DFD within each patch attains approximately the same value.

two. Include these points in the list of selected node points at the level l.
4. Compute the average displaced frame difference (DFD) $\mathrm{DFD}_{\mathrm{avg}}[l]$ given by

$$\mathrm{DFD}_{\mathrm{avg}}[l] = \frac{\sum_{(x,y)} \mathrm{DFD}(x,y)}{K[l]} \qquad (1)$$

where the summation is over all the image pixels at the level l, and $K[l]$ is the number of node points at that level.
The goal is to partition the image into patches such that the predefined function of the DFD within each patch attains approximately the same value as shown in Fig. 4.
5. Grow a region about all selected node points until $\sum \mathrm{DFD}(x,y)$ in this region is greater than $\mathrm{DFD}_{\mathrm{avg}}[l]$. Label all pixels within this region as "marked."
6. If the number of retained nodes (from the coarser level of the hierarchy) is less than $K[l]$, then select more node points as follows.
 (a) Compute a cost function $C(x,y)$ associated with each unmarked pixel as a predefined function of spatio-temporal intensity gradients **[4]**. The cost function includes terms that are functions of both spatial and temporal intensity gradients so that selected node points, hence, the boundaries of the patches, coincide with spatial and motion edges.
 (b) Find the unmarked pixel with the highest $C(x,y)$, which is not closer to any other previously selected node point than a prespecified distance. Label this point as a node point.
 (c) Grow a region about this node point until $\sum \mathrm{DFD}(x,y)$ in this region is greater than $\mathrm{DFD}_{\mathrm{avg}}[l]$. Label all pixels within this region as "marked."
 (d) Go to 6(c) until a desired number of node points ($K[l]$) are selected.
7. Given the selected node points at the level l, apply a triangulation procedure (e.g., Delaunay triangulation [19]) to obtain a content-based mesh.
8. Decrease l by one. Unless $l < 0$ go to step 2.

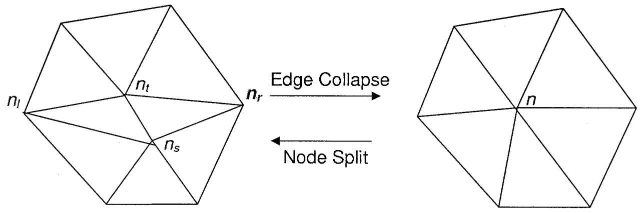

Fig. 5. Constructing fine meshes from a given coarse mesh and visa versa.

Intuitively, at each level, the algorithm tries to place more densely populated node points where the motion is erratic. It should also be noted that previously selected node points are retained at the next level. These node points claim their neighborhood before any additional node points are selected at the next level. That is, a region is grown and marked around already selected nodes from the previous level before we start selecting additional nodes at the current level.

B. Fine-to-Coarse Mesh Construction

The fine mesh construction algorithm used in the second implementation depends on two related operations: node split and edge collapse, which are shown in Fig. 5. These two operations are the 2-D mesh equivalent operations to the 3-D mesh ones, which are called vertex-split and edge collapse, respectively. Practically, a series of node-split operations will produce a fine mesh from a coarse one, while a series of edge-collapse operations will result in a coarse mesh from a fine mesh. The following algorithm outlines the main steps in constructing levels of coarse meshes from a fine one.

1. Start with a fine mesh M_0.
2. Apply the edge-collapse transform $\mathrm{ecol}_i(n_s, n_t)$ on the edge connecting the nodes n_s and n_t, where n_s and n_t are internal nodes of the mesh. The index i in ecol_i indicates the level at which this edge-collapse command is taking place.
3. The choice of the edge as well as the new location of the new node, with connectivity-K and A scalar attributes, depends on minimizing the following energy function:

$$E_K = \min_{N,A}\{E_{\mathrm{spring}}(N) + E_{\mathrm{scalar}}(N,A)\}$$

TABLE II
EXPERIMENTAL RESULTS FOR THE SEQUENCE "FLOWER&GARDEN" (352×240)

Method	PSNR	Time in sec	♯ of Lev.	♯ of internal Nodes	Search Size	Step Size
I	14.23	45.16	3	[100, 16, 1]	[2, 4, 8]	[0.5, 1.5, 4.0]
II	14.35	35.51	3	[100, 16, 1]	[2, 2, 2]	[0.5, 0.75, 1.0]
I	14.34	157.75	3	[100, 16, 1]	[4, 8, 16]	[0.5, 1.5, 4.0]
II	15.96	125.81	3	[100, 16, 1]	[4, 4, 4]	[0.5, 0.75, 1.0]
I	12.94	57.84	5	[100, 64, 36, 16, 1]	[2, 4, 8, 16, 32]	[0.5, 1.5, 4.0, 10.0, 24.0]
II	16.96	36.13	5	[100, 64, 36, 16, 1]	[2, 2, 2, 2, 2]	[0.5, 0.75, 1.0, 1.25, 1.5]
I	14.72	196.20	5	[100, 64, 36, 16, 1]	[4, 8, 16, 32, 64]	[0.5, 1.5, 4.0, 10.0, 24.0]
II	17.19	127.52	5	[100, 64, 36, 16, 1]	[4, 4, 4, 4, 4]	[0.5, 0.75, 1.0, 1.25, 1.5]

TABLE III
EXPERIMENTAL RESULTS FOR THE SEQUENCE "FOREMAN" (352×288)

Method	PSNR	Time in sec	♯ of Lev.	♯ of internal Nodes	Search Size	Step Size
I	27.09	49.12	3	[100, 16, 1]	[2, 4, 8]	[0.5, 1.5, 4.0]
II	27.87	33.42	3	[100, 16, 1]	[2, 2, 2]	[0.5, 0.75, 1.0]
I	24.61	170.58	3	[100, 16, 1]	[4, 8, 16]	[0.5, 1.5, 4.0]
II	27.83	118.28	3	[100, 16, 1]	[4, 4, 4]	[0.5, 0.75, 1.0]
I	27.50	68.99	5	[100, 64, 36, 16, 1]	[2, 4, 8, 16, 32]	[0.5, 1.5, 4.0, 10.0, 24.0]
II	29.22	42.59	5	[100, 64, 36, 16, 1]	[2, 2, 2, 2, 2]	[0.5, 0.75, 1.0, 1.25, 1.5]
I	27.63	235.61	5	[100, 64, 36, 16, 1]	[4, 8, 16, 32, 64]	[0.5, 1.5, 4.0, 10.0, 24.0]
II	29.22	150.41	5	[100, 64, 36, 16, 1]	[4, 4, 4, 4, 4]	[0.5, 0.75, 1.0, 1.25, 1.5]

where E_{spring} is used to regularize this optimization problem and $E_{\text{scalar}}(N, A)$ calculates the cost in terms of attributes of the new vertices (intensity and motion vector). More specifically, E_{scalar} calculates the distortion between the new frame when the edge is collapsed and the frame when no edges are collapsed. This is done for all edges and the edge with the minimum distortion is chosen to be collapsed. This energy function is minimized over all edges and the edge with the minimum E is collapsed at this ecol_i transform. In order to achieve scale invariance of the terms, the mesh is uniformly scaled to fit in a unit square. The reader is referred to [18] for more information on this energy function minimization for 3-D meshes.

4. If

$$\text{the total number of nodes} \\ > \text{number of boundary nodes in } M_i,$$

then repeat step 2 for another set of nodes. Otherwise, stop the algorithm and the resulting mesh is the coarsest mesh.

V. RESULTS

We have designed experiments to compare the proposed method (method II) with the algorithm in [17] (method I) with two video sequences: FLOWER&GARDEN and FOREMAN. These experiments were conducted using a 933-MHz PC. Furthermore, the motion-estimation method, proposed by Lucas and Kanade, was used for methods I and II [1]. We used the first implementation we discussed in Section III because both implementations give similar performance. The results are documented in Tables II and III for FLOWER&GARDEN and FOREMAN sequences, respectively. The parameters used in these experiments are also tabulated in the same tables. The number of nodes at each level of hierarchy is provided in the fifth column where the first number in the array represents the number of nodes at the finest level while the last number in the array represents the number of nodes at the coarsest level. The number of nodes at each level is kept the same for both methods and the same mesh design is used for both methods. The search size in method I is increased by a factor of two in both the horizontal and vertical directions at each level of the hierarchy since it does not utilize an image pyramid representation. Without increasing the search size, method I would be at a disadvantage. The PSNR of the motion compensated frame difference is calculated and tabulated in the second column for a pair of frames in each sequence. The PSNR before motion compensation is 11.31 and 22.41 for the FLOWER&GARDEN and FOREMAN sequences, respectively. Notably, in Table III, the PSNR deteriorates from 27.09 to 24.61 when the search size increases using method I. This is because the search algorithm got trapped at a local minima when the search range is increased, as explained in Section III. However, it is less likely that the proposed method gets trapped at local minima since the motion-estimation algorithm is applied on coarse images.

TABLE IV
THEORETICAL COMPARISON RESULTS

	N. of additions
Method I	$\sum_{l=0}^{L-1} 9N^2 \frac{R^2}{S^2} I$
Method II	$\sum_{l=0}^{L-1} 9N^2 \frac{R^2}{S^2} \frac{1}{2^l} I$

The execution time is given in the third column of Tables II and III. This is the total processing time including the decimation time. No effort has been directed to optimize algorithms in terms of execution speed. As shown in these tables, the main advantage of the proposed method is its speed. The execution time for the proposed method is smaller than the execution time of method I for all conducted experiments.

The theoretical computational complexity for both methods is given in Table IV, where N by N is the original image size, P is the number of nodes at the finest level,[1] R by R is the search range at the finest mesh level, I is the number of iterations at the hexagonal matching procedure, S is the search step size at the finest mesh level, and L is the number of levels. This table provides an approximate number of addition operations involved in the motion-estimation stage. Since the affine warping operations along a horizontal line can be computed using additions, the multiplication operations is negligible. In the calculation for method I, the search range R is assumed to be increased at each level by a factor of two. Similarly, the step size S is increased by the same ratio. The number of nodes P is reduced by four at each level for both methods. This table assumes a regular mesh as opposed to a content-based mesh to make the calculations tractable. From the results in Tables II and III and the comparisons in Table IV, we can claim that the proposed method performs equally or slightly better in terms of PSNR and runs significantly faster. The latter is a direct consequence of pyramid image representation.

VI. CONCLUSION

Method II has three main advantages as compared to method I. These advantages are valid in both implementations. First, the pyramid/hierarchical image representation will reduce the computational complexity considerably. Node-point motion-estimation computational complexity is mainly a function of the number of nodes, patch sizes, and search range. Second, since the image is blurred and downsampled at higher levels, the proposed method is more likely to lock onto global motions at these levels rather than being trapped by local motions. Third, hierarchical image representation utilized in method II will decrease the execution time for content-based mesh design/modification process at higher levels.

[1]P has been dropped while simplifying the expressions.

REFERENCES

[1] A. M. Tekalp, *Digital Video Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

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

[3] Y. Wang and O. Lee, “Active mesh – A feature seeking and tracking image sequence representation scheme,” in *IEEE Trans. Image Processing*, vol. 3, Sept. 1994, pp. 610–624.

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

[5] C.-S. Fuh and P. Maragos, “Affine models for image matching and motion detection,” in *Proc. Int. Acoustics Speech and Signal Processing Conf.*, Toronto, ON, Canada, May 1991, pp. 2409–2412.

[6] V. Seferidis and M. Ghanbari, “General approach to block-matching motion estimation,” *Opt. Eng.*, vol. 32, no. 7, pp. 1464–1474, July 1993.

[7] O. Lee and Y. Wang, “Motion compensated prediction using nodal based deformable block matching,” *J. Vis. Commun. Image Processing*, vol. 6, no. 1, pp. 26–34, Mar. 1995.

[8] H. Brusewitz, “Motion compensation with triangles,” in *Proc. 3rd Int. 64 kbit coding of moving video Conf.*, Rotterdam, The Netherlands, Sept. 1990.

[9] G. J. Sullivan and R. L. Baker, “Motion compensation for video compression using control grid interpolation,” in *Proc. Int. Acoustics Speech and Signal Processing Conf.*, Toronto, ON, Canada, May 1991, pp. 2713–2716.

[10] 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, Jan. 1994.

[11] Y. Wang, O. Lee, and A. Vetro, “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 Systems Video Technol.*, vol. 6, pp. 647–659, Dec. 1996.

[12] Y. Altunbasak, A. M. Tekalp, and G. Bozdagi, “Two-dimensional object-based coding using a content-based mesh and affine motion parameterization,” in *Proc. IEEE Int. Image Conf.*, vol. 2, Washington, DC, Oct. 1995, pp. 394–397.

[13] Y. Altunbasak and A. M. Tekalp, “Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes,” *IEEE Trans. Image Processing*, vol. 6, pp. 1255–1269, Sept. 1997.

[14] P. J. L. van Beek and A. M. Tekalp, “Object-based video coding using forward tracking 2-D mesh layers,” in *Proc. Visual Communications and Image Processing*, San Jose, CA, Feb. 1997, pp. 699–710.

[15] A. T. Erdem, C. Toklu, A. M. Tekalp, and M. I. Sezan, “2D mesh based tracking of deformable objects with occlusion,” in *Proc. Int. Image Processing Conf.*, Lausanne, Switzerland, Sept. 1996.

[16] C. Toklu, A. T. Erdem, M. I. Sezan, and A. M. Tekalp, “Tracking motion and intensity variations using hierarchical 2D mesh modeling for synthetic object transfiguration,” *Graph. Models Image Process.*, vol. 58, no. 6, pp. 553–573, Nov. 1996.

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

[18] H. Hoppe, “Progressive meshes,” in *Proc. ACM SIGGRAPH*, 1996, pp. 99–108.

[19] J. R. Shewchuk, “Triangle: Engineering a 2D quality mesh generator and Delaunay triangulators,” in *Proc. 1st Applied Computational Geometry Workshop*, PA, 1996, pp. 124–133.