TWO DIMENSIONAL HIERARCHICAL MESH BASED VIDEO COMPRESSION TECHNIQUES

Shan Du, Wael Badawy and K. V. I. S. Kaler

Department of Electrical and Computer Engineering
University of Calgary
2500 University Drive NW
Calgary, Alberta, Canada T2N 1N4
{dus, badawy, kaler}@enel.ucalgary.ca

ABSTRACT

In this paper, a novel mesh generation and coding method that is suitable for low bit rate video applications is proposed. It is compared with other mesh approaches in terms of their construction, visual quality and code size. The proposed technique significantly reduces the number of bits required for the coding of the node positions defining the mesh topology. This reduction induces an improvement of either the image quality or the global bit rate. The performance analysis shows that the proposed technique outperforms others.

1. INTRODUCTION

With the rapid advancement of telecommunication and video applications, how to compress video data and transmit it via low bit rate channels attracts more attention. Existing video compression methods use motion compensation to decorrelate the data along the temporal domain. 2D mesh based motion segmentation and video coding is one of the commonly used techniques. This method removes the high temporal redundancy presented in video sequences to achieve high compression ratios.

Mesh-based motion estimation and compensation that uses control grid to track the deformation of video frames [1] is one of the most advanced methods for video representation and processing. Compared with the widely used block-matching algorithm (BMA), the mesh-based approach is capable of representing more types of motions such as rotation, shear, etc., and thus is an attractive method for many applications in video processing.

Recently, hierarchical mesh representation attracts much attention because it allows scalable / progressive transmission of the mesh geometry and motion vectors [2], [3], where the mesh can be coded at different resolution to satisfy the bandwidth constraints. Hierarchy is defined as either coarse-to-fine or fine-to-coarse resolution, where the former means subdividing patches into three or four sub-triangles or quadrangles and the latter means merging smaller patches into larger triangles or quadrangles.

The mesh-based motion estimation algorithm generates the mesh and estimates the motion vectors for the mesh nodes. Then the encoder sends the mesh code and the nodes' motion vectors to the decoder. The decoder uses the mesh topology and the nodes' motion vectors to predict the frame by tracking the deformation of the mesh. The mesh-based motion compensation is achieved by using parametric mappings such as affine, bilinear, and perspective texture mappings. There are different mesh structures can be used for mesh-based motion estimation and compensation such as Uniform mesh, Delaunay mesh, Hierarchical Adaptive Structured Mesh (HASM), and the newly proposed Hierarchical Adaptive Merge/Split Structure Mesh (HAMSM).

The rest of this paper is organized as follows: Section 2 introduces the popular mesh-based video compression techniques. Section 3 analyzes the performances of different techniques. In the last section, the conclusion is presented.

2. 2D MESH BASED METHODS

A 2-D mesh is a planar graph that partitions a 2-D image region into polygonal patches. The vertices of the polygonal patches are referred to as node points [4]. The motion of mesh nodes represents the dynamics of the video objects.

2.1 The Mesh Generation

The location of mesh nodes is defined using uniform mesh, Delaunay mesh, HASM, or HAMSM. Each technique can be used for hierarchical representation.

2.1.1 Uniform Hierarchical Mesh

The uniform mesh topology refers to fixed patch size as shown in Figure 1. The hierarchical structure can be achieved by partitioning the mesh into finer resolution. The uniform mesh poorly detects the dynamics of the video objects because the mesh node pattern is fixed for any video sequence.

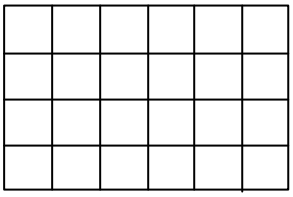


Figure 1. The uniform mesh topology

2.1.2 Delaunay Hierarchical Mesh

Delaunay mesh is widely used in video object motion tracking (shown in Figure 2). The mesh topology is generated using Voronoi diagrams. Voronoi diagram is a geometric dual of Delaunay mesh and one can be derived from the other. In summary, the Delaunay mesh can be generated as follows: Given a set of N points in a plane, Voronoi diagram divides the domain into a set of polygonal regions, the boundaries of which are the perpendicular bisectors of the lines joining the points. Furthermore each tile contains only one of the N points. If both these conditions are satisfied then the lines joining the points form a mesh known as the Delaunay mesh. The hierarchical structure can be obtained with ranking each node and placing it in a certain level of details. The Delaunay mesh suffers from the processing requirements for mesh processing. Moreover, the nodes of the mesh are located based on the constrained algorithm, which does not capture the dynamics of the video.

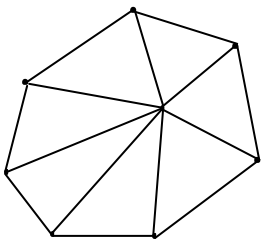


Figure 2. The Delaunay mesh topology

2.1.3 Hierarchical Adaptive Structured Mesh (HASM)

The HASM is a coarse-to-fine adaptive mesh that captures the dynamics of video objects. The mesh is generated at its coarse level as shown in Figure 3 (uniform mesh with a defined resolution). Then each patch with more motion components (dynamics) is divided into smaller patches as shown in Figure 4. So it captures more information about the motion of the patch.

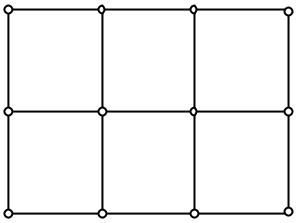


Figure 3. Coarse level mesh of HASM

(white nodes are mesh nodes)

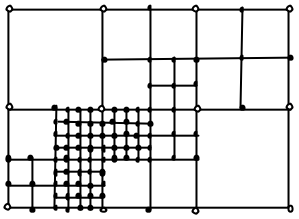


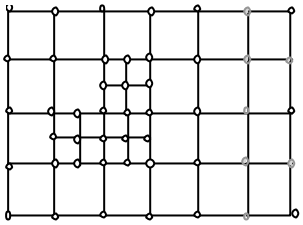
Figure 4. Refined mesh of HASM

(black nodes are newly generated nodes)

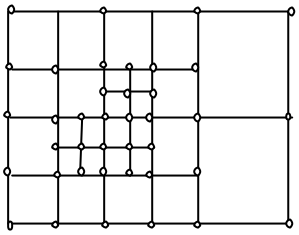
HASM can be applied using different techniques. Technique 1 is a central technique that the split begins from a whole frame while technique 2 is a parallel technique that split begins from an initial mesh at resolution *r* and each patch with more motion components is split into finer resolution. HASM suffers from the initial resolution selection, where a large resolution will require more splitting operation while a small resolution adds more nodes which may be classified as redundant information as shown in Figure 5.

2.1.4 HAMSM

The Hierarchical Adaptive Merge/Split Structured Mesh (HAMSM) is a technique that constructs mesh topology using both coarse-to-fine and fine-to-coarse techniques. It reduces the overhead of mesh coding



a) HASM topology



b) Optimal mesh

Figure 5. HASM topology and optimal mesh
(redundant node)

since it codes the mesh topology as a count of splitting and merging instead of the mesh nodes' locations. The mesh is constructed at different levels of detail l_m-l_s, where *s* is the maximum splitting level and *m* is the maximum merging level. At any level of details, the decoder can predict the frame using texture mapping to achieve hierarchical approach.

The mesh topology is generated through a split and merge procedure. The initial mesh at resolution *r* is shown in Figure 6. The mesh patch used in this paper is square. The white nodes are the mesh nodes, and *r* presents the width or height of a patch.

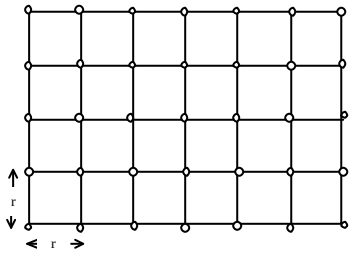


Figure 6. Initial mesh topology

Then according to a split and merge criterion, patches with high prediction error are divided into four smaller patches evenly while patches with similar motion parameters are merged together to form a larger square patches. The splitting and merging procedure are processed recursively until the criterion is satisfied or the maximum level is reached. Figure 7 shows the merge and split operations where the black nodes are newly generated nodes (split operation), and the dotted lined nodes are eliminated nodes (merge operation).

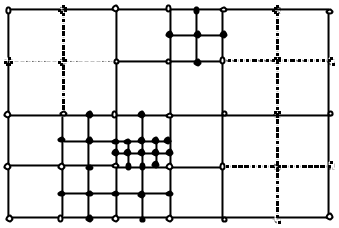


Figure 7. Final mesh topology

In summary, the algorithm can be described as split step and merge step separately that are presented as follows:

a) Split step

For each patch p_i which is represented as a square *abcd*:

1) Give the maximum splitting level l_s
2) $l_{current}$=0
3) Compute the motion vectors of the four nodes of the patch v_a, v_b, v_c, v_d
4) If ($l_{current} < l_s$)
5) If (* $|p_{i,k} - p_{i,k+1}| >$threshold)
6) Locate new nodes e, f, g, h, i
7) Generate v_e, v_f, v_g, v_h, v_i
8) Increment $l_{current}$ by 1
9) For each new patch repeat steps 3-8

* k means frame k, and k+1 means frame k+1

b) Merge step

For every four unsplit patches (NW, NE, SW, SE) which compose a bigger square patch p_i:

1) Give the maximum merging level l_m
2) $l_{current}$=0
3) Compute the motion vectors v_{NWa}, v_{NEb}, v_{SWc}, v_{SEd}
4) If ($l_{current} < l_m$)
5) If ($|p_{i,k} - p_{i,k+1}|$ <threshold)
6) Merge the four patches and generate a new bigger patch
7) Increment $l_{current}$ by 1
8) For each new patch repeat steps 3-7

The motion vectors of mesh nodes are estimated by generating an 8×8 block around the node and then using block-matching to generate the motion vector. The search criterion used in this paper is Three Step Search (TSS) and the matching criterion used is

Minimum Sum Absolute Difference (SAD). But any search or matching criterion can be used.

2.2 The Mesh Coding

2.2.1 Uniform Hierarchical Mesh

The uniform mesh topology can be coded using three numbers that describe the number of patches in column, the number of patches in row, and the size of each patch (hierarchical level). The uniform mesh topology has the fewest number of bits.

2.2.2 Delaunay Hierarchical Mesh

The mesh topology can be coded by nodes' locations or by predictive coding which codes the relative distance between neighboring nodes. The predictive coding requires a sorting algorithm that orders the mesh nodes.

2.2.3 HASM

HASM is a coarse-to-fine mesh that is coded using tree structure and it can be coded using a single bit for each tree node. This bit represents either split or non-split. Figure 8 shows the central HASM technique where the frame is used as one patch. Figure 9 shows the parallel HASM technique where the frame is split into six patches. This technique speeds up the mesh construction operation. The node of the tree represents the mesh patch. The node that has four child nodes represents a patch that is split into four smaller patches.

For HASM central technique in Figure 8, the bit stream is

11111101000000100000100000000000000000

For HASM parallel technique in Figure 9, the bit stream is

01111000010000011111000000111100001111111
11000000000000000000000000000000000000000000000
0000000000

Where 1 represents split and 0 represents no operation.

For HASM parallel technique, three numbers are needed to represent the initial mesh. The first one describes the number of patches in column, the second one describes the number of patches in row, and the third one describes the patch size.

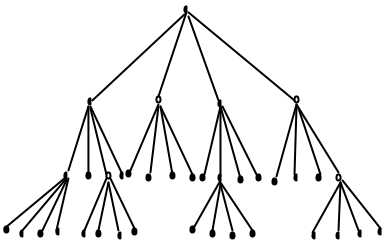


Figure 8. HASM mesh coding
(tree representation for central technique)

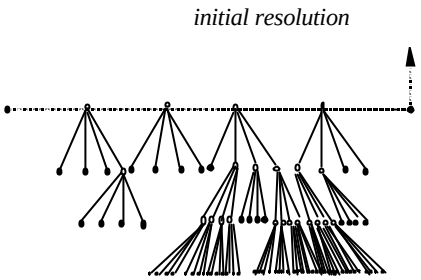


Figure 9. HASM mesh coding
(tree representation for parallel technique)

2.2.4 HAMSM

Since the patch is split into four patches or four patches are merged into a bigger patch, the quadtree data structure has been used to code the split and merge operation. The mesh topology can be coded as a count of splitting and merging instead of the node's location. In the case of quadtree, the size of the bits that represents the mesh topology will be significantly reduced. This reduction can be used to improve the image quality.

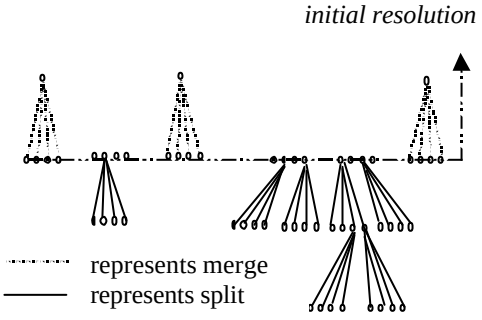


Figure 10. HAMSM quadtree structure

HAMSM is coded using tree structure and it can be coded using only two bits for each tree node. 00

represents leaf nodes, 01 represents a patch split, 10 represents a patch merge. In addition, three numbers are needed to represent the initial mesh. The first one describes the number of patches in column, the second one describes the number of patches in row, and the third one describes the patch size. The following bit stream represents the bit representation for the mesh topology in Figure 10.

10000100001000010001010001001000000000000000
00000000000000001010000000000000000000000000

Because Three Step Search (TSS) is used in HAMSM, the region of motion vectors is [-7, 7]. So four bits are enough to represent the *x* component of a motion vector, and four bits represent *y* component. One byte represents one motion vector.

2.3 The Motion Compensation

All mesh topology can use affine transformation to compute motion compensation because each quadrantal patch can be easily split into two triangles.

2.3.1 Uniform Hierarchical Mesh

The uniform mesh has a patch topology that allows the motion compensation to use the multiplication-free scanline algorithm [8]. This saves the power consumption.

2.3.2 Delaunay Hierarchical Mesh

The Delaunay mesh has a different patch topology. It uses the scanline algorithm for affine transformation.

2.3.3. HASM

HASM has a patch topology that allows the use of the multiplication-free scanline algorithms. It saves the power consumption.

2.3.4 HAMSM

HAMSM has a patch topology that allows the use of the multiplication-free scanline algorithms. It saves the power consumption.

3. PERFORMANCE ANALYSIS

The performances of uniform mesh, HASM, and HAMSM are analyzed in terms of visual quality and mesh code size.

3.1 Visual Quality

The visual quality is measured with Peak-Signal-to-Noise-Ratio (PSNR). Figure 11 demonstrates that HAMSM has approximate visual quality as HASM and 8×8 Uniform mesh. Figure 12 demonstrates that HAMSM has higher visual quality than 16×16 Uniform mesh.

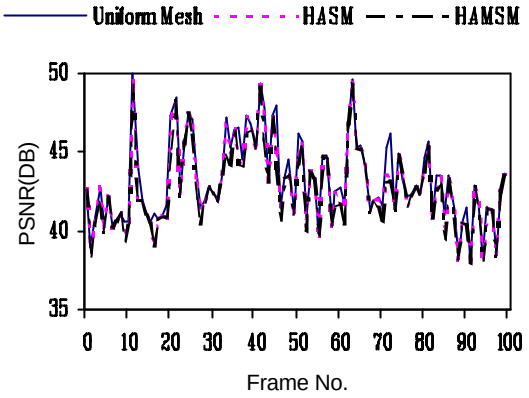


Figure 11. The PSNR using 8×8 Uniform Mesh, HASM and HAMSM (Claire sequence)

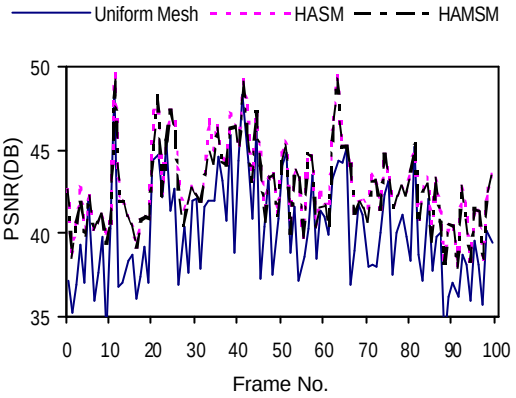


Figure 12. The PSNR using 16×16 Uniform Mesh, HASM and HAMSM (Claire sequence)

3.2 Mesh Code Size

Figure 13 and Figure 14 shows in the average case, HAMSM needs less bit number to represent mesh topology and motion vectors while it has similar or even higher visual quality. Figure 15 displays the Uniform mesh, HASM and HAMSM on the real video sequence.

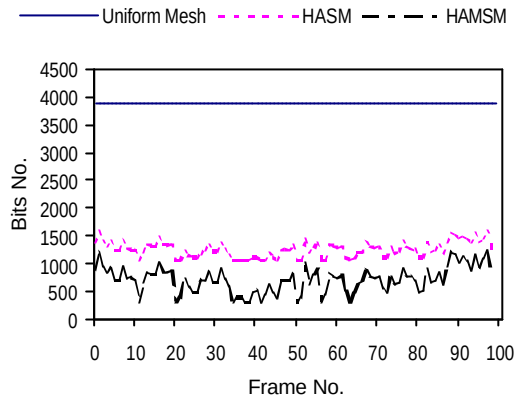


Figure 13. The number of bits used to code 8×8 Uniform Mesh, HAMSM and HASM mesh topology and motion vectors (Claire sequence)

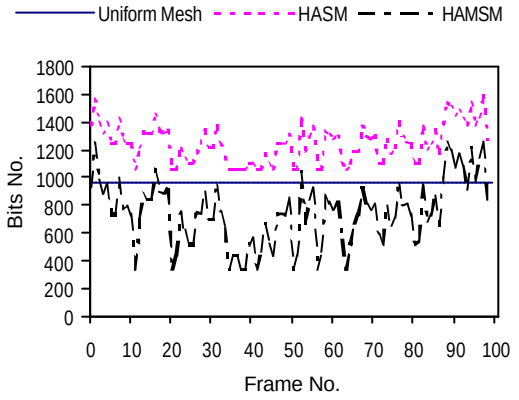


Figure 14. The number of bits used to code 16×16 Uniform Mesh, HAMSM and HASM mesh topology and motion vectors (Claire sequence)

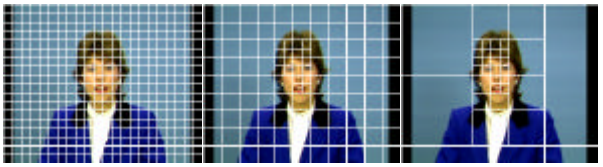


Figure 15. 8×8 Uniform mesh, HASM, and HAMSM on the sixth frame of Claire sequence.

4. CONCLUSION

This paper presents the new Hierarchical Adaptive Merge/Split Structured Mesh (HAMSM) in terms of its algorithm, mesh coding and performance. It shows that HAMSM has acceptable characteristics. It captures the dynamics of video objects and ignores the subtle motions. It merges the regions that have little motions. So it reduces the mesh code in terms of bit number. And it has acceptable visual performance. This technique outperforms Uniform mesh and HASM.

REFERENCES

[1] Nosratinia, "New kernels for fast mesh-based motion estimation," IEEE Trans. On Circuits and Syst. for video Technol. vol. 11, no. 1, pp. 40-51, Jan. 2001.

[2] P. J. L. Van Beek, A. M. Tekalp, and A. Puri, " 2D mesh geometry and motion compression for efficient object-based video representation," Int. Conf. On Image Processing'97, Santa Barbara, CA, Oct. 1997.

[3] A. M. Tekalp, P.V. Beek, C. Toklu, and B. Gunsel, " Two-dimensional mesh-based visual-object representation for interactive synthetic/natural digital video," Proc. Of the IEEE, vol. 86, no. 6, pp. 1029-1051, June 1998.

[4] G. Wolberg, Digital Image Warping. Los Alamitos, CA: Computer Society Press, 1990.

[5] Y. Wang and O. Lee, "Active mesh - A feature seeking and tracking image sequence representation scheme," IEEE Trans. On Image Processing, Vol. 3, pp. 610-624, 1994.

[6] Wael Badawy and Magdy Bayoumi, "On minimizing hierarchical mesh coding overhead: (HASM) Hierarchical Adaptive Structured Mesh approach," IEEE International Conference on Acoustics, Speech, and Signal Processing, Istanbul, Turkey, June 5-9, 2000, pp. 1923-1926.

[7] Wael M. Badawy and Magdy A. Bayoumi, "A low cost real time motion estimation VLSI core," 1999 International Workshop on Design, Test and Applications, WTAD'99, Dubrovnik, Croatia, June 14-16, 1999.

[8] Wael Badawy and Magdy Bayoumi, "A multiplication-free parallel architecture for affine transformation," The IEEE International Conference on Application-specific Systems, Architectures and Processors, Boston, MA, July 10-12, 2000, pp.25-34.