Semi-Regular Representation and Progressive Compression of 3-D Dynamic Mesh Sequences

Jeong-Hyu Yang, Chang-Su Kim, *Senior Member, IEEE*, and Sang-Uk Lee, *Senior Member, IEEE*

***Abstract*—We propose an algorithm that represents three-dimensional dynamic objects with a semi-regular mesh sequence and compresses the sequence using the spatiotemporal wavelet transform. Given an irregular mesh sequence, we construct a semi-regular mesh structure for the first frame and then map it to subsequent frames based on the hierarchical motion estimation. The regular structure of the resulting mesh sequence facilitates the application of advanced coding schemes and other signal processing techniques. To encode the mesh sequence compactly, we develop an embedded coding scheme, which supports signal-to-noise ratio and temporal scalability modes. Simulation results demonstrate that the proposed algorithm provides significantly better compression performance than the static mesh coder, which encodes each frame independently.**

***Index Terms*—Progressive compression, semi-regular mesh sequence, three-dimensional (3-D) mesh compression, 3-D motion estimation.**

I. Introduction

THREE-DIMENSIONAL (3-D) meshes are emerging multimedia contents to represent realistic visual data. They consist of geometrical positions of sampled points (or vertices) and connectivity relations among the vertices. The huge size of a typical 3-D mesh has necessitated the development of mesh compression technology. Many algorithms have been proposed to compress 3-D static meshes, among which the semi-regular mesh coding provides the state-of-the-art performance [1], [2]. The systematic structure of a semi-regular mesh enables the use of the zero-tree coding and thus supports progressive compression and transmission.

Besides 3-D static meshes, 3-D mesh sequences can be created with animation tools or obtained by capturing dynamic 3-D objects in successive time instances. For example, multiple camera systems can be used to capture real dynamic objects such as facial expressions [3], [4]. A dynamic mesh sequence requires an even larger storage space than a static mesh. Since Lengyel first proposed a mesh sequence coder similar to two-dimensional (2-D) video coding techniques [5], several algorithms have been developed for 3-D mesh sequence coding. In [6]–[8], predictive coding schemes have been proposed, which encode the position or the motion vector of each vertex compactly by exploiting spatial and temporal correlations in mesh sequences. In [9], mesh sequences are encoded based on the principal component analysis (PCA). In [10], the principal components are further analyzed by identifying spatial components and temporal components, and a predictive coding scheme is employed to encode PCA coefficients. In [11], mesh frames are mapped onto 2-D square planes by mesh cutting and parameterization. Then, they are compressed with 2-D video coding techniques. In [12], Gupta *et al.* proposed an algorithm, which partitions each frame into segments, compensates the motion of each segment with an affine mapping, and then encodes the prediction residuals. In [13], Guskov and Khodakovsky proposed a wavelet-based approach to encode mesh sequences progressively.

There are some limitations in the existing algorithms. First, most algorithms [5]–[11], [13] can encode only isomorphic sequences, in which the number of vertices and the connectivity information among vertices are invariant over all frames. They also assume that motion trajectories are already known. Gupta *et al.*'s algorithm [12] is more flexible and can be applied to non-isomorphic sequences. However, when the mesh structure changes substantially between frames, their algorithm spends many bits to encode the connectivity changes and yields even worse performance than static mesh coders. Second, most existing algorithms [5]–[12] do not support progressive coding. In progressive coding, a coarse mesh is first transmitted and rendered, and additional data are then transmitted to refine the mesh successively. Therefore, progressive coding is desirable for the transmission of complex mesh sequences over networks with limited bandwidths. However, most existing algorithms encode a mesh sequence at a single resolution.

In this paper, we propose a progressive coding algorithm for 3-D mesh sequences. Given an irregular sequence with a time-varying structure, we construct a semi-regular mesh for the first frame using remeshing techniques. Then, we map the semi-regular structure to subsequent frames based on the hierarchical motion estimation. The semi-regular mesh sequence is then decomposed into wavelet coefficients, which are compressed progressively by an embedded coding scheme.

The paper is organized as follows. Section II reviews the related work. Section III introduces the semi-regular mesh structure. Sections IV and V describe the motion estimation schemes for base vertices and subdivision points, respectively. Section VI proposes the progressive coder for semi-regular

Manuscript received November 29, 2004; revised September 29, 2005. This work was supported in part by a National Research Laboratory Grant (2000-N-NL-01-C-238) and a Korea University Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Charles D. Creusere.

J.-H. Yang is with LG Electronics, Seoul 151-919, Korea (e-mail: jayu@lge.com).

C.-S. Kim is with the Department of Electronics Engineering, Korea University, Seoul, Korea (e-mail: cskim@ieee.org).

S.-U. Lee is with the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Korea (e-mail: sanguk@ipl.snu.ac.kr).

Digital Object Identifier 10.1109/TIP.2006.877413

mesh sequences. Section VII presents simulation results. Finally, Section VIII concludes this paper.

II. Related Work

The proposed algorithm is related to several research topics: mesh-based 2-D motion estimation, remeshing, and wavelet video coding. Let us review these related works subsequently.

A. Mesh-Based 2-D Motion Estimation

Although block-based motion models have been successfully employed in video compression technology, they cannot represent complex motions effectively and may produce discontinuous motion fields. This problem can be partly overcome by mesh-based motion models [14]. To use a mesh model in the motion-compensated prediction, an initial 2-D mesh is constructed on the first frame. Then, the motion vector of each node point is estimated using the optical flow method or the block matching algorithm. Pixels within each mesh element are then motion-compensated by the affine or bilinear transformation.

Early works employed regular meshes, in which each node is connected to the same number of neighboring nodes. For example, in a regular triangle mesh, each node is connected to six neighboring nodes. While regular meshes are easy to initialize and handle, they cannot adapt to scene contents effectively. To overcome this problem, a mesh element can be divided into smaller elements, if it contains multiple motions [15], [16]. However, the regular mesh model and its hierarchical variation are not sufficient to represent the motions of arbitrary shaped objects.

Content-based meshes have been proposed to track scene features such as object boundaries. Wang and Lee [17] modeled object motion as elastic deformation and tracked object features by minimizing the energy function. The energy function includes the feature term for tracking edges and the matching term to minimize motion estimation errors. Altunbasak and Tekalp [18] investigated the mesh adaptation to object occlusion. Beek *et al.* [19] proposed a hierarchical content-based mesh model. Their method constructs a hierarchical mesh in a fine-to-coarse order, and then tracks object motions in a coarse-to-fine order to obtain robust motion information. This work was extended by Celasun and Tekalp in [20] so that the hierarchical mesh fits object boundaries more precisely.

Similar to these methods, the proposed algorithm tracks the motions of 3-D objects using the mesh structure. However, since we deal with complex 3-D motions of real-world objects, the correspondence matching between frames is a more challenging problem. Moreover, in contrast to the 2-D mesh-based motion models, it will be shown that the regular mesh structure is more suitable for modeling of 3-D objects and their motions.

B. Remeshing

In a 3-D semi-regular triangle mesh, most vertices have degree 6, i.e., most vertices are connected to six adjacent vertices. Remeshing is a resampling process, which converts an irregular mesh into a semi-regular one. In a semi-regular mesh, the whole connectivity structure can be compactly represented, since it is fully determined by the base mesh connectivity and the subdivision level. Therefore, in the semi-regular mesh compression, more bits can be assigned to the geometry information. Since the multiresolution analysis for arbitrary topology surfaces was first introduced by Lounsbery [21], many remeshing algorithms have been proposed, *e.g.*, the algorithms in [22]–[24].

The remeshing process consists of simplification and refinement. In the simplification step, the original irregular mesh is decimated to the base mesh, and the points in the original mesh are mapped onto the corresponding triangles in the base mesh. Then, we can obtain a piecewise linear parameterization, which is a mapping from the set of base triangles to the 3-D surface. In [22], Eck *et al.* constructed the base triangles by partitioning the original mesh using the Voronoi diagram, and computed the parameterization based on the harmonic mapping. In [23], Lee *et al.* proposed a faster parameterization algorithm, which computes the conformal mapping during the simplification.

The base mesh is then refined to be a semi-regular mesh with subdivision connectivity, which is achieved by iteratively subdividing a triangle into four triangles. In the subdivision, the position of each new vertex is determined from the parameterization data. The position can also be predicted from the neighboring vertices in the butterfly configuration [25]. The difference between the original and the predicted positions is called a wavelet coefficient. Thus, the geometry of the semi-regular mesh can be fully specified by that of the base mesh and the hierarchical set of wavelet coefficients.

The normal mesh representation [24] is a special case of the semi-regular mesh representation. In a semi-regular mesh, wavelet coefficients are 3-D vectors in general. On the other hand, in a normal mesh, most wavelet coefficients are represented by scalars. This is possible because each subdivision point is constrained to lie on the normal line, which passes through the butterfly prediction point. Thus, the direction of the prediction error is implicit, and only the magnitude needs to be specified. With this property, the normal mesh representation provides a better coding gain than the general semi-regular mesh representation [2].

In this work, we conduct the remeshing process to obtain the semi-regular mesh sequences for 3-D dynamic objects. In addition to obtaining the semi-regular mesh frame at each time instance, we achieve the vertex correspondences between frames through the motion estimation. This requires us to solve accompanying problems, which will be discussed later.

C. Wavelet Video Coding

In wavelet video coding, video signals are treated as 3-D data (two spatial dimensions + one temporal dimension). If the temporal wavelet transform is applied without the alignment of objects along motion trajectories, highpass subbands contain significant energies and lowpass subbands exhibit undesirable ghost artifacts. Therefore, attempts have been made to compensate object motions prior to the wavelet transform. Early work used the Haar transform with limited global or block motion compensation [26], [27]. Recently, the lifting scheme [28] has drawn a lot of attention as a new wavelet construction method, and the motion-compensated lifting scheme has been successfully employed for the wavelet video coding. In [29], the biorthogonal 5/3 wavelet kernel was used with a mesh-based motion model. In [30], the performances of the Haar and the biorthogonal 5/3 kernels for the motion-compensated lifted wavelet were investigated experimentally and theoretically.

The temporal wavelet transform is followed by the spatial wavelet transform. The wavelet coefficients are then compressed by an entropy coder, which produces an embedded bit-stream with various modes of scalability. Kim *et al.*'s algorithm [31] is based on the SPIHT algorithm and provides the signal-to-noise ratio (SNR) scalability. Similar to JPEG2000 [32], the video coder in [29] supports all the SNR, spatial and temporal scalability modes.

In this work, we also attempt to generate scalable bitstreams for 3-D dynamic objects using the lifting based wavelet transform and the embedded coding.

III. Semi-Regular Mesh Sequence

A. Motivations

Consider a temporal sequence of irregular meshes, $\{\mathcal{I}_n : n = 0, 1, 2, \ldots\}$, which can be synthesized by animation tools or acquired from real 3-D objects with vision techniques. The nth frame $\mathcal{I}_n$ is represented by a set of vertex positions and the connectivity relations between vertices. The mesh structure, including the number of vertices and the connectivity relations, is time-varying in general. Gupta *et al.* [12] proposed a compression algorithm for mesh sequences with time-varying structures, which encodes the changes in the connectivity relations as additional information. However, their algorithm yields good performances only when input mesh sequences experience minor structural changes. For general mesh sequences, their algorithm may consume too many bits for the structural changes. Furthermore, the point correspondences between frames cannot be easily extracted from mesh sequences with time-varying structures.

These observations motivate us to construct a connectivity structure for the first frame and map the same connectivity structure to the following frames using the motion estimation. We adopt the semi-regular structure, which is employed in the state-of-the-art static mesh compression [1], [2].

B. Construction of Initial Semi-Regular Mesh

To obtain a unified connectivity, we convert the first frame $\mathcal{I}_0$ into a semi-regular normal mesh $\mathcal{S}_0 = \langle \mathcal{P}_0, \mathcal{T} \rangle$. $\mathcal{P}_0$ is the set of resampled vertex positions, and $\mathcal{T}$ denotes the unified connectivity for the whole sequence. As mentioned in Section II-B, the remeshing begins with the mesh simplification. We employ the Garland and Heckbert's algorithm [33] to simplify $\mathcal{I}_0$ to a base mesh $\mathcal{S}_0^0$. Concurrently, we map the vertices in $\mathcal{I}_0$ onto the surface of $\mathcal{S}_0^0$ using the conformal mapping in [23].

Fig. 1 shows an example of the remeshing procedure. In the simplification, the vertices within the white curve in Fig. 1(a) are mapped to the dots on the triangle in Fig. 1(b). Then, the base mesh is refined to $\mathcal{S}_0^1$ by dividing each triangle into four triangles as shown in Fig. 1(c). By iteratively applying the subdivisions, we obtain the hierarchy

$$\mathcal{S}_0^0 \subset \mathcal{S}_0^1 \subset \cdots \subset \mathcal{S}_0^{L-1} = \mathcal{S}_0. \tag{1}$$

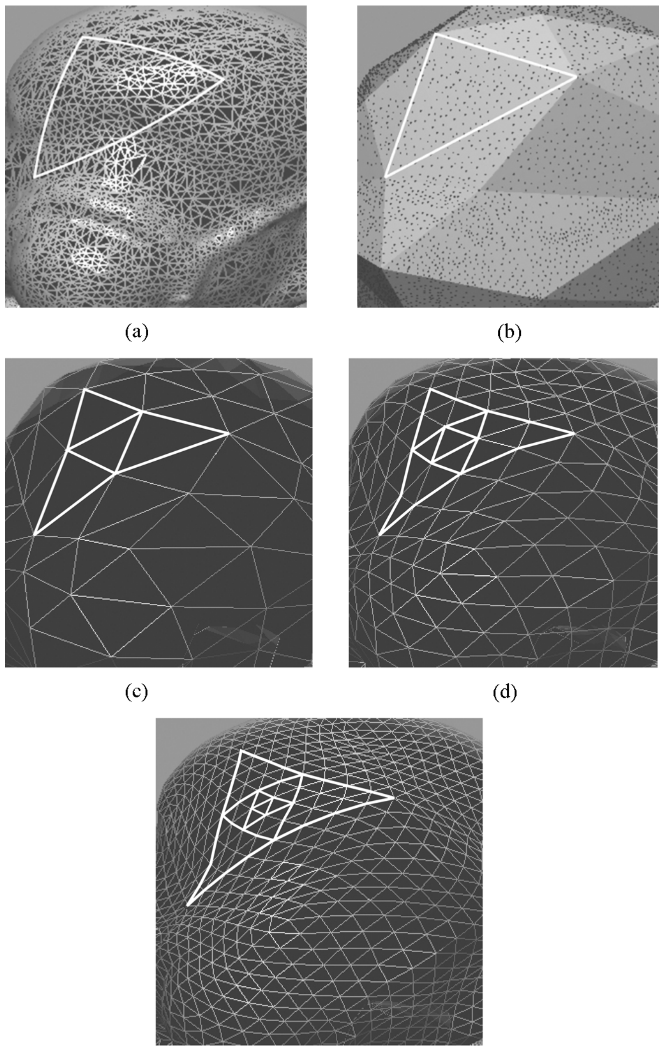

(a) (b) (c) (d) (e)

Fig. 1. Example of remeshing procedure: (a) irregular mesh, (b) base mesh, (c) first subdivision, (d) second subdivision, and (e) third subdivision.

Note that the connectivity $\mathcal{T}$ is perfectly determined by the connectivity of the base mesh and the number of levels L.

The position of a subdivision point $\mathbf{p}^l$ at level l is computed by the butterfly prediction and the normal piercing scheme, as shown in Fig. 2. First, its prediction $\mathbf{p}_b$ is obtained from the butterfly neighbors $\mathbf{p}_i^{l-1}$ $(i = 0, 1, \ldots, 7)$ at level $l-1$. Then, $\mathbf{p}_b$ is translated along the normal direction to find the piercing point $\mathbf{p}^l$ on the original surface, which becomes the new subdivision point. In rare cases, piercing points may not be valid and can distort the shape of the original mesh. As in [24], we first perform the validity test. Then, if a piercing point is declared to be invalid, we find an alternative remeshing point in the parametric domain without the constraint of the normal direction. The wavelet coefficient $\psi_{\mathbf{p}^l}$ for $\mathbf{p}^l$ is then defined as

$$\psi_{\mathbf{p}^l} = \mathbf{p}^l - \mathbf{p}_b. \tag{2}$$

Therefore, the geometry information of the hierarchy in (1) can be represented by the base mesh points and the set of wavelet coefficients.

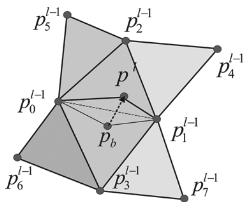
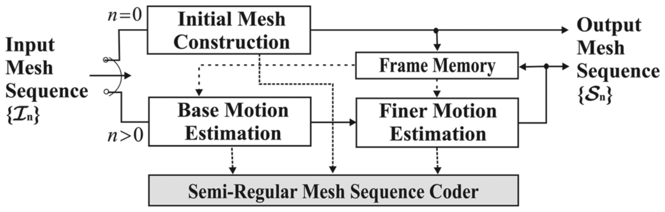


Fig. 2. Mesh refinement in a regular mesh.



Input Mesh Sequence $\{\mathcal{I}_n\}$
$n=0$
Initial Mesh Construction
$n>0$
Base Motion Estimation
Frame Memory
Finer Motion Estimation
Output Mesh Sequence $\{\mathcal{S}_n\}$
Semi-Regular Mesh Sequence Coder

Fig. 3. Block diagram of the proposed algorithm.

C. Mapping of Semi-Regular Mesh Structure

The unified connectivity $\mathcal{T}$ is mapped to the following frames, which is achieved by the hierarchical motion estimation. Fig. 3 shows the overall block diagram of the proposed algorithm. In the base motion estimation block, we track the movements of vertices in the base mesh. Then, in the finer-level motion-estimation block, we predict and refine the motion vectors of subdivision points level by level. Finally, the semi-regular mesh sequence is compactly encoded by the coding block. Let us describe these operational blocks subsequently in the following sections.

IV. Base Motion Estimation

Given the previous semi-regular mesh $\mathcal{S}_{n-1}$, we construct the base mesh $\mathcal{S}_n^0$ of the current frame by estimating the motion from $\mathcal{S}_{n-1}^0$ to the surface of the input irregular mesh $\mathcal{I}_n$.

A. Joint Mesh Segmentation and Motion Estimation

In 2-D video coding, the region-based motion estimation was introduced to describe dissimilar motions of multiple objects more accurately [14]. The region segmentation and the motion estimation are jointly performed so that the motion of each segment is represented accurately with a parametric model. The joint optimization is often performed by applying the region segmentation and the motion estimation alternately.

We extend this idea to perform the 3-D motion estimation. To reduce computational complexity, $\mathcal{I}_n$ is represented at multiple levels of detail, given by

$$\mathcal{I}_n^0 \subset \mathcal{I}_n^1 \subset \cdots \subset \mathcal{I}_n^{L'-1} = \mathcal{I}_n \tag{3}$$

where L' is less than or equal to L in (1), and the number of triangles in $\mathcal{I}_n^l$ is about one fourth of that in $\mathcal{I}_n^{l+1}$.

Beginning with the whole base mesh $\mathcal{S}_{n-1}^0$ as an initial segment, we compute its motion parameters. The motion parameters T_s of a segment are composed of the rotation matrix $\mathbf{R}$ and

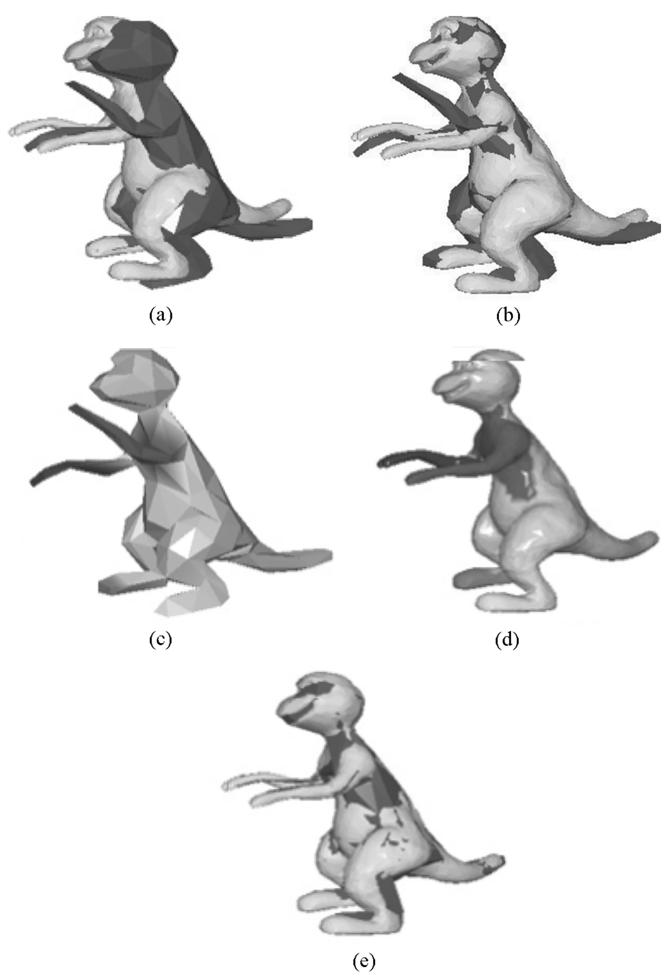


Fig. 4. Joint mesh segmentation and motion estimation: (a) initial alignment, (b) first iteration, (c) segmentation of $\mathcal{S}_{n-1}^0$, (d) motion search regions in $\mathcal{I}_n^0$, and (e) final iteration.

the translation vector $\mathbf{t}$, which are computed by a modified version of the ICP algorithm [34] as follows:

$$T_s = \arg\min_T \sum_{\mathbf{p}\in\mathcal{V}} w_{\mathbf{p}} \|\mathbf{q} - T(\mathbf{p})\|^2 \tag{4}$$

where $T(\mathbf{p}) = \mathbf{R}\mathbf{p} + \mathbf{t}$ and $\mathcal{V}$ is the set of vertex positions in the segment. The point $\mathbf{q}$ is selected from $\mathcal{I}_n^0$ such that it is closest to $T(\mathbf{p})$. The weight $w_{\mathbf{p}}$ denotes the reliability of $\mathbf{p}$ in the computation of the motion parameter, which is computed based on the similarity of the normals at $T(\mathbf{p})$ and $\mathbf{q}$.

Fig. 4 illustrates the procedure of the joint mesh segmentation and motion estimation. Fig. 4(a) shows $\mathcal{S}_{n-1}^0$ in dark gray and $\mathcal{I}_n^0$ in light gray. After the motion estimation, they are aligned as shown in Fig. 4(b). The object has dissimilar motions, which cannot be well described by a single rigid motion, especially around the arms, legs and tail. These outliers are extracted based on the matching distances. When the distance between $\mathbf{q}$ and $T(\mathbf{p})$ is larger than a threshold, the point $\mathbf{q}$ becomes an outlier. The outlier points are processed by a merging-splitting-merging procedure to yield compactly connected segments.

In the merging procedure, a segment with a small number of vertices is merged with near segments. In the splitting procedure, if a segment contains a vertex cut [35], whose removal disconnects the segment, it is divided into two segments. The merging procedure is performed once more to yield final segments. Fig. 4(c) shows new segments in different gray levels. After selecting a segment with the maximum matching distance, we constrain its search region in $\mathcal{I}_n^0$ for reliable motion estimation. When an object moves non-rigidly, it is unreliable to find the motion in the whole region of $\mathcal{I}_n^0$. We first extend the boundary of the segment by one triangle strip. Then, the matching points of the extended boundary points confine the search region in $\mathcal{I}_n^0$, which is shown in Fig. 4(d) for each segment. We alternate the segmentation and the motion estimation of segments, until the decrease in the overall matching distance becomes negligible. After the final iteration, the estimated base mesh is well fitted to the current geometry $\mathcal{I}_n^0$, as shown in Fig. 4(e).

B. Vertex-Wise Motion Refinement

The joint mesh segmentation and motion estimation with the ICP algorithm can deal with only piecewise rigid motions. Therefore, the motion vector of each vertex in the base mesh should be refined to express flexible motions. As mentioned before, a point $\mathbf{p}$ in $\mathcal{S}_{n-1}^0$ is moved to $\bar{\mathbf{q}} = T(\mathbf{p})$ in (4) and then mapped to the closest point $\mathbf{q}$ on $\mathcal{I}_n^0$. In this refinement step, the position of $\mathbf{q}$ is refined to a position in $\mathcal{I}_n^1$, then in $\mathcal{I}_n^2$, and so on. The final refined positions in $\mathcal{I}_n^{L'-1}$ form the base mesh $\mathcal{S}_n^0$ of the current frame.

Since the refinement in each level mesh $\mathcal{I}_n^l$ is done in the same way, we describe the refinement of $\mathbf{q}$ without specifying the level index l for the simplicity of notations. Let $\mathbf{q} \in \mathcal{S}_n^0$ denote the matched point of $\mathbf{p} \in \mathcal{S}_{n-1}^0$ to the end of this section. We refine the positions of the matched points to minimize the energy function

$$\begin{aligned} E(\mathcal{S}_n^0) &= w_s E_s(\mathcal{S}_n^0) + w_m E_m(\mathcal{S}_n^0) + w_d E_d(\mathcal{S}_n^0) \\ &= \sum_{\mathbf{q} \in \mathcal{S}_n^0} \{ w_s E_s(\mathbf{q}) + w_m E_m(\mathbf{q}) + w_d E_d(\mathbf{q}) \} \end{aligned} \tag{5}$$

where w_s, w_m, and w_d are weighting coefficients. The first term $E_s(\mathbf{q})$ represents the temporal change in the spring energies of the edges which are incident to $\mathbf{q}$, given by

$$E_s(\mathbf{q}) = \sum_{\mathbf{r}_1 \in \mathcal{O}(\mathbf{q})} \frac{1}{N_r} (\|\mathbf{q} - \mathbf{r}_1\| - \|\mathbf{p} - \mathbf{r}_2\|)^2$$

where $\mathcal{O}(\mathbf{q})$ is the 1-ring of $\mathbf{q}$, i.e., the set of vertices adjacent to $\mathbf{q}$. N_r is the size of $\mathcal{O}(\mathbf{q})$, and $\mathbf{r}_2$ is the point in $\mathcal{S}_{n-1}^0$ that corresponds to $\mathbf{r}_1$. The second term $E_m(\mathbf{q})$ is introduced to preserve the surface shapes, defined as

$$E_m(\mathbf{q}) = (1 - \mathbf{n} \cdot \bar{\mathbf{n}}) + (1 - \mathbf{c} \cdot \bar{\mathbf{c}})$$

where $\bar{\mathbf{n}}$ and $\mathbf{n}$ are the normals of $\bar{\mathbf{q}}$ and $\mathbf{q}$, respectively. Similarly, $\bar{\mathbf{c}}$ and $\mathbf{c}$ are the principal curvatures of $\bar{\mathbf{q}}$ and $\mathbf{q}$. The final term $E_d(\mathbf{q})$ is adopted to fit the base mesh to the current frame with little distortion, given by

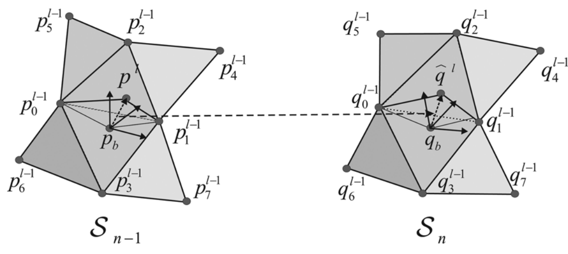


Fig. 5. Motion interpolation.

$$E_d(\mathbf{q}) = \sum_{\mathbf{r} \in \mathcal{O}(\mathbf{q})} \frac{1}{N_r} \left\| \frac{\mathbf{q} + \mathbf{r}}{2} - \mathbf{s} \right\|^2$$

where $\mathbf{s}$ denotes the closet position on $\mathcal{I}_n^l$ from the mid-point $(\mathbf{q} + \mathbf{r})/2$. The n-ring of $\mathbf{q}$ in $\mathcal{I}_n^l$ is the set of candidate points for the updated $\mathbf{q}$, where the size n decreases as the iteration goes on. The three energy terms are computed for each candidate point. Then, $\mathbf{q}$ is moved to the point that yields the maximum energy reduction, which guarantees that the energy function in (5) decreases monotonically.

To summarize, the base motion estimation is done first for the whole base mesh, then at the segment level, and finally at the vertex level. This enables us to obtain a robust and smooth vertex-wise motion vector field.

V. Finer-Level Motion Estimation

After the base motion estimation, we have the previous semi-regular mesh $\mathcal{S}_{n-1}$ and the current base mesh $\mathcal{S}_n^0$. In the finer-level motion estimation, the position of each subdivision point $\mathbf{q}^l$ in $\mathcal{S}_n^l$ $(1 \le l < L)$ is determined using the motion interpolation and the parameterization.

A. Motion Interpolation

The position of $\mathbf{q}^l$ can be predicted based on the motions of lower level points in $\mathcal{S}_n^{l-1}$. Let $\hat{\mathbf{q}}^l$ denote the predicted position of $\mathbf{q}^l$. Then, as illustrated in Fig. 5, $\hat{\mathbf{q}}^l$ is computed by

$$\begin{aligned} \hat{\mathbf{q}}^l &= \sum_{i=0}^{7} w_i \mathbf{q}_i^{l-1} + C\left(\mathbf{p}^l - \sum_{i=0}^{7} w_i \mathbf{p}_i^{l-1}\right) \\ &= \mathbf{q}_b + C(\mathbf{p}^l - \mathbf{p}_b) \end{aligned} \tag{6}$$

where $\mathbf{q}_b = \sum_{i=0}^{7} w_i \mathbf{q}_i^{l-1}$ and $\mathbf{p}_b = \sum_{i=0}^{7} w_i \mathbf{p}_i^{l-1}$. Also, w_i is the ith butterfly filter coefficient, and $\mathbf{p}_i^{l-1} \in \mathcal{S}_{n-1}^{l-1}$ and $\mathbf{q}_i^{l-1} \in \mathcal{S}_n^{l-1}$ are the ith butterfly points for $\mathbf{p}^l$ and $\mathbf{q}^l$, respectively. $C(\cdot)$ is the coordinate transform from the local coordinate at $\mathbf{p}_b$ to that at $\mathbf{q}_b$. The local coordinates are defined such that z-axes are along the surface normal directions at $\mathbf{p}_b$ and $\mathbf{q}_b$. The tangential axes are determined also using the lower level information. From (6), $\hat{\mathbf{q}}^l - \mathbf{q}_b = C(\mathbf{p}^l - \mathbf{p}_b)$. Therefore, $\hat{\mathbf{q}}^l$ and $\mathbf{p}^l$ are, respectively, located at the same position in the local coordinates at $\mathbf{q}_b$ and $\mathbf{p}_b$. This means that $\hat{\mathbf{q}}^l$ is equal to $\mathbf{q}^l$, provided

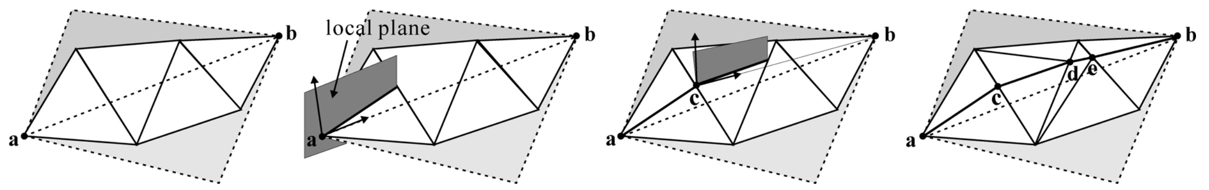


Fig. 6. Boundary mapping. Dashed lines depict the edges of the base mesh $\mathcal{S}_n^0$, while solid lines depict the edges of the original irregular mesh $\mathcal{I}_n$.

that the object moves rigidly and the motion of the base mesh is estimated accurately.

Notice that, in (2), the wavelet coefficient $\psi_{\mathbf{p}^l}$ for $\mathbf{p}^l$ is defined as the difference $\mathbf{p}^l - \mathbf{p}_b$. Thus, (6) indicates that the position of $\mathbf{q}^l$ is predicted using the butterfly prediction $\mathbf{q}_b$ and the wavelet coefficient $\psi_{\mathbf{p}^l}$ in the previous frame. In other words, both the spatial and the temporal correlations are exploited to predict $\mathbf{q}^l$ in this work.

B. Parameterization

The motion-interpolated subdivision point $\hat{\mathbf{q}}^l$ in (6) is mapped onto the surface of $\mathcal{I}_n$ to complete the finer-level motion estimation. However, we need the parameterization of $\mathcal{I}_n$ for robust motion estimation, since a naive mapping may yield severe geometrical distortions.

We construct a piecewise linear parameterization, which maps a triangular patch in $\mathcal{I}_n$ onto a base triangle in $\mathcal{S}_n^0$. The parameterization for the first frame $\mathcal{I}_0$ is built during the mesh simplification in the initial semi-regular mesh construction, as shown in Fig. 1(a) and (b). In contrast, the parameterization for $\mathcal{I}_n$ $(n > 0)$ is not available immediately after the base motion estimation. Therefore, we should find the triangular patch in $\mathcal{I}_n$ that maps onto each base triangle in $\mathcal{S}_n^0$.

Base edges are mapped onto the surface of $\mathcal{I}_n$ by a boundary mapping scheme shown in Fig. 6. The base edge $\overline{\mathbf{ab}}$ is mapped to the boundary $\overline{\mathbf{acdeb}}$, which consists of intersecting line segments between the original irregular mesh $\mathcal{I}_n$ and the local plane. Initially, the local plane includes the vertex $\mathbf{a}$ and is parallel with the normal vector at $\mathbf{a}$ and the directional vector $\overrightarrow{\mathbf{ab}}$. The boundary point $\mathbf{c}$ is determined by finding the intersecting point of the local plane. Then, the local plane is updated to be parallel with the normal vector at $\mathbf{c}$ and the directional vector $\overrightarrow{\mathbf{cb}}$, and the boundary point $\mathbf{d}$ is found. Similarly, another boundary point $\mathbf{e}$ is found. As shown in Fig. 6, the boundary points introduce additional edges and triangles on the irregular mesh.

Occasionally, the boundary mapping scheme produces intersecting boundaries. For example, in Fig. 7(b), black and white boundaries intersect each other. In this case, their upper parts are interchanged to yield the configuration in Fig. 7(c). Then, we compute a harmonic mapping from the rectangular patch, depicted by the black curve in Fig. 7(d), to the base mesh. Using the mapping, the white boundary is recomputed more reliably.

After the boundary mapping, we compute the piecewise linear parameterization, which maps the vertices in a triangular patch onto the corresponding base triangle using a harmonic mapping.

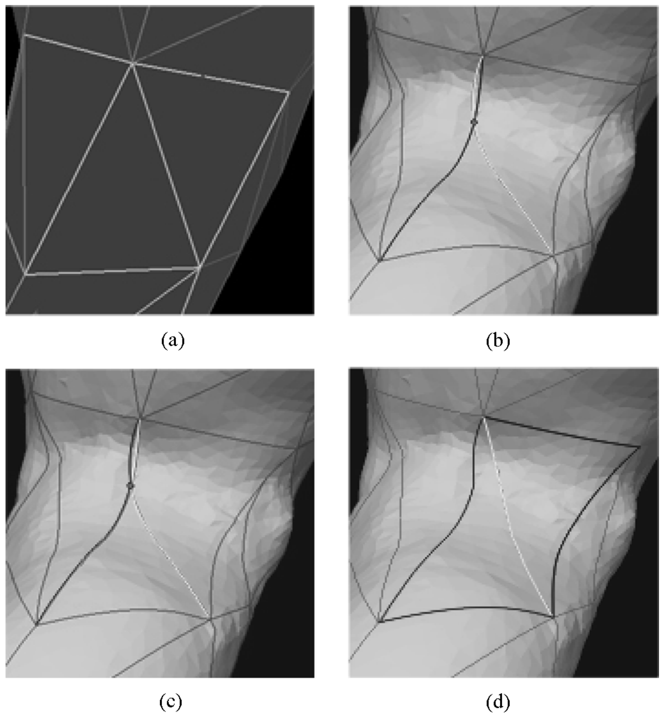


Fig. 7. Exception handling in the boundary mapping: (a) base mesh, (b) intersecting boundaries, (c) rearrangement of the boundaries, and (d) boundary remapping.

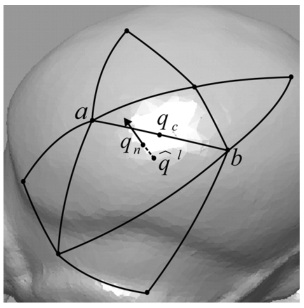
C. Determination of Subdivision Points

As shown in Fig. 8(a), there are two candidates for the subdivision point $\mathbf{q}^l$. One is $\mathbf{q}_n$ that is obtained by the normal piercing from the predicted point $\hat{\mathbf{q}}^l$ in (6). The other is the parametric center point $\mathbf{q}_c$ of the chord $\widehat{\mathbf{ab}}$. As in [24], the normal piercing should be validated in the parameter domain for robustness. Fig. 8(b) is the 2-D parameter domain of two triangular patches in Fig. 8(a). If $\mathbf{q}_n$ is within the circle of center $\mathbf{q}_c$ and radius r, the normal piercing is declared as valid and $\mathbf{q}_n$ is selected as the final position of $\mathbf{q}^l$. Then, the parameter domain is adjusted by setting the $\mathbf{q}_n$ as a new center point. If $\mathbf{q}_n$ is outside the circle, the normal piercing point may cause undesirable artifacts such as folding of mesh surfaces. Thus, in such case, $\mathbf{q}_c$ is selected as $\mathbf{q}^l$.

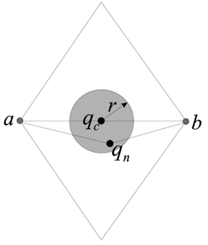
The position of $\mathbf{q}^l$ can be compactly represented by subtracting the predicted position $\hat{\mathbf{q}}^l$ in (6). From (2) and (6), the residual vector $\mathbf{q}^l - \hat{\mathbf{q}}^l$ can be expressed as

$$\mathbf{q}^l - \hat{\mathbf{q}}^l = \psi_{\mathbf{q}^l} - C(\psi_{\mathbf{p}^l}) \tag{7}$$

which is then encoded by the following coding scheme.



(a)



(b)

Fig. 8. Selection of a subdivision point: (a) normal piercing and (b) validity check.

VI. Progressive Coding of Semi-Regular Mesh Sequence

In this section, we discuss how to decompose a semi-regular mesh sequence in the spatiotemporal domain and compress it with a progressive coder.

A. Spatiotemporal Decomposition

Each frame in the semi-regular mesh sequence is decomposed into base mesh points and wavelet coefficients for subdivision points. Note that in (2), each subdivision point is predicted from the butterfly prediction using lower level points to get a wavelet coefficient. This decomposition can be seen as a lifting scheme without updating [2], [28]. The wavelet coefficients are further decomposed in the temporal domain along motion trajectories.

Suppose that a temporal sequence of wavelet coefficients along a motion trajectory, $\{\psi_{\mathbf{p}_n}\}$, is divided into the even subsequence $\{\psi_{\mathbf{p}_{2k}}\}$ and the odd subsequence $\{\psi_{\mathbf{p}_{2k+1}}\}$, where n, $2k$ and $2k+1$ are time indices. Then, the temporal lifting decomposition can be written as

$$\begin{aligned} \mathbf{h}_k &= \psi_{\mathbf{p}_{2k+1}} - P(\{\boldsymbol{\psi}_{\mathbf{p}_{2k}}\}) \\ \mathbf{l}_k &= \psi_{\mathbf{p}_{2k}} + U(\{\mathbf{h}_k\}) \end{aligned} \tag{8}$$

where $\mathbf{h}_k$ and $\mathbf{l}_k$ are the high-pass and the low-pass terms, respectively. P is the prediction function using the even subsequence. U is the updating function using the highpass terms. For example, when $P(\{\boldsymbol{\psi}_{\mathbf{p}_{2k}}\}) = \psi_{\mathbf{p}_{2k}}$ and $U(\{\mathbf{h}_k\}) = 0.5\mathbf{h}_k$, (8) becomes the Haar decomposition. To exploit the temporal correlation more effectively, we modify the prediction function based on the motion-compensated lifting [29], so that

$$\mathbf{h}_k = \psi_{\mathbf{p}_{2k+1}} - C_{\mathbf{p}_{2k},\mathbf{p}_{2k+1}}(\psi_{\mathbf{p}_{2k}}) \tag{9}$$

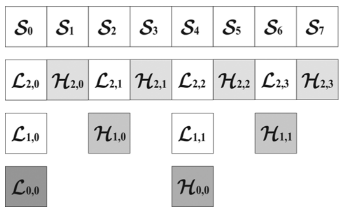


Fig. 9. Example of temporal decomposition.

where $C_{\mathbf{p}_{2k},\mathbf{p}_{2k+1}}$ denotes the coordinate transform from the local coordinate at the butterfly prediction point of $\mathbf{p}_{2k}$ to that of $\mathbf{p}_{2k+1}$. In other words, the wavelet coefficient $\psi_{\mathbf{p}_{2k}}$ is motion-compensated and then subtracted from $\psi_{\mathbf{p}_{2k+1}}$ to generate the highpass term $\mathbf{h}_k$. Note that (9) is essentially the same equation as (7) except for different notations. The updating function can be similarly modified as $U(\{\mathbf{h}_k\}) = 0.5\mathcal{C}_{\mathbf{p}_{2k+1},\mathbf{p}_{2k}}(\mathbf{h}_k)$. However, we observed that the updating incurs visual artifacts and large distortions. Thus, we do not employ the updating function and set $U(\{\mathbf{h}_k\}) = 0$.

While the Haar-based decomposition uses the unidirectional motion compensation, a bidirectional motion-compensation scheme can be derived from the biorthogonal 5/3 filter [29], given by

$$\mathbf{h}_k = \psi_{\mathbf{p}_{2k+1}} - 0.5\left[\mathcal{C}_{\mathbf{p}_{2k},\mathbf{p}_{2k+1}}(\psi_{\mathbf{p}_{2k}}) + \mathcal{C}_{\mathbf{p}_{2k+2},\mathbf{p}_{2k+1}}(\psi_{\mathbf{p}_{2k+2}})\right] \tag{10}$$

where $\psi_{\mathbf{p}_{2k+1}}$ is predicted from the average of the forwardly motion-compensated $\psi_{\mathbf{p}_{2k}}$ and the backwardly motion-compensated $\psi_{\mathbf{p}_{2k+2}}$.

The temporal filtering decomposes a semi-regular mesh sequence into the low-pass subband and the high-pass subband. This is repeatedly applied to low-pass subbands. For example, in Fig. 9, eight consecutive mesh frames form a group of frames (GOF). They are first decomposed into four low-pass frames $\{\mathcal{L}_{2,n}\}$ and four high-pass frames $\{\mathcal{H}_{2,n}\}$. Then, the four low-pass frames $\{\mathcal{L}_{2,n}\}$ are further decomposed into $\{\mathcal{L}_{1,n}\}$ and $\{\mathcal{H}_{1,n}\}$. After one more decomposition, we can obtain the lowest frequency frame $\mathcal{L}_{0,0}$ and a set of hierarchical high-pass frames.

B. Progressive Coding

We propose a progressive coding algorithm that compresses the base mesh sequence and the decomposed subbands. The proposed algorithm supports both SNR and temporal scalability modes.

The base mesh sequence is isomorphic. For its compression, we adopt the isomorphic sequence coder in [7], which is based on the vertex-wise motion vector prediction. In [7], each frame is predicted from the previous frame. However, in this work, the order of the motion predictions is modified to be compatible with the temporal decomposition in the previous subsection. Fig. 10 shows the order of the motion predictions, when a GOF, consisting of eight frames, is decomposed into three levels of temporal subbands as in Fig. 9. The solid and the dotted arrows denote the forward and the backward predictions, respectively.

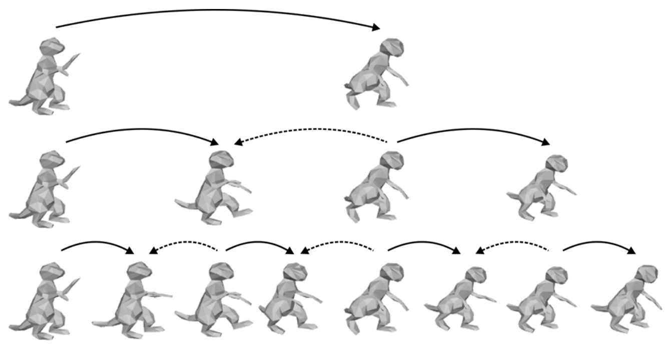


Fig. 10. Order of motion predictions in the base mesh sequence coding.

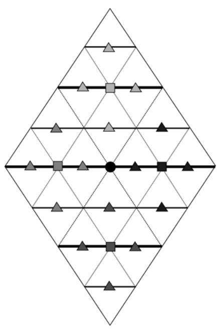


Fig. 11. Parent-child relations among subdivision points in a unit tree.

The proposed algorithm encodes each temporal subband independently of the other subbands to support temporal scalability. Thus, in the example of Fig. 9, the proposed algorithm can provide three different frame rates. After receiving two subbands $\mathcal{L}_{0,0}$ and $\mathcal{H}_{0,0}$, the decoder can reconstruct the sequence at the lowest frame rate, as shown at the top row of Fig. 10. The decoder can double the frame rate when it receives the subbands $\mathcal{H}_{1,0}$ and $\mathcal{H}_{1,1}$. Finally, after receiving all the remaining subbands, it can reconstruct the sequence at the full frame rate.

Each subband is encoded by the SPIHT algorithm, which was originally developed for 2-D image coding [36] and later applied to the subdivision geometry coding of static meshes [1], [2]. It is a zero-tree coding scheme based on the hierarchical parent-child relations among wavelet coefficients. Fig. 11 shows the parent-child relations among the subdivision points (or corresponding wavelet coefficients) in a semi-regular mesh. The first subdivision point, which corresponds to the midpoint on a base edge, becomes the root node. It is depicted by a black circle in Fig. 11. It has four children depicted by squares, each of which has four children depicted by triangles. These subdivision points constitute a unit tree. Thus, there are as many unit trees as the number of edges in the base mesh. In the semi-regular mesh sequence, most wavelet coefficients have only one component in the normal direction of the surface. However, a few exceptional wavelet coefficients have non-zero tangential components. Thus, we employ three SPIHT coders: one for the normal component and two for the tangential components. The bitstreams from these three coders are interweaved into a single bitstream.

The bitstream for each subband is divided into packets to support SNR scalability as well. Specifically, the last bit for each bit plane information from the SPIHT coder is defined as the truncation point for a packet. Let R_n denote the cumulative number of bits in the first n packets, and D_n denote the distortion of the subband reconstructed using the first n packets. The nth packet is then associated with the incremental rate ΔR_n $(= R_n - R_{n-1})$ and the decremental distortion ΔD_n $(= D_n - D_{n-1})$. The ratio, $-\Delta D_n / \Delta R_n$, is referred to as the rate-distortion slope S_n. A larger slope indicates that the corresponding packet is more valuable in the rate-distortion sense. If S_n is larger than S_{n-1}, the nth packet is merged with the $(n-1)$th packet to guarantee the convex rate-distortion curve [37].

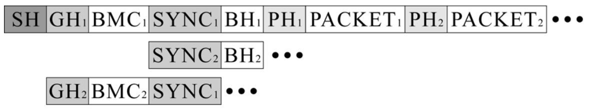


Fig. 12. Organization of the compressed data. SH: sequence header. GH: GOF header; BMC: data for base meshes; SYNC: synchronization code for a subband; BH: subband header; PH: packet header; PACKET: packet data.

Fig. 12 shows the hierarchical organization of the compressed data for a whole mesh sequence. A sequence consists of GOFs, and each GOF is decomposed into subbands. Finally, each subband is composed of several packets. Thus, a packet is the smallest decoding unit and its header (PH) contains the packet size, the rate-distortion slope, and the index of the subband to which the packet belongs. Note that a subband can be decoded independently of the other subbands. Therefore, the transmission order of packets can be flexibly determined according to the request from the receiver. For example, for each subband, the encoder can select the number of packets to be transmitted, achieving a coarse SNR scalability. It is worthy to point out that a finer SNR scalability can be supported by the independent coding of each unit tree [37] at the cost of additional overhead bits.

In this work, the frame rate or temporal resolution is first determined. This determines the temporal subbands to be transmitted. Then, given the bit budget, the packets from those subbands are transmitted in the decreasing order of their rate-distortion slopes. This approach provides a good rate-distortion performance.

C. Distortion Model

The quantization of wavelet coefficients produces the distortion between the original geometry and the reconstructed one. Since the spatiotemporal decomposition in Section VI-A is not an orthogonal transform, the mean square error of the wavelet coefficients is different from the geometry distortion (i.e., the mean square error of reconstructed points). Thus, we assign weights to the wavelet coefficient errors to approximate the distortion of the reconstructed geometry. In [37], the weights were computed by assuming the independence of neighboring vertex errors. However, more accurate weights can be computed from the impulse-response simulation [38], in which an impulse is invoked at a remeshing point and its propagation to finer remeshing levels is computed.

The synthesis in the spatial domain is performed with the butterfly filter. Thus, even if only a single wavelet coefficient has a nonzero distortion, the neighboring subdivision points

TABLE I
DATA STRUCTURES OF TEST SEQUENCES

Test sequence	Number of frames	Average number of vertices in a frame	Average number of triangles in a frame	Uncompressed file size (Kbytes/frame)
Dancer	8	43,025	86,044	1,549
Dolphin	16	48,266	96,528	1,738
Dinosaur	16	54,955	109,905	1,978

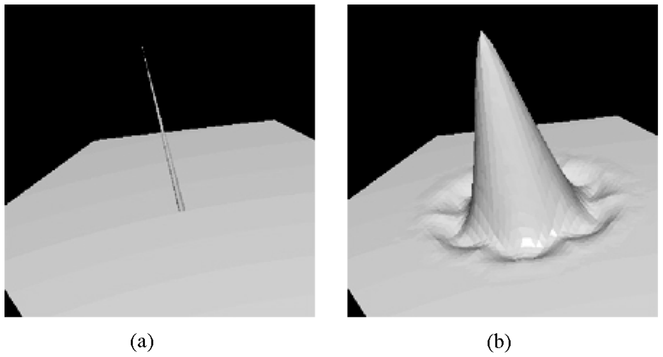


Fig. 13. Error propagation in a semi-regular mesh: (a) an impulse at level 1 and (b) its response at the finest level 4.

are reconstructed erroneously due to the recursive butterfly filtering. For example, suppose that a semi-regular mesh has four remeshing levels, and a point at level 1 is associated with a unit error as illustrated in Fig. 13(a). Then, it propagates to higher remeshing levels and its response at the finest level 4 is shown in Fig. 13(b). Therefore, the distortion of each wavelet coefficient at level 1 should be weighted by the squared sum of the propagated errors in Fig. 13(b). The weights for the other levels can be similarly computed by the impulse-response simulation. When a semi-regular mesh has four remeshing levels, the weights are given by

$$W_1 = 160.62,\ W_2 = 40.16,\ W_3 = 10.08,\ \text{and}\ W_4 = 1$$

where W_i denotes the weight for a wavelet coefficient at level i. In implementation, wavelet coefficients at level i are multiplied by $\sqrt{W_i}$ before the bit-plane encoding, and divided by the same factor $\sqrt{W_i}$ at the decoder side.

VII. Simulation Results

To obtain test sequences, we made clay models which have bones and joints. We animated the models by hands, and captured each frame with the Cyberware 3030MS scanner. Three mesh sequences "dancer," "dolphin," and "dinosaur" were acquired and used to evaluate the proposed algorithm. The acquired sequences are not isomorphic, thus each frame has a different number of vertices and different connectivity relations. Table I summarizes the data structure of the test sequences. These sequences are converted into semi-regular mesh sequences. The first frame of each sequence is simplified to a base mesh consisting of 177 vertices and 350 triangles. Then, the base mesh is recursively subdivided four times and refined to a semi-regular normal mesh, which is composed of 44 802 vertices and 89 600 triangles. Then, the following frames are converted into semi-regular normal meshes of the same connectivity based on the base motion estimation and the finer motion estimation.

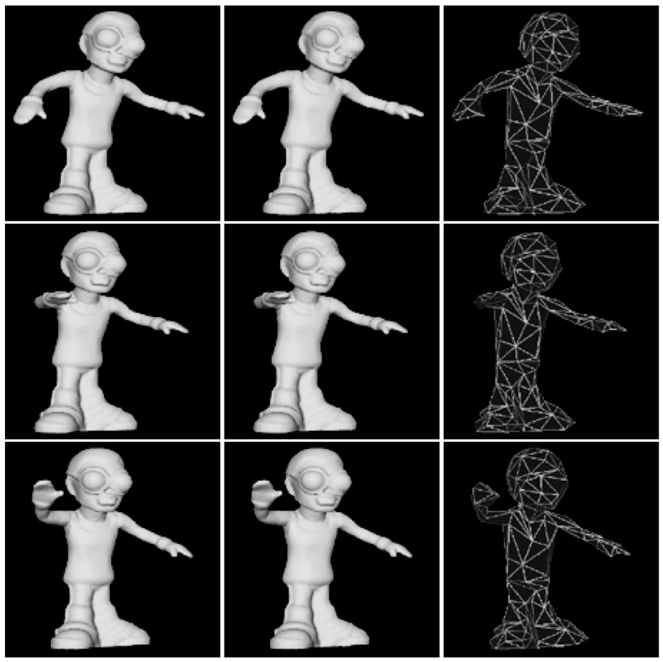


Fig. 14. "Dancer" sequence. From top to bottom, the first, fourth, and eighth frames. From left to right, the original irregular meshes, the semi-regular meshes, and the base meshes.

Figs. 14–16 compare the original irregular meshes and the semi-regular meshes. The base meshes for the semi-regular frames are also shown in the rightmost columns. We can see that the irregular sequences are converted into the semi-regular mesh sequences very faithfully. To evaluate the accuracy of the mesh conversion objectively, we compute the remeshing error of each frame mesh using the "Metro" tool in [39]. Let $\mathcal{A}$ and $\mathcal{B}$ be two meshes. The directed distance from $\mathcal{A}$ to $\mathcal{B}$, denoted by $h(\mathcal{A},\mathcal{B})$, is defined as the mean of squared distances from sampled points on $\mathcal{A}$ to their closest points on $\mathcal{B}$. Then, the Metro algorithm defines the distance between $\mathcal{A}$ and $\mathcal{B}$ as

$$H(\mathcal{A},\mathcal{B}) = \frac{\max\{h(\mathcal{A},\mathcal{B}), h(\mathcal{B},\mathcal{A})\}}{L}$$

where L is the length of the bounding box diagonal. In this work, for each sequence, L is fixed to the diagonal length of the first frame. Fig. 17 shows the remeshing errors of the test sequences. It is observed that the remeshing errors do not accumulate along frames.

Figs. 18–20 show the compression performances on the three test sequences. In this test, each GOF consists of eight frames and is decomposed into three levels of subbands as shown in

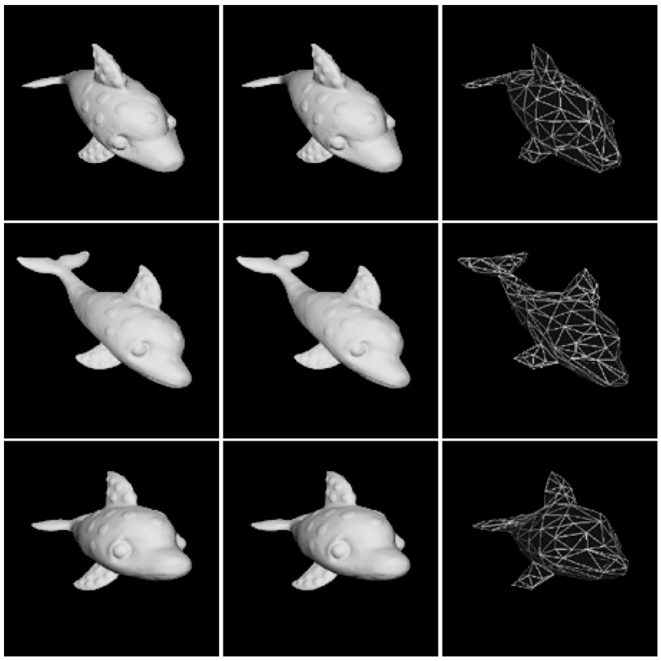


Fig. 15. "Dolphin" sequence. From top to bottom, the first, eighth and sixteenth frames. From left to right, the original irregular meshes, the semi-regular meshes, and the base meshes.

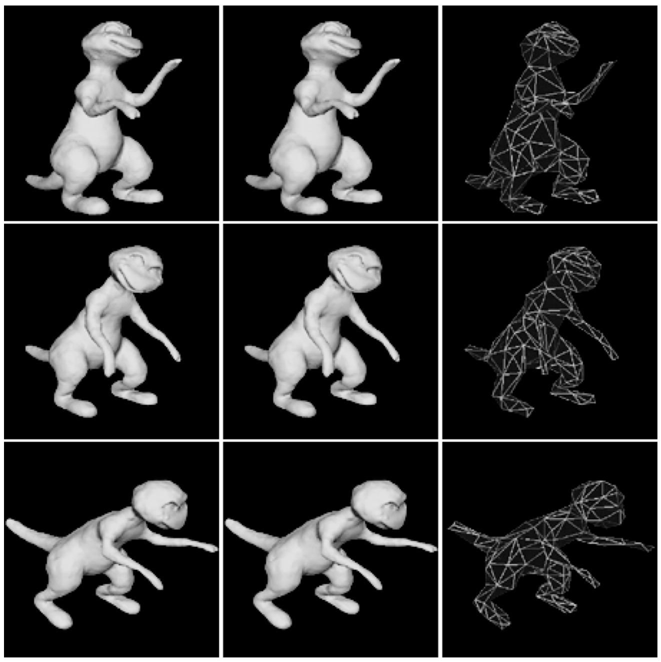


Fig. 16. "Dinosaur" sequence. From top to bottom, the the first, eighth and sixteenth frames. From left to right, the original irregular meshes, the semi-regular meshes, and the base meshes.

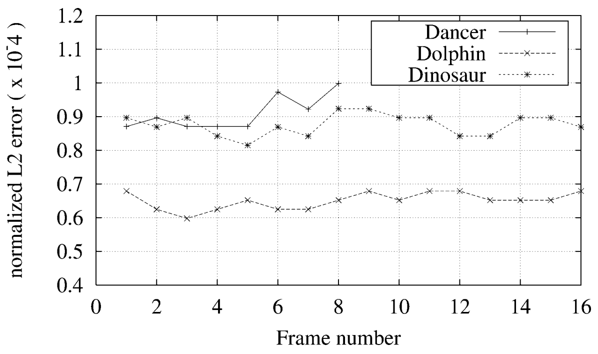


Fig. 17. Remeshing errors of the test sequences.

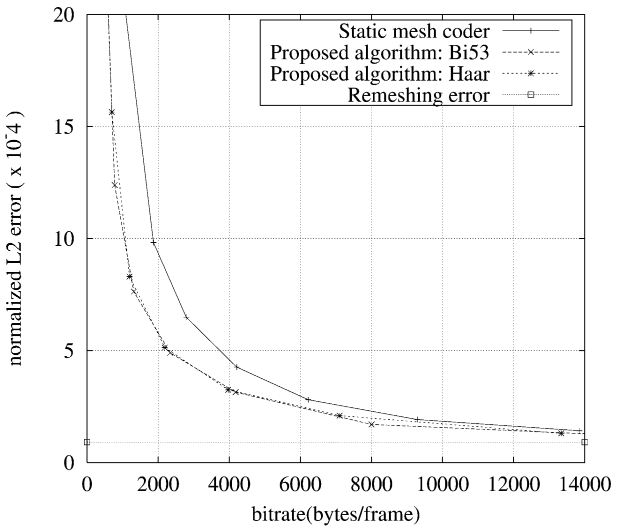


Fig. 18. Compression performances on the "Dancer" sequence. "Bi53" denotes that the biorthogonal 5/3 filter is used for the temporal decomposition, and "Haar" denotes the Haar filter.

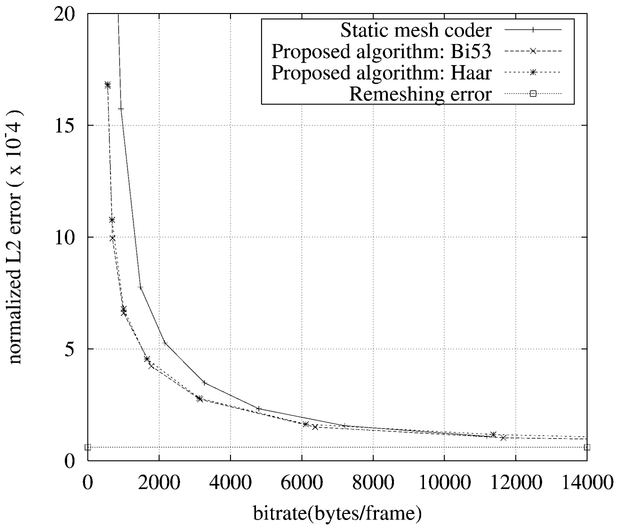


Fig. 19. Compression performances on the "Dolphin" sequence.

Fig. 9. The distortion of each reconstructed sequence is measured against the original irregular sequence by the Metro algorithm. The remeshing error of the uncompressed semi-regular sequence is shown as a guideline. For comparison, the performances of the static mesh coder, which remeshes and compresses each frame independently without motion compensation, are also shown in Figs. 18–20. We can see that, for all three test sequences, the proposed algorithm outperforms the static mesh coder significantly especially at low bit rates. For example, to achieve an error 0.001 on the "Dancer" model, the proposed algorithm needs about 1000 bytes/frame while the static mesh coder requires about 1800 bytes/frame. Also, for the same model, when the bit rate is 2000 bytes/frame, the proposed algorithm yields an error 0.00055, while the static coding yields

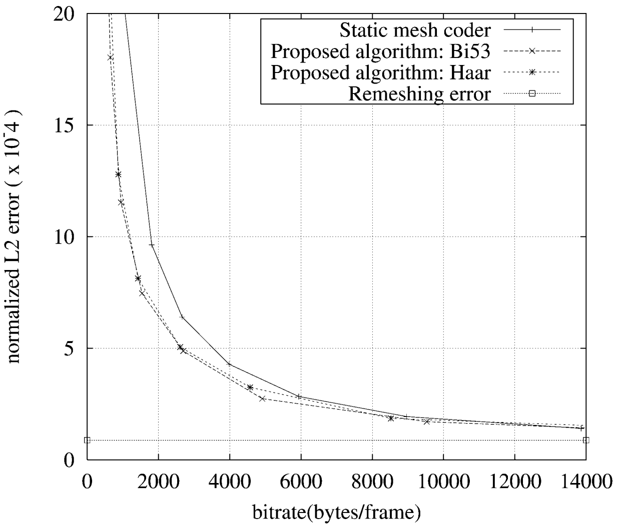


Fig. 20. Compression performances on the "Dinosaur" sequence.

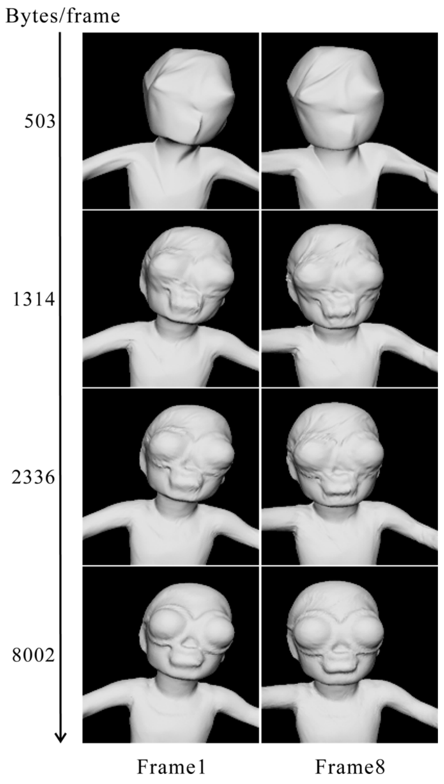


Fig. 21. Reconstructed "Dancer" first and eighth frames at different bit rates.

0.0008. The biorthogonal 5/3 filter yields slightly better compression performance than the Haar filter. Note that we obtain the semi-regular mesh sequences sequentially by minimizing only the forward prediction errors in (4). Thus, in the biorthogonal filter in (10), the backward prediction term provides only a marginal gain as compared with the forward prediction term. Therefore, the performance differences between the Haar filter and the biorthogonal filter are negligible. In the following tests, we use the biorthogonal filter.

Fig. 21 shows the reconstructed frames of the "Dancer" sequence at different frame rates. At very low bit rate, the structure of the base mesh can be inferred from the smoothly reconstructed image, since only a few wavelet coefficients are used and they contain mostly low frequency information. As more bits are received, the decoder can use a larger number of wavelet coefficients for the reconstruction. Also, the accuracy of each coefficient is improved. Thus, the reconstructed image has a higher quality, and detailed geometrical parts such as eye glasses are more faithfully decoded.

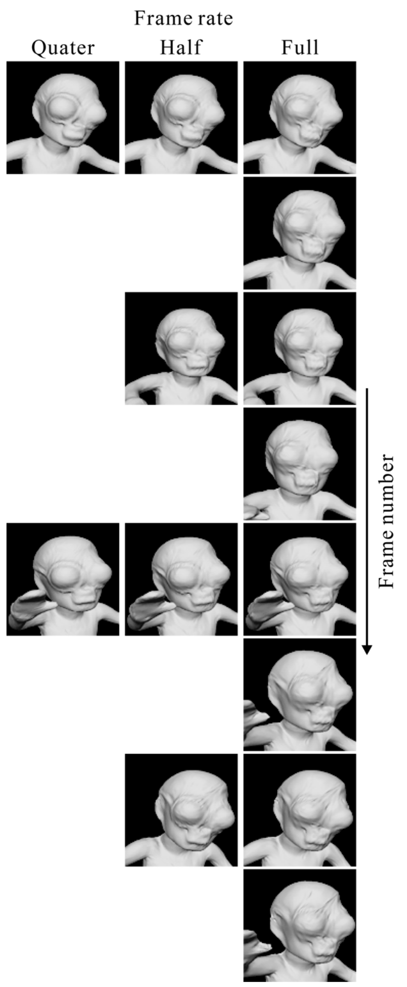
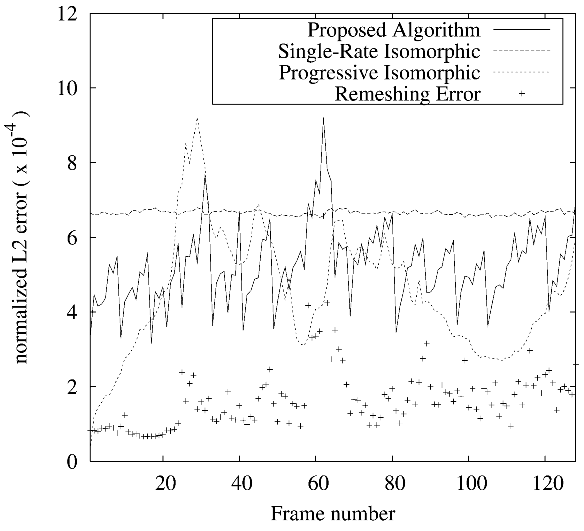


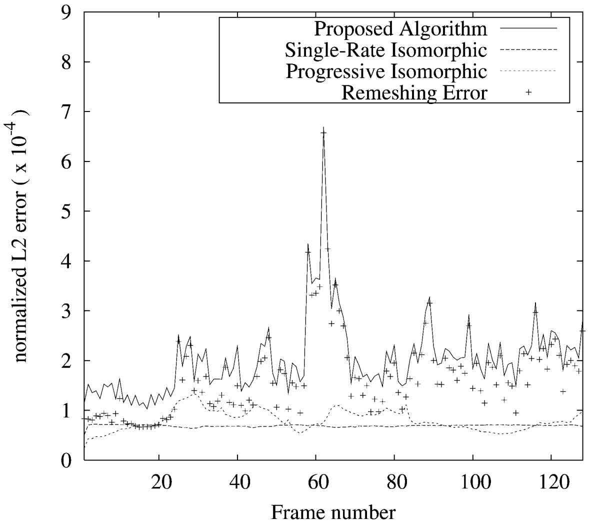
Fig. 22. Effect of temporal frame rate on the overall performance.

Fig. 22 shows the effect of temporal frame rate on the overall performance. The images at the left, middle and right columns are reconstructed at the quarter, half, and full frame rates, respectively. Each column is reconstructed using the same bit budget (about 10 500 bytes). Therefore, at a lower frame rate, a frame is allocated a larger amount of bits and reconstructed with a higher quality. The quality differences are noticeable at feature points, such as eyes and a mouth. However, a lower temporal frame rate can cause motion jerkiness. It is worthy to point out that the sequence is compressed only once, but the required information for a different frame rate sequence is extracted flexibly from the same compressed bitstream.

Finally, Fig. 23 evaluates the performance of the proposed algorithm on the "Jump" sequence in [13], which is a long isomorphic sequence of a jumping human. For comparison, Fig. 23 also shows the performances of two isomorphic sequence coders: Yang *et al.*'s algorithm in [7] is a single-rate isomorphic coder, while Guskov and Khodakovsky's algorithm



(a)



(b)

Fig. 23. Comparison of the compression performances of the proposed algorithm, the single-rate isomorphic coder in [7] and the progressive isomorphic coder in [13] on the "Jump" sequence at (a) 1 bpv and (b) 4 bpv.

[13] is a progressive isomorphic coder. The bit rate in bits per vertex (bpv) is computed by dividing the whole file size by the number of vertices in the isomorphic sequence. The two isomorphic coders exploit the exact motion trajectories in the sequence to achieve coding gain. On the contrary, the proposed algorithm does not assume any *a priori* knowledge on the sequence and remeshes it into a semi-regular sequence based on the motion estimation. The remeshing errors are also plotted in Fig. 23. At 1 bpv, in spite of the remeshing errors, the proposed algorithm provides comparable performances with the two isomorphic coders. However, at a high bit rate of 4 bpv, the coding errors become very small and the remeshing errors are dominant. Therefore, the two isomorphic coders provide better performances than the proposed algorithm. But, the proposed algorithm can encode non-isomorphic sequences as well as isomorphic sequences, whereas the conventional algorithms can handle only isomorphic sequences. Table II

TABLE II
AVERAGE DISTORTIONS OF THE PROPOSED ALGORITHM, THE SINGLE-RATE ISOMORPHIC CODER IN [7] AND THE PROGRESSIVE ISOMORPHIC CODER IN [13] ON THE "JUMP" SEQUENCE

	1 bpv	2 bpv	4 bpv
Proposed algorithm	0.0005262	0.0002854	0.0002040
Single-rate isomorphic	0.0006654	0.0001662	0.0000691
Progressive isomorphic	0.0004528	0.0001944	0.0000796
Remeshing error	0.0001681		

summarizes the average distortion performances on the "Jump" sequence at different bit rates.

VIII. CONCLUSION

In this work, we first proposed an algorithm to represent 3-D dynamic objects with semi-regular mesh sequences. Given an irregular mesh sequence, the proposed algorithm constructs a semi-regular mesh structure for the first frame mesh, and then maps it to the subsequent frames based on the hierarchical motion estimation. The semi-regular mesh sequence has a highly regular structure, which enables us to apply an effective progressive coding scheme. Specifically, we developed an embedded coding scheme based on the spatiotemporal decomposition. Simulation results demonstrated that the proposed coding algorithm outperforms the static mesh coding.

Future research will include the following issues to make our work more complete.

- The current implementation of the proposed algorithm can process only surfaces that are homeomorphic to a sphere. We will extend the proposed algorithm to deal with general surfaces, *e.g.*, surfaces with holes and boundaries.
- After the remeshing, the proposed algorithm yields an isomorphic sequence. However, this can be a constraint, if a dynamic object breaks into multiple parts or conversely multiple parts merge into one object in a motion sequence. The remeshing scheme will be generalized to remove this constraint, but it should maintain the regularity of the sequence and the motion trajectory information as much as possible.
- We used the butterfly subdivision for the wavelet transform of mesh sequences because of its straightforward implementation. However, the other schemes such as the Loop subdivision also can be used, although the forward-loop transform requires the solution of a sparse linear system [2]. We will incorporate different subdivision schemes into our mesh sequence coder and investigate their effects on the overall performance.
- The EBCOT algorithm was successfully adopted as the embedded coder in the lifting-based wavelet video coder [29], [32]. It was shown to be more efficient than the SPIHT algorithm in supporting various modes of scalability. We will apply the EBCOT scheme to the semi-regular mesh sequence coding to support spatial scalability mode in addition to SNR and temporal modes.

ACKNOWLEDGMENT

The authors would like to thank Dr. I. Guskov for providing the "Jump" sequence and their compression results.

References

[1] A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *Proc. SIGGRAPH*, Jul. 2000, pp. 271–278.

[2] A. Khodakovsky and I. Guskov, "Normal mesh compression," in *Geometric Modeling for Scientific Visualization*. Heidelberg, Germany: Springer-Verlag, 2002.

[3] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin, "Making faces," in *Proc. SIGGRAPH*, 1998, pp. 55–66.

[4] J. Neumann and Y. Aloimonos, "Spatio-temporal stereo using multi-resolution subdivision surfaces," *Int. J. Comput. Vis.*, vol. 47, pp. 181–193, 2002.

[5] J. E. Lengyel, "Compression of time-dependent geometry," in *ACM Symp. Interactive 3D Graph.*, 1999, pp. 89–96.

[6] J.-H. Ahn, C.-S. Kim, C.-C. J. Kuo, and Y.-S. Ho, "Motion compensated compression of 3D animation models," *IEE Electron. Lett.*, vol. 37, no. 24, pp. 1445–1446, Nov. 2001.

[7] J.-H. Yang, C.-S. Kim, and S.-U. Lee, "Compression of 3D triangle mesh sequences based on vertex-wise motion vector prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1178–1184, Dec. 2002.

[8] L. Ibarria and J. Rossignac, "Dynapack: Space-time compression of the 3D animations of triangle meshes with fixed connectivity," in *Proc. ACM Symp. Computer Animation*, 2003, pp. 126–135.

[9] M. Alexa and W. Müller, "Representing animations by principal components," in *Comput. Graph. Forum*, 2000, vol. 19, pp. 411–418.

[10] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Comput. Graph.*, vol. 28, pp. 25–34, 2004.

[11] H. M. Briceno, P. V. Sander, L. McMillan, S. Gotler, and H. Hoppe, "Geometry videos: A new representation for 3D animations," in *Proc. ACM Symp. Computer Animation*, 2003, pp. 136–146.

[12] S. Gupta, K. Sengupta, and A. Kassim, "Registration and partitioning-based compression of 3-D dynamic data," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 11, pp. 1144–1155, Nov. 2003.

[13] I. Guskov and A. Khodakovsky, "Wavelet compression of parametrically coherent mesh sequences," in *Proc. ACM/EG Symp. Computer Animation*, Aug. 2004.

[14] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

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

[16] 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, no. 1, pp. 42–52, Feb. 1994.

[17] Y. Wang and O. Lee, "Active mesh-a feature seeking and tracking image sequence representation scheme," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 610–624, Sep. 1994.

[18] Y. Altunbasak and A. M. Tekalp, "Occlusion-adaptive, content-based mesh design and forward tracking," *IEEE Trans. Image Process.*, vol. 6, no. 9, pp. 1270–1280, Sep. 1997.

[19] P. V. Beek, A. M. Tekalp, I. Celasun, N. Zhuang, and M. Xia, "Hierarchical 2-D mesh representation, tracking, and compression for object-based video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 3, pp. 353–369, Mar. 1999.

[20] I. Celasun and A. M. Tekalp, "Optimal 2-D hierarchical content-based mesh design and update for object-based video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 10, pp. 1135–1153, Oct. 2000.

[21] M. Lounsbery, Multiresolution Analysis for Surfaces of Arbitrary Topological Type, Dept. Comput. Sci., Univ. Washington, Seattle, 1994, Ph.D. dissertation.

[22] M. Eck, T. DeRose, and T. Duchamp, "Multiresolution analysis of arbitrary meshes," in *Proc. SIGGRAPH*, 1995, pp. 173–182.

[23] A. W. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," in *Proc. SIGGRAPH*, 1998, pp. 95–104.

[24] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder, "Normal meshes," in *Proc. SIGGRAPH*, 2000, pp. 95–102.

[25] N. Dyn, D. Levin, and J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Trans. Graphics*, vol. 9, pp. 160–169, 1990.

[26] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Trans. Image Process.*, vol. 3, no. 9, pp. 559–571, Sep. 1994.

[27] S. Choi and J. Wood, "Motion compensated 3-D subband coding of video," *IEEE Trans. Image Process.*, vol. 8, no. 2, pp. 155–167, Feb. 1999.

[28] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *SIAM J. Mathemat. Anal.*, vol. 29, no. 2, pp. 511–546, 1998.

[29] A. Secker and D. Taubman, "Lifting-based invertible motion adaptive transform (LIMAT) framework for highly scalable video compression," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1530–1542, Dec. 2003.

[30] M. Flierl and B. Girod, "Investigation of motion-compensated lifted wavelet transforms," in *Proc. Picture Coding Symposium*, Apr. 2003, pp. 59–62.

[31] B.-J. Kim, Z. Xiong, and W. A. Pearlman, "Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 12, pp. 1374–1387, Dec. 2000.

[32] D. Taubman and M. W. Marcellin*, JPEG2000: Image Compression, Fundamentals, Standards and Practice*. Boston, MA: Kluwer, 2002.

[33] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proc. SIGGRAPH*, 1997, pp. 209–216.

[34] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

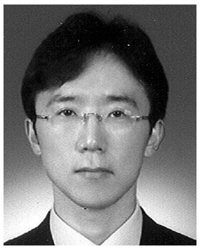
[35] J. Gross and J. Yellen*, Graph Theory and its Applications*. Boca Ration, FL: CRC, 1999.

[36] A. Said and W. A. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[37] J.-Y. Sim, C.-S. Kim, C.-C. J. Kuo, and S.-U. Lee, "Rate-distortion optimized compression and view-dependent transmission of 3D normal meshes," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 7, pp. 854–868, Jul. 2005.

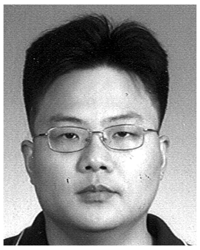
[38] G. Davis and A. Nosratinia, "Wavelet-based image coding: An overview," *Appl. Comput. Control, Signals, Circuits*, vol. 1, no. 1, 1998.

[39] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," in *Comput. Graph. Forum*, 1998, vol. 17, pp. 167–174.



Jeong-Hyu Yang received the B.S. degree in electrical engineering in 1998, and the M.S. and Ph.D. degrees in electrical engineering and computer science in 2000 and 2004, respectively, all from Seoul National University (SNU), Seoul, Korea.

In August 2004, he joined LG Electronics, Seoul, as a Senior Research Engineer. His research topics include video and 3-D graphics processing, 3DTV, and system-on-chip design.



Chang-Su Kim (S'95–M'01–SM'05) received the B.S. and M.S. degrees in control and instrumentation engineering from Seoul National University (SNU), Seoul, Korea, in 1994 and 1996, respectively, and the Ph.D. degree in electrical engineering from SNU with a Distinguished Dissertation Award.

From 2000 to 2001, he was a Visiting Scholar with the Signal and Image Processing Institute, University of Southern California, Los Angeles, and a Consultant for InterVideo Inc., Los Angeles. From 2001 to 2003, he coordinated the 3D Data Compression Group, National Research Laboratory for 3D Visual Information Processing, SNU. From 2003 and 2005, he was an Assistant Professor in the Department of Information Engineering, Chinese University of Hong Kong. In September 2005, he joined the Department of Electronics Engineering, Korea University, as an Assistant Professor. His research topics include video and 3-D graphics processing and multimedia communications. He has published more than 80 technical papers in international conferences and journals.

Sang-Uk Lee (S'75–M'80–SM'99) received the B.S. degree from Seoul National University, Seoul, Korea, in 1973, the M.S. degree from Iowa State University, Ames, in 1976, and the Ph.D. degree from the University of Southern California, Los Angeles, in 1980, all in electrical engineering.

From 1980 to 1981, he was with the General Electric Company, Lynchburg, VA, working on the development of digital mobile radio. From 1981 to 1983, he was a Member of Technical Staff, M/A-COM Research Center, Rockville, MD. In 1983, he joined the Department of Control and Instrumentation Engineering, Seoul National University (SNU), Seoul, Korea, as an Assistant Professor, where he is now the Head of the School of Electrical Engineering and Computer Science. He is also affiliated with the Automation and Systems Research Institute and the Institute of New Media and Communications at SNU. He is currently the President of the Korean Institute of Communication Science. His current research interests are in the areas of image and video signal processing, digital communication, and computer vision.

Dr. Lee served as an Editor-in-Chief for the *Transaction of the Korean Institute of Communication Science* from 1994 to 1996. He is currently an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and is on the Editorial Board of the *Journal of Visual Communication and Image Representation* and the EURASIP *Journal of Applied Signal Processing.* He is a member of Phi Kappa Phi.