Graph-based compression of dynamic 3D point cloud sequences

Dorina Thanou, Philip A. Chou, and Pascal Frossard

***Abstract*—This paper addresses the problem of compression of 3D point cloud sequences that are characterized by moving 3D positions and color attributes. As temporally successive point cloud frames share some similarities, motion estimation is key to effective compression of these sequences. It however remains a challenging problem as the point cloud frames have varying numbers of points without explicit correspondence information. We represent the time-varying geometry of these sequences with a set of graphs, and consider 3D positions and color attributes of the points clouds as signals on the vertices of the graphs. We then cast motion estimation as a feature matching problem between successive graphs. The motion is estimated on a sparse set of representative vertices using new spectral graph wavelet descriptors. A dense motion field is eventually interpolated by solving a graph-based regularization problem. The estimated motion is finally used for removing the temporal redundancy in the predictive coding of the 3D positions and the color characteristics of the point cloud sequences. Experimental results demonstrate that our method is able to accurately estimate the motion between consecutive frames. Moreover, motion estimation is shown to bring significant improvement in terms of the overall compression performance of the sequence. To the best of our knowledge, this is the first paper that exploits both the spatial correlation inside each frame (through the graph) and the temporal correlation between the frames (through the motion estimation) to compress the color and the geometry of 3D point cloud sequences in an efficient way.**

***Index Terms*—3D sequences, voxels, graph-based features, spectral graph wavelets, motion compensation**

I. Introduction

Dynamic 3D scenes such as humans in motion can now be captured by arrays of color plus depth (or 'RGBD') video cameras [1], and such data is getting very popular in emerging applications such as animation, gaming, virtual reality, and immersive communications. The geometry captured by RGBD camera arrays, unlike computer-generated geometry, has little explicit spatio-temporal structure, and is often represented by sequences of colored point clouds. Frames, which are the point clouds captured at a given time instant as shown in Fig. 1, may have different numbers of points, and there is no explicit association between points over time. Performing motion estimation, motion compensation, and effective compression of such data is therefore a challenging task.

D. Thanou and P. Frossard are with Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Laboratory-LTS4, Lausanne, Switzerland. P. A. Chou is with Microsoft Research, Redmond, WA, USA. (e-mail:{dorina.thanou, pascal.frossard}@epfl.ch, pachou@microsoft.com).

Part of this work has been presented in the International Conference on Image Processing (ICIP), Quebec, Canada, September 2015.

Fig. 1. Sequence of point cloud frames captured at different time instances (frames 1, 3, 36, 42) in the 'man' sequence.

In this paper, we focus on the compression of the 3D geometry and color attributes and propose a novel motion estimation and compensation scheme that exploits temporal correlation in sequences of point clouds. To deal with the large size of these sequences, we consider that the point clouds are voxelized, that is, their 3D positions are quantized to a regular, axis-aligned, 3D grid having a given stepsize. This quantization of the space is commonly achieved by modeling the 3D point cloud sequences as a series of octree data structures [1], [2], [3]. In contrast to polygonal mesh representations, the octree structure exploits the spatial organization of the 3D points, which results in easy manipulations and permits real-time processing of the point cloud data. In more details, an octree is a tree structure with a predefined depth, where every branch node represents a certain cube volume in the 3D space, which is called a voxel. A voxel containing at least a point is said to be occupied. Although the overall voxel set lies in a regular grid, the set of occupied voxels are non-uniformly distributed in space. To uncover the irregular structure of the occupied voxels inside each frame, we consider voxels as vertices in a graph $\mathcal{G}$, with edges between nearby vertices. Attributes of each voxel n, including 3D position $p(n) = [x, y, z](n)$ and color components $c(n) = [r, g, b](n)$, are treated as signals residing on the vertices of the graph. Such an example is illustrated in Fig. 2. As frames in the 3D point cloud sequences are correlated, the graph signals at consecutive time instants are also correlated. Hence, removing temporal correlation implies comparing the signals residing on the vertices of consecutive graphs. The estimation of the correlation is however a challenging task as the graphs usually have different numbers of nodes and no explicit correspondence information between the nodes is available in the sequence.

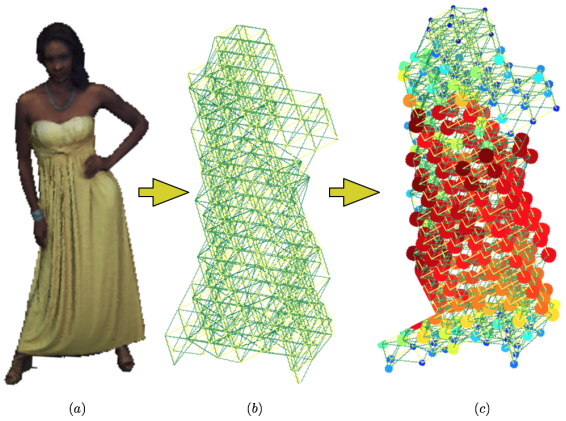


Fig. 2. Example of a point cloud of the 'yellow dress' sequence (a). The geometry is captured by a graph (b) and the r component of the color is considered as a signal on the graph (c). The size and the color of each disc indicate the value of the signal at the corresponding vertex.

We build on our previous work [4], and propose a novel algorithm for motion estimation and compensation in 3D point cloud sequences. We cast motion estimation as a feature matching problem on dynamic graphs. In particular, we compute new local features at different scales with spectral graph wavelets (SGW) [5] for each node of the graph. Our feature descriptors, which consist of the wavelet coefficients of each of the signals placed in the corresponding vertex, are then used to compute point-to-point correspondences between graphs of different frames. We match our SGW features in different graphs with a criterion that is based on the Mahalanobis distance and trained from the data. To avoid inaccurate matches, we first compute the motion on a sparse set of matching nodes that satisfy the matching criterion. We then interpolate the motion of the other nodes of the graph by solving a new graph-based quadratic regularization problem, which promotes smoothness of the motion vectors on the graph in order to build a consistent motion field.

Then, we design a compression system for 3D point cloud sequences, where we exploit the estimated motion information in the predictive coding of the geometry and color information. The basic blocks of our compression architecture are shown in Fig. 3. We code the motion field in the graph Fourier domain by exploiting its smoothness on the graph. Temporal redundancy in consecutive 3D positions is removed by coding the structural difference between the target frame and the motion compensated reference frame. The structural difference is efficiently described in a binary stream format as described in [6]. Finally, we predict the color of the target frame by interpolating it from the color of the motion compensated reference frame. Only the difference between the actual color information and the result of the motion compensation is actually coded with a state-of-the-art encoder for static octree data [7]. Experimental results illustrate that our motion estimation scheme effectively captures the correlation between consecutive frames. Moreover, introducing motion compensation in compression of 3D point cloud sequences results in significant improvement in terms of rate-distortion performance of the overall system, and in particular in the compression of the color attributes where we achieve a gain of up to 10 dB in comparison to state-of-the-art encoders.

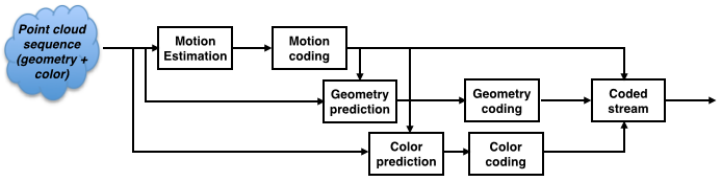


Fig. 3. Schematic overview of the encoding architecture of a point cloud sequence. Motion estimation is used to reduce the temporal redundancy for efficient compression of the 3D geometry and the color attributes.

The contribution of the paper is summarized as follows. The proposed encoder is the first one to exploit motion estimation in efficient coding of point cloud sequences, without going first through the expensive conversion of the data into a temporally consistent polygonal mesh. Second, we represent the point cloud sequences as a set of graphs and we solve the motion estimation problem as a new feature matching problem in dynamic graphs. Third, we propose a differential coding scheme for geometry and color compression that provides significant overall gain in terms of rate-distortion performance.

The rest of the paper is organized as follows. First, in Section II, we review the existing work in the literature that studies the problem of compression of 3D point clouds. Next, in Section III, we describe the representation of 3D point clouds by performing an octree decomposition of the 3D space and we introduce graphs to capture the irregular structure of this representation. The motion estimation scheme is presented in Section IV. The estimated motion is then applied to the predictive coding of the geometry and the color in Section V. Finally, experimental results are given in Section VI.

II. Related work

The direct compression of 3D point cloud sequences has been largely overlooked so far in the literature. A few works have been proposed to compress static 3D point clouds. Some examples include the 2D wavelet transform based scheme of [8], and the subdivision of the point cloud space in different resolution layers using a kd-tree structure [9]. An efficient binary description of the spatial point cloud distribution is performed through a decomposition of the 3D space using octree data structures. The octree decomposition, in contrast to the mesh construction, is quite simple to obtain. It is the basic idea behind the geometry compression algorithms of [3], [2]. The octree structure is also adopted in [7], to compress point cloud attributes. The authors construct a graph for each branch of leaves at certain levels of the octree. The graph transform, which is equivalent to the Karhunen-Loève transform, is then applied to decorrelate the color attributes that are treated as signals on the graph. The proposed algorithm has been shown to remove the spatial redundancy for compression of the 3D point cloud attributes, with significant improvement over traditional methods. Sparse representations in a trained

dictionary have further been used in [10] to compress the geometry of 3D point clouds surfaces. An alternative representation of 3D data is obtained by assigning a sphere to each point through the QSplat algorithm. The compression of the position, spherical radius, and normal data of each sphere has been studied in [11], [12]. Recently, the authors in [13], proposed a novel geometry compression algorithm for large-scale 3D point clouds obtained by terrestrial laser scanners. In their work, the point clouds are converted into a range image and the radial distance in the range image is encoded in an adaptive way. However, all the above methods are designed mainly for static point clouds. In order to apply them to point cloud sequences, we need to consider each frame of the sequence independently, which is clearly suboptimal.

Temporal and spatial redundancy of point cloud sequences has been recently exploited in [6]. The authors compress the geometry by comparing the octree data structure of consecutive point clouds and encoding their structural difference. The proposed compression framework can handle general point cloud streams of arbitrary and varying size, with unknown correspondences. It enables detection and differential encoding of spatial changes within temporarily adjacent octree structures by modifying the octree data structure without computing the exact motion of the voxels. Motion estimation in point cloud sequences can be quite challenging due to the fact that point-to-point correspondences between consecutive frames are not known. While there exists a huge amount of works in the literature that study the problem of motion estimation in video compression, these methods cannot be extended easily to graph settings. In classical video coding schemes, motion in 3-D space is mainly considered as a set of displacements in the regular image plane. Pixel-based methods [14], such as block matching algorithms, or optical and scene flow algorithms, are designed for regular grids. Their generalization to the irregular graph domain is however not straightforward. Feature-based methods [15], such as interest point detection, have also been widely used for motion estimation in video compression. These features usually correspond to key points of images such as corners or sharp edges [16]–[18]. With an appropriate definition of features on graphs, these methods can be extended to graphs. To the best of our knowledge though, they have not been adapted so far to estimate the motion on graphs, nor on point clouds. Someone could also apply classical 3D descriptors such as [19]–[24] to define 3D features. However, these types of descriptors assume that the point cloud represents a surface, which is not well adapted to the case of graphs. An overview of classical 3D descriptors can be found in [25].

For the sake of completeness, we should mention that 3D point clouds are often converted into polygonal meshes, which can be compressed with a large body of existing methods. In particular, there exists literature for compressing dynamic 3D meshes with either fixed connectivity and known correspondences (e.g., [26]–[30]) or varying connectivity (e.g., [31]–[33]). A different type of approach consists of the video based methods. The irregular 3D structure of the meshes is parametrized into a rectangular 2D domain, obtaining the so called geometry images [34] in the case of a single mesh and geometry videos [35], [36] in the case of 3D mesh sequences. The mapping of the 3D mesh surface onto a 2D array, which can be done either by using only the 3D geometry information or both the geometry and the texture information [37], allows conventional video compression to be applied to the projected 2D videos. Within the same line of work, emphasis has been given to extending these types of algorithms to handling sequences of meshes with different numbers of vertices and exploiting temporal correlation between them. An example is the recent work in [38], which proposes a framework for compressing 3D human motion oriented geometry videos by constructing key frames that are able to reconstruct the whole motion sequence. Comparing to the mesh-based compression algorithms, the advantage is that the mesh connectivity information (i.e., vertices and faces) does not need to be sent to the decoder, and the complexity is reduced by performing the operations from the 3D to the 2D space. All the above mentioned works however require the conversion process of the point cloud into a mesh in the encoder and the inverse at rendering, which might be computationally expensive. Finally, marching cubes algorithm [39] can be used to extract a polygonal mesh in a fast way, but it requires a "filled" volume.

Thus, we believe that octree representations are efficient for modeling temporally changing unorganized point clouds, where input 3D points correspond to sampling of surfaces. In what follows, we use such representations to design a framework for compressing effectively 3D point cloud sequences.

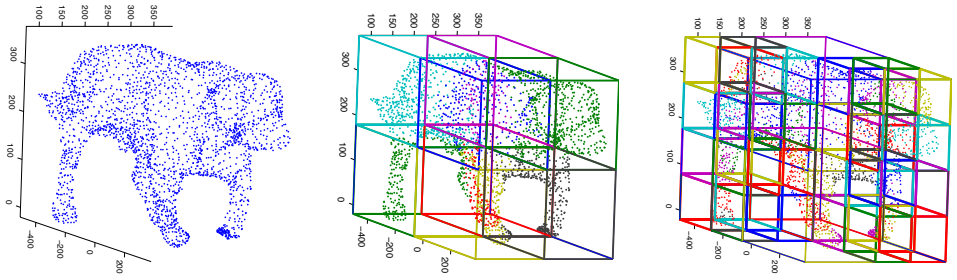
III. Structural representation of 3D point clouds

3D point clouds usually have little explicit spatial structure. Someone can however organize the 3D space by converting the point cloud into an octree data structure [1], [2], [3]. In what follows, we recall the octree construction process, and introduce graphs as a tool for capturing the structure of the leaf nodes of the octree.

A. Octree representation of 3D point clouds

An octree is a tree structure with a predefined depth, where every branch node represents a certain cube volume in the 3D space, which is called a voxel. A voxel containing at least one sample from the 3D point cloud is said to be occupied. Initially, the 3D space is hierarchically partitioned into voxels whose total number depends on the number of 3D volume subdivisions, i.e., the depth of the resulting tree structure. For a given depth, an octree is constructed by traversing the tree structure in depth-first order. Starting from the root, each node can generate eight children voxels. At the maximum depth of the tree, all the points are mapped to leaf voxels. An example of the voxelization of a 3D model for different depth levels, or equivalently for different quantization stepsizes, is shown in Fig. 4.

In contrast to temporally consistent polygonal mesh representations, the octree structures are appropriate for modeling 3D point cloud sequences as they are easy to obtain. Thanks to the different depths of the tree, they permit a multiresolution representation of the data that leads to efficient data



(a) Original point cloud (b) Depth 1 (c) Depth 2

Fig. 4. Octree decomposition of a 3D model for two different depth levels. The points belonging to each voxel are represented by the same color.

processing in many applications. In particular, this multiresolution representation permits a progressive compression of the 3D positions of the data, which is lossless within each representation level [6].

B. Graph-based representation of 3D point clouds

Although the overall voxel set lies on a regular grid, the set of occupied voxels is non-uniformly distributed in space, as most of the leaf voxels are unoccupied. In order to represent the irregular structure formed by the occupied voxels, we use a graph-based representation. Graph-based representations are flexible and well adapted to data that live on an irregular domain [40]. In particular, we represent the set of occupied voxels of the octree using a weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ and $\mathcal{E}$ represent the vertex and edge sets of $\mathcal{G}$. Each of the N nodes in $\mathcal{V}$ corresponds to an occupied voxel, while each edge in $\mathcal{E}$ connects neighboring occupied voxels. We define the connectivity of the graph based on the K-nearest neighbors (K-NN) graph, which is widely used in the literature. We usually set K to 26 as it corresponds to the maximum number of neighbors for a node that has a maximum distance of one step along any axis of the 3D space. However, since in general not all 26 voxels are occupied, we extend our construction to the general K-NN graph. Two vertices are thus connected if they are among the 26 nearest neighbors in the voxel grid, which results in a connected graph. This property is useful in the interpolation of the motion vectors, as we see in the following section. The matrix W is a matrix of positive edge weights, with $W(i,j)$ denoting the weight of an edge connecting vertices i and j. This weight captures the connectivity pattern of nearby occupied voxels and is chosen to be inversely proportional to the 3D distance between voxels.

After the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ is constructed, we consider the attributes of the 3D point cloud — the 3D coordinates $p = [x, y, z]^T \in \mathbb{R}^{3\times N}$ and the color components $c = [r, g, b]^T \in \mathbb{R}^{3\times N}$ — as signals that reside on the vertices of the graph $\mathcal{G}$. A spectral representation of these signals can be obtained with the help of the Graph Fourier Transform (GFT). The GFT is defined through the eigenvectors of the graph Laplacian operator $\mathcal{L} = D - W$, where D is the diagonal degree matrix whose i^{th} diagonal element is equal to the sum of the weights of all the edges incident to vertex i [41]. The graph Laplacian is a real symmetric matrix that has a complete set of orthonormal eigenvectors with corresponding nonnegative eigenvalues. We here denote its eigenvectors by $\chi = [\chi_0, \chi_1, ..., \chi_{N-1}]$, and the spectrum of eigenvalues by $\Lambda := \left\{0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{(N-1)}\right\}$, where N is the number of vertices of the graph. The GFT of any graph signal $f \in \mathbb{R}^N$ is then defined as

$$F_f(\lambda_\ell) := < f, \chi_\ell > = \sum_{n=1}^{N} f(n)\chi_\ell^*(n),$$

where the inner product is conjugate-linear in the first argument. Then, the inverse graph Fourier transform (IGFT) is given by

$$f(n) = \sum_{\ell=0}^{N-1} F_f(\lambda_\ell)\chi_\ell(n).$$

The GFT provides a useful spectral representation of the data. Furthermore, it has been shown to be optimum for decorrelating a signal following the Gaussian Markov Random Field model with precision matrix $\mathcal{L}$ [42]. The GFT will be used later to define spectral features and to code effectively data on the graph.

IV. Motion estimation in 3D point cloud sequences

As the frames have irregular structures, we use a feature-based matching approach to find correspondences in temporally successive point clouds. We use the graph information and the signals residing on its vertices to define feature descriptors on each vertex. We first define simple octant indicator functions to capture the signal values in different orientations. We then characterize the local topological context of each of the point cloud signals in each of these orientations, by using spectral graph wavelets (SGW) computed on the color and geometry signals at different resolutions [5]. Our feature descriptors, which consist of the wavelet coefficients of these signals are then used to compute point-to-point correspondences between graphs of different frames. We select a subset of best matching nodes to define a sparse set of motion vectors that describe the temporal correlation in the sequence. A dense motion field is eventually interpolated from the sparse set of motion vectors to obtain a complete mapping between two frames. The overall procedure is detailed below.

A. Multi-resolution features on graphs

We define features in each node by computing the variation of the signal values, i.e., geometry and color components, in

different parts of its neighborhood. For each node i belonging to the vertex set $\mathcal{V}$ of a graph $\mathcal{G}$, i.e., $i \in \mathcal{V}$, we first define the octant indicator function $o_{k,i} \in \mathbb{R}^N, \forall\ k = [1, 2, ..., 8]$, for the eight octants around the node i. For example, for the first octant it is given as follows

$$o_{1,i}(j) = \mathbf{1}_{\{x(j)\geq x(i), y(j)\geq y(i), z(j)\geq z(i)\}}(j),$$

where $\mathbf{1}_{\{\cdot\}}(j)$ is the indicator function on $j \in \mathcal{V}$, evaluated in a set $\{\cdot\}$ of voxels given by specific 3D coordinates. The first octant indicator function is thus nonzero only in the entries corresponding to the voxels whose 3D position coordinates are bigger than the ones of node i. We consider all possible combinations of coordinates, which results in a total of 2^3 indicator functions for the eight octants around i. These functions provide a notion of orientation of each node in the 3D space with respect to i, which is given by the octree decomposition.

We then compute graph spectral features based on both geometry and color information, by treating their values independently in each orientation. In particular, for each node $i \in \mathcal{V}$ and each geometry and color component $f \in \mathbb{R}^N$, where $f \in \{x, y, z, r, g, b\}$, we compute the spectral graph wavelet coefficients by considering independently the values of f in each orientation k with respect to node i such that

$$\phi_{i,s,o_{k,i},f} =< f \cdot o_{k,i}, \psi_{s,i} >, \tag{1}$$

where $k \in \{1, 2, ..., 8\}$, $s \in \mathcal{S} = \{s_1, ..., s_{max}\}$, is a set of discrete scales, and $\cdot$ denotes the pointwise product. The function $\psi_{s,i}$ represents the spectral graph wavelet of scale s placed at that particular node i. We recall that the spectral graph wavelets [5] are operator-valued functions of the graph Laplacian defined as

$$\psi_{s,i} = T_g^s \delta_i = \sum_{\ell=0}^{N-1} g(s\lambda_\ell)\chi_\ell^*(i)\chi_\ell.$$

The graph wavelets are determined by the choice of a generating kernel g, which acts as a band-pass filter in the spectral domain, and a scaling kernel h that acts as a lowpass filter and captures the low frequency content. The scaling is defined in the spectral domain, i.e., the wavelet operator at scale s is given by $T_g^s = g(s\mathcal{L})$. Spectral graph wavelets are finally realized through localizing these operators via the impulse δ on a single vertex i. The application of these wavelets to signals living on the graph results in a multi-scale descriptor for each node. We finally define the feature vector $\boldsymbol{\phi}_i$ at node i as the concatenation of the coefficients computed in (1) with wavelets at different scales, including the features obtained from the wavelet scaling function, i.e., $\boldsymbol{\phi}_i = [\phi_{i,s,o_{k,i},f},\ \phi_{i,h,o_{k,i},f}] \in \mathbb{R}^{8\times 6\times(|\mathcal{S}|+1)}$, where

$$\phi_{i,h,o_{k,i},f} =< f \cdot o_{k,i}, h(\mathcal{L})\delta_i > .$$

Finally, we note that spectral features have recently started to gain attention in the computer vision and shape analysis community. The heat kernel signatures [43], their scale-invariant version [44], the wave kernel signatures [45], the optimized spectral descriptors of [46], have already been used in 3D shape processing with applications in graph matching [47] or in mesh segmentation and surface alignment problems [48]. These features have been shown to be stable under small perturbations of the edge nodes of the graph. In all these works though, the descriptors are defined based only on the graph structure, and the information about attributes of the nodes such as color and 3D positions, if any, is assumed to be introduced in the weights of the graph. The performance of these descriptors depends on the quality of the defined graph. In contrast to this line of works, we define features by considering attributes as signals that reside on the vertices of a graph and characterize each vertex by computing the local evolution of these signals at different scales. Furthermore, this approach gives us the flexility to consider the signal values in different orientations as discussed above, and makes the descriptor of each node more informative.

B. Finding correspondences on dynamic graphs

We translate the problem of finding correspondences in two consecutive point clouds or frames of the sequence into finding correspondences between the vertices of their representative graphs. For the rest of this paper, we denote the sequence of frames as $\mathcal{I} = \{\mathcal{I}_1,\ \mathcal{I}_2,\ ...,\ \mathcal{I}_{max}\}$ and the set of graphs corresponding to each frame as $\mathcal{G} = \{\mathcal{G}_1,\ \mathcal{G}_2,\ ...,\ \mathcal{G}_{max}\}$. For two consecutive frames of the sequence, $\mathcal{I}_t,\ \mathcal{I}_{t+1}$, called also reference and target frame respectively, our goal is to find correspondences between the vertices of their representative graphs $\mathcal{G}_t$ and $\mathcal{G}_{t+1}$. The number of vertices in the respective vertex sets $\mathcal{V}_t,\ \mathcal{V}_{t+1}$ can differ between the graphs and is denoted as N_t and N_{t+1} respectively.

We use the features defined in the previous subsection to measure the similarity between vertices. We compute the matching score between two nodes $m \in \mathcal{V}_t, n \in \mathcal{V}_{t+1}$ as the Mahalanobis distance between the corresponding feature vectors, i.e.,

$$\sigma(m, n) = (\boldsymbol{\phi}_m - \boldsymbol{\phi}_n)^T P(\boldsymbol{\phi}_m - \boldsymbol{\phi}_n), \quad \forall m \in \mathcal{V}_t,\ n \in \mathcal{V}_{t+1}, \tag{2}$$

where P is a matrix that characterizes the relationships between the geometry and the color feature components (measured in different units), as well as the contribution of each of the wavelet scales in the matching performance. As a result, if $m \in \mathcal{V}_t$ corresponds to $n \in \mathcal{V}_{t+1}$, $\boldsymbol{\phi}_m$ is a Gaussian random vector with mean $\boldsymbol{\phi}_n$ and covariance P^{-1}, while if m does not correspond to n, $\boldsymbol{\phi}_m$ comes from a very flat (essentially uniform) distribution. Hence the matching score $\sigma(m, n)$ can be considered a log likelihood ratio for testing the hypothesis that m corresponds to n. We learn the positive definite matrix P by estimating the sample inverse covariance matrix from a set of training features that are known to be in correspondence. More precisely, we consider two frames $\mathcal{I}_\alpha,\ \mathcal{I}_\beta$, for which the correspondences are known. For each $m \in \mathcal{V}_\alpha$ corresponding to $n \in \mathcal{V}_\beta$, we compute the error obtained from their feature difference, i.e., $\epsilon_{m,n} = \boldsymbol{\phi}_m - \boldsymbol{\phi}_n$. The error vectors defined by all the corresponding nodes are then used to estimate the sample covariance matrix of the feature differences and subsequently the inverse sample covariance matrix P.

For each node in $\mathcal{G}_{t+1}$, we then use the matching score of Eq. (2) to define the best matching node in $\mathcal{G}_t$. In particular,

for each $n \in \mathcal{V}_{t+1}$, we define as its best match in $\mathcal{V}_t$, the node m_n with the minimum Mahalanobis distance, i.e.,

$$m_n = \underset{m \in \mathcal{V}_t}{\operatorname{argmin}}\, \sigma(m, n). \tag{3}$$

From the global set of correspondences computed for all the nodes of $\mathcal{V}_{t+1}$, we select a sparse set of significant matches, namely correspondences with best scores. The objective of this selection is to take into consideration only accurate matches and ignore others since inaccurate correspondences are possible in the case of large displacements. We also want to avoid matching points in $\mathcal{I}_{t+1}$ that do not have any true correspondence in the preceding frame $\mathcal{I}_t$. In order to ensure that we keep correspondences in all areas of the 3D space, we cluster the vertices of $\mathcal{G}_{t+1}$ into different regions and we keep only one correspondence, i.e., one representative vertex, per region. Clustering is performed by applying K-means in the 3D coordinates of the nodes of the target frame, where K is usually set to be equal to the target number of significant matches. In order to avoid inaccurate matches, a representative vertex per cluster is included in the sparse set only if its best matching distance given by Eq. (3) is smaller than a predefined threshold. This procedure results in detecting a sparse set of vertices n in $\mathcal{V}_{t+1}$, denoted $\mathcal{V}_{t+1}^S \subset \mathcal{V}_{t+1}$, and the set of their correspondences m_n in $\mathcal{V}_t$, $\mathcal{V}_t^S \subset \mathcal{V}_t$. Moreover, our sparse set of matching points tend to represent accurate correspondences that are well distributed spatially.

C. Computation of the motion vectors

We now describe how we generate a dense motion field from the sparse set of matching nodes $(m_n, n) \in \mathcal{V}_t^S \times \mathcal{V}_{t+1}^S$. Our implicit assumption is that vertices that are close in terms of 3D positions, namely close neighbors in the underlying graph, undergo a similar motion. We thus use the structure of the graph in order to interpolate the motion field, which is assumed to be smooth on the graph.

In more detail, our goal is to estimate the dense motion field $v_t = [v_t(m)]$, for all $m \in \mathcal{G}_t$, using the correspondences $(m_n, n) \in \mathcal{V}_t^S \times \mathcal{V}_{t+1}^S$. To determine $v_t(m)$ for $m = m_n \in \mathcal{V}_t^S$, we use the vector between the pair of matching points (m_n, n),

$$v_t(m_n|n) \triangleq p_{t+1}(n) - p_t(m_n). \tag{4}$$

Here we recall that p_t and p_{t+1} are the 3D positions of the vertices of $\mathcal{G}_t$ and $\mathcal{G}_{t+1}$, respectively. To determine $v_t(m)$ for $m \notin \mathcal{V}_t^S$, we consider the motion field v_t to be a vector-valued signal that lives on the vertices of $\mathcal{G}_t$. Then we smoothly interpolate the sparse set of motion vectors (4). The interpolation is performed by treating each component independently. Given the motion values on some of the vertices, we cast the motion interpolation as a regularization problem that estimates the motion values on the rest of the vertices by requiring the motion signal to vary smoothly across vertices that are connected by an edge in the graph. Moreover, we allow some smoothing on the known entries. The reason for that is that the proposed matching scheme does not necessarily guarantee that the sparse set of correspondences, and the estimated motion vectors associated with them, are correct. To limit the effect of motion estimation inaccuracies, for each matching pair $(m_n, n) \in \mathcal{V}_t^S \times \mathcal{V}_{t+1}^S$, we model the matching score in the local neighborhood of $m_n \in \mathcal{V}_t^S$ with a smooth signal approximation. Specifically, for each $n \in \mathcal{V}_{t+1}^S$, we extend the definition (4) to all $m \in \mathcal{V}_t$, i.e.,

$$v_t(m|n) = p_{t+1}(n) - p_t(m).$$

Then, for each node that belongs to the two-hop neighborhood of m_n i.e., $m \in \mathcal{N}_{m_n}^2$, we express $\sigma(m, n)$ as a function of the geometric distance of $p_t(m)$ from $p_t(m_n)$, using a second-order Taylor series expansion around $p_t(m)$. That is,

$$\begin{aligned}
&\sigma(m, n) \approx \sigma(m_n, n) \\
&+ (p_t(m) - p_t(m_n))^T M_n^{-1} (p_t(m) - p_t(m_n)) \\
&= \sigma(m_n, n) \\
&+ (v_t(m|n) - v_t(m_n|n))^T M_n^{-1} (v_t(m|n) - v_t(m_n|n)).
\end{aligned} \tag{5}$$

For each $n \in \mathcal{V}_{t+1}^S$, we take $\sigma(m, n)$ to be a discrete sampled version of a continuous function $\sigma(v, n)$ where the second order Taylor approximation is

$$\sigma(v, n) \approx \sigma(m_n, n) + (v - v_t(m_n|n))^T M_n^{-1} (v - v_t(m_n|n))).$$

Thus for each $n \in \mathcal{V}_{t+1}^S$, we assume that the matching score with respect to nodes that are in the neighborhood of its best match $m_n \in \mathcal{V}_t^S$ can be well modeled by a quadratic approximation function. We estimate M_n of this quadratic approximation as the normalized covariance matrix of the 3D offsets,

$$M_n = \frac{1}{|\mathcal{N}_{m_n}^2|} \sum_{m \in \mathcal{N}_{m_n}^2} \frac{(p_t(m) - p_t(m_n))(p_t(m) - p_t(m_n))^T}{\sigma(m, n) - \sigma(m_n, n)}.$$

This is motivated by the fact that if $\sigma(m, n) - \sigma(m_n, n) = (v_t(m) - v_t(m_n|n))^T M_n^{-1} (v_t(m) - v_t(m_n|n))$, then

$$u = \frac{v_t(m) - v_t(m_n|n)}{\sqrt{\sigma(m, n) - \sigma(m_n, n)}}$$

satisfies $1 = u^T M_n^{-1} u$. Hence, u lies in an ellipsoid whose second moment is proportional to M_n. Although there are other ways for computing M_n in (5), this moment-matching method is fast and guarantees that M_n is positive semi-definite. Next, we use the covariance matrices of the 3D offsets to define a diagonal matrix $Q \in \mathbb{R}^{3N_t \times 3N_t}$, such that

$$Q = \begin{bmatrix} M_1^{-1} & \cdots & \mathbf{0}_{3\times3} \\ \vdots & \ddots & \vdots \\ \mathbf{0}_{3\times3} & \cdots & M_{N_t}^{-1} \end{bmatrix},$$

where $M_m^{-1} = M_n^{-1}$ if $m = m_n$ for some $n \in \mathcal{V}_{t+1}^S$ and $M_m^{-1} = \mathbf{0}_{3\times3}$ otherwise. The matrix Q captures the second order Taylor approximation of the total matching score as a function of the motion vectors and the 3D geometry coordinates in the neighborhoods of the nodes in $\mathcal{V}_t^S$ and is used to regularize the motion vectors of the known entries in v_t as shown next.

Finally, we interpolate the dense set of motion vectors ${v_t}^*$ by taking into account the covariance of the motion vectors in the neighborhoods around the points that belong to the sparse

set $\mathcal{V}_t^S$ and imposing smoothness of the dense motion vectors on the graph

$$v_t{}^* = \underset{v\in\mathbb{R}^{3N_t}}{\operatorname{argmin}}(v-v_0)^T Q(v-v_0)+\mu\sum_{i=1}^{3}(S_i v)^T\mathcal{L}_t(S_i v), \quad (6)$$

where $\{S_i\}_{i=1,2,3}$ is a selection matrix for each of the 3D components respectively, and $\mathcal{L}_t$ is the Laplacian matrix of the graph $\mathcal{G}_t$. The motion field $v_0 = [v_t(1), v_t(2), \cdots, v_t(N_t)]^T \in \mathbb{R}^{3N_t}$ is the concatenation of the initial motion vectors, with $v_t(m) = \mathbf{0}_{3\times 1}$, if $m \notin \mathcal{V}_t^S$. We note that the optimization problem consists of a fitting term that penalizes the excess matching score on the sparse set of matching nodes, and of a regularization term that imposes smoothness of the motion vectors in each of the position components independently. The tradeoff between the two terms is defined by the constant μ. A small μ promotes a solution that is closed to v_0, while a big μ favors a solution that is very smooth. Similar regularization techniques, which are based on the notion of smoothness of the graph Laplacian, have been widely used in the semi-supervised learning literature [49], [50]. The corresponding optimization problem is convex and it has a closed form solution given by

$$v_t^* = \left(Q + \mu\sum_{i=1}^{3} S_i^T\mathcal{L}_t S_i\right)^{-1} Q v_0, \quad (7)$$

which can be computed iteratively using MINRES-QLP [51] in large systems. With a slight abuse of notation, we will from now on denote as v_t^* the reshaped motion vectors of dimensionality $3 \times N_t$, where each row represents the motion in one of the three coordinates. Finally, $v_t^*(m) \in \mathbb{R}^3$ denotes the 3D motion vector of node $m \in \mathcal{V}_t$.

D. Implementation details

The proposed spectral features can be efficiently computed by approximating the spectral graph wavelets with Chebyshev polynomials of degree M, as described in [5]. Given this approximation, the wavelet coefficients at each scale can then be computed as a polynomial of $\mathcal{L}$ applied to a graph signal f. The latter can be performed in a way that accesses $\mathcal{L}$ only through iterative matrix-vector multiplications. The polynomial approximation can be particularly efficient when the graph is sparse, which is indeed the case of our K-NN graph. Using a sparse matrix representation, the computation cost of applying $\mathcal{L}$ to a vector is proportional to the number $|\mathcal{E}|$ of nonzero edges in the graph. The overall computational complexity is $\mathcal{O}(M|\mathcal{E}|+N(M+1)(|\mathcal{S}|+1))$ [5], where $|\mathcal{S}|$ are the number of scales. Moreover, this approximation avoids the need to compute the complete spectrum of the graph Laplacian matrix. Thus, the computational cost of the features can be substantially reduced.

Regarding the computation of correspondences, we note that the motion between consecutive frames is expected to be relatively smooth. We can avoid computing pairwise distances with all the vertices of the reference frame, by only comparing with vertices whose distance in geometry is smaller than a predefined threshold. Moreover, although in our experiments we have used K-means clustering, dividing the space into small blocks could be enough for our purposes. An example is the procedure followed in [7], where for efficiency the octree is divided into smaller blocks containing $k \times k \times k$ voxels, where k is relatively small. Thus, in the case when the number of vertices is big and K-means may not be appropriate for grouping them, the procedure that we describe above can be very efficient and possibly applied in real time.

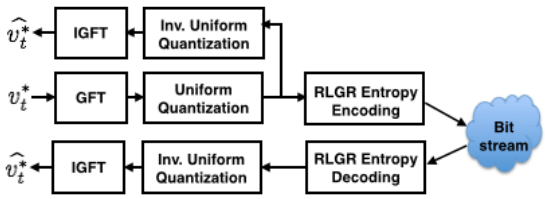


Fig. 5. Schematic overview of the motion vector coding scheme. The motion vectors v_t^* between two consecutive frames of the sequence are transformed in the graph Fourier domain, quantized uniformly, and sent to entropy coded. The decoder performs the reverse procedure to obtain $\widehat{v_t^*}$.

V. Compression of 3D point cloud sequences

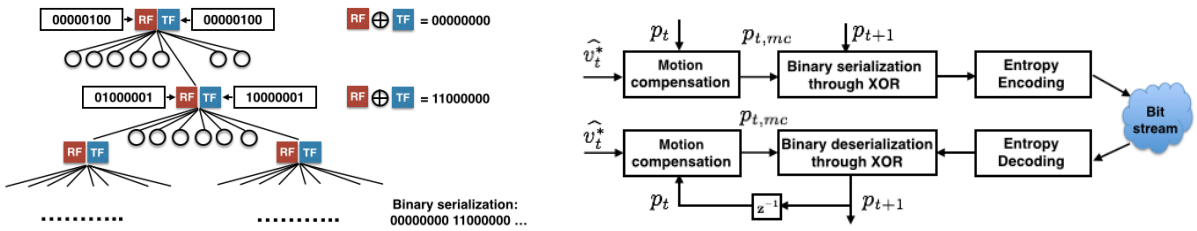
We describe now how the above motion estimation can be used to reduce temporal redundancy in the compression of 3D point cloud sequences, as shown in Fig. 3. The first frame of the sequence is always encoded using intra-frame coding [52], [7]. For the rest of the frames, we code the motion vectors by transforming them to the graph Fourier domain. We assume that the reference frame has already been sent and is known to the decoder. Coding of the 3D positions is then performed by comparing the structural difference between the target frame ($\mathcal{I}_{t+1}$) and the motion compensated reference frame ($\mathcal{I}_{t,mc}$). Temporal redundancy in color compression is finally exploited by encoding the difference between the target frame and the color prediction obtained with motion compensation.

A. Coding of motion vectors

We recall that, for each pair of two consecutive frames $\mathcal{I}_t$, $\mathcal{I}_{t+1}$, the sparse set of motion vectors is initially smoothed at the encoder side. The estimated dense motion field is then transmitted to the decoder. We exploit the fact that the graph Fourier transform is suitable for compressing smooth signals [53], [42], by coding the motion vectors in the graph Fourier domain. In particular, since the motion v_t^* is estimated in each of the nodes of the graph $\mathcal{G}_t$, we use the eigenvectors $\chi_t = [\chi_{t,0}, \chi_{t,1}, ..., \chi_{t,N_t-1}]$ of the graph Laplacian operator corresponding to the graph $\mathcal{G}_t$ of the reference frame, to transform the motion in each of the 3D directions separately such as

$$F_{v_t^*}(\lambda_\ell) =< v_t^*, \chi_{t,\ell} >, \quad \forall \ell = 0, 1, ..., N_t - 1.$$

The transformed coefficients are uniformly quantized as $round(\frac{F_{v_t^*}}{\Delta})$, where Δ is the quantization stepsize that is constant across all the coefficients, and $round$ refers to the rounding operation. The quantized coefficients are then entropy coded independently with the adaptive run-length / golomb-rice (RLGR) entropy coder [54] and sent to the



(a) Differential encoding of consecutive frames

(b) Schematic overview of the compression architecture

Fig. 6. Illustration of the geometry compression of the target frame (TF) based on the motion compensated reference frame (RF). (a) Differential encoding of the consecutive frames where structural changes within octree occupied voxels are extracted during the binary serialization process and encoded using the XOR operator. The bit stream of the XOR operator is sent to the decoder. The figure is inspired by [6]. (b) Schematic overview of the overall 3D geometry coding scheme.

decoder. The decoder performs the reverse operations to obtain the decoded motion vectors $\widehat{v_t^*}$. Note that given that the decoder already knows the 3D positions of the reference frame, it can recover the K-NN graph. Thus, the connectivity of the graph does not have to be sent. A block diagram of the encoder and the decoder is shown in Fig. 5.

B. Motion compensated coding of 3D geometry

From the reference frame $\mathcal{I}_t$ and its quantized motion vectors $\widehat{v_t^*}$, both of which are signals on $\mathcal{G}_t$, it is possible to predict the 3D positions of the points in the target frame $\mathcal{I}_{t+1}$, which is a signal on $\mathcal{G}_{t+1}$. Since the two graphs are of different size, a vector space prediction of $\mathcal{I}_{t+1}$ from $\mathcal{I}_t$ is not possible. One can however warp $\mathcal{I}_t$ to $\mathcal{I}_{t+1}$ in order to obtain a warped frame $\mathcal{I}_{t,mc}$ that is close to $\mathcal{I}_{t+1}$. Given that the 3D positions p_t and the decoded motion vectors $\widehat{v_t^*}$ of $\mathcal{I}_t$ are known to both the encoder and the decoder, the position of node m in the warped frame $\mathcal{I}_{t,mc}$ can be estimated on both sides as

$$p_{t,mc}(m) = p_t(m) + \widehat{v_t^*}(m), \quad \forall m \in \mathcal{V}_t. \tag{8}$$

Note that the 3D coordinates of the warped frame $\mathcal{I}_{t,mc}$ remain signals on the graph $\mathcal{G}_t$.

Given the warped frame $\mathcal{I}_{t,mc}$, we use the real-time compression algorithm proposed in [6] to code the structural difference between the 3D positions of $\mathcal{I}_{t+1}$ and $\mathcal{I}_{t,mc}$. Specifically, we assume that the point clouds corresponding to $\mathcal{I}_{t,mc}$ and $\mathcal{I}_{t+1}$ have already been spatially decomposed into octree data structures at a predefined depth. By knowing the occupied voxels of the reference frame $\mathcal{I}_t$ and the motion vectors $\widehat{v_t^*}$, both the encoder and decoder are able to compute the occupied voxels of the motion compensated reference frame $\mathcal{I}_{t,mc}$ and the representative bit indicator function. The encoding of the occupied voxels of the target frame $\mathcal{I}_{t+1}$ is performed by computing the exclusive-OR (XOR) between the indicator functions for the occupied voxels in frames $\mathcal{I}_{t,mc}$ and $\mathcal{I}_{t+1}$. This can be implemented by an octree decomposition of the set of voxels that are occupied in $\mathcal{I}_{t,mc}$ but not in $\mathcal{I}_{t+1}$, or vice versa, as illustrated in Fig. 6(a). Thus, motion compensation is expected to reduce the set difference and hence the number of bits used by the octree decomposition. The decoder can eventually use the motion compensated reference frame and the bits from the octree decomposition to recover exactly the set of occupied voxels (and hence the graph and 3D positions) of the target frame $\mathcal{I}_{t+1}$. We note that the first frame of the sequence is coded based on a static octree coding scheme. A schematic overview of the encoding and decoding architecture is shown in Fig. 6(b). A detailed description of the algorithm can be found in the original paper [6].

C. Motion compensated coding of color attributes

After coding the 3D positions and the motion vectors, motion compensation is used to predict the color of the target frame from the motion compensated reference frame. While the 3D positions $p_{t,mc}$ of the points in the warped frame $\mathcal{I}_{t,mc}$ are based on the 3D positions of the reference frame $\mathcal{I}_t$ and the motion field on the graph $\mathcal{G}_t$ according to (8), the colors $c_{t,mc}$ of the warped frame $\mathcal{I}_{t,mc}$ can be transferred directly from $\mathcal{I}_t$ according to

$$c_{t,mc}(m) = c_t(m), \quad \forall m \in \mathcal{V}_t.$$

Unfortunately, the graphs $\mathcal{G}_t$ and $\mathcal{G}_{t+1}$ have different sizes and there is no direct correspondence between their nodes. However, since $\mathcal{I}_{t,mc}$ is obtained by warping $\mathcal{I}_t$ to $\mathcal{I}_{t+1}$, we can use the colors of the points in $\mathcal{I}_{t,mc}$ to predict the colors of nearby points in $\mathcal{I}_{t+1}$. To be specific, for each $n \in \mathcal{V}_{t+1}$, we compute a predicted color value $\widetilde{c_{t+1}}(n)$ by averaging the color values of the nearest neighbors NN_n in terms of the Euclidean distance of the 3D positions $p_{t+1}(n)$ and $p_{t,mc}$, i.e.,

$$\widetilde{c_{t+1}}(n) = \frac{1}{|\mathrm{NN}_n|} \sum_{m \in \mathrm{NN}_n} c_{t,mc}(m),$$

where the number of nearest neighbors $|\mathrm{NN}_n|$ is usually set to 3.

Overall, the color coding is implemented as follows. We code the first frame using the coding algorithm of [7]. For the rest of the frames, temporal redundancy in the color information is removed by coding with the graph-based compression algorithm in [7] only the residual of the target frame with respect to the color prediction obtained with the above method, i.e., $\Delta c_{t+1} = c_{t+1} - \widetilde{c_{t+1}}$. The algorithm in [7] is designed for compressing the 3D color attributes in static frames and it essentially removes the spatial correlation within each frame by coding each color component in the graph Fourier domain. The algorithm divides each octree in small blocks containing $k \times k \times k$ voxels. In each of these blocks, it constructs a

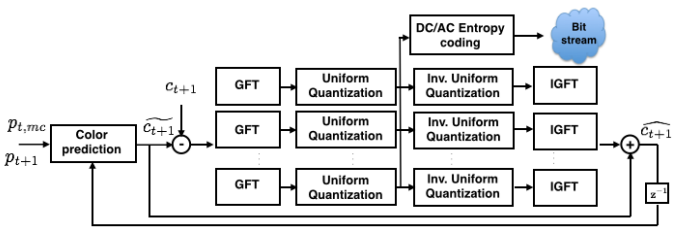


Fig. 7. Schematic overview of the predictive color coding scheme. The color residual in each block of the octree is projected in the graph Fourier domain. The graph Fourier coefficients are uniformly quantized and entropy coded based on the scheme of [7]. The bit stream is sent to the decoder where the inverse operations are performed to decode the color of the target frame.

Fig. 8. Illustrative frames of the upper body sequence.

graph and computes the graph Fourier transform. We adapt the algorithm to point cloud sequences by applying the graph Fourier transform to the color residual Δc_{t+1}. The residuals in each of the three color components are encoded separately. The graph Fourier coefficients are quantized uniformly with a stepsize Δ, as $round(\frac{\chi_{t+1}^T \Delta c_{t+1}}{\Delta})$, where $round$ denotes the rounding operator and χ_{t+1} are the eigenvectors of the graph Laplacian matrix of the corresponding block. The quantized coefficients are then entropy coded, where the structure of the graph is exploited for better efficiency. More details about the color coding scheme are given in [7] and a schematic overview is given in Fig. 7. Finally, we recall that, while the algorithm was originally used for coding static frames, in this paper we use it for coding the residual of the target frame from the motion compensated reference frame. The algorithm however remains a valid choice as the statistical distributions are carefully adapted to the actual signal characteristics.

VI. Experimental results

We illustrate in this section the matching performance of our motion estimation scheme and the performance of the proposed compression scheme. We use three different sequences that capture human bodies in motion, i.e., the yellow dress (see Fig. 2) and the man (see Fig. 1) sequences, which have been captured according to [36] and voxelized to resemble data collected by the real-time high resolution sparse voxelization algorithm [1], and a human upper body sequence, which has been captured according to [1] (see Fig. 8). The first sequence consists of 64 frames, the second one of 30 frames, and the third one of 63. The latter sequence, illustrated in Fig. 8, is more noisy and incomplete. We voxelize the point cloud of each frame in these sequences to an octree with a depth of seven. The depth of the octree acts as a sort of quantization of the 3D space. However, our motion estimation and compression scheme can be applied to any other octree level, with similar performance.

A. Motion estimation

We first illustrate the performance of our motion estimation algorithm by studying its effect in motion compensation experiments. We select two consecutive frames for each sequence, namely the reference ($\mathcal{I}_t$) and the target frame ($\mathcal{I}_{t+1}$). The graph for each frame is constructed as described in Section III. We define spectral graph wavelets of 4 scales on these graphs, and for computational efficiency, we approximate them with Chebyshev polynomials of degree 30 [5]. We select the number of representative feature points to be around 500, which corresponds to fewer than 10% of the total occupied voxels, and we compute the sparse motion vectors on the corresponding nodes by spectral matching. We estimate the motion on the rest of the nodes by smoothing the motion vectors on the graph according to (6).

In Figs. 9(a), 9(d), 9(g) we superimpose the reference and the target frames for the yellow dress, the man, and the upper body sequences accordingly in order to illustrate the motion involved between two consecutive frames. The key points used for spectral matching in each of the two frames are shown in Figs. 9(b), 9(e), 9(h), and they are represented in red for the target and in green for the reference frame. For the sake of clarity, we highlight only some of the correspondences used for computing the motion vectors. We observe that the sparse set of matching vertices are accurate and well-distributed in space for both sequences. Finally, in Figs. 9(c), 9(f), 9(i) we superimpose the target frame and the voxel representation of the motion compensated reference frame. By comparing visually these three figures to 9(a), 9(d), 9(g) respectively, we observe that in all the cases the motion compensated reference frame is much closer to the target frame than the reference frame. The result is actually true also for the quite noisy frames of the upper body sequence. The obtained results confirm that our algorithm is able to estimate accurately the motion even in pretty adverse conditions.

B. 3D geometry compression

We now study the benefits of motion estimation in the compression of geometry in 3D point cloud sequences. The compressed geometry information includes motion vectors and the geometry difference between the target frame and the motion compensated reference frame captured by the XOR encoded information. We note that the compression is performed on the whole sequence. The frames of the sequences are coded sequentially in the following way. Only the first frame is coded independently using a classical octree compression scheme based on children pattern sequence [52], while all the other frames are coded by using as a reference frame the previously coded frame. We first code the motion vectors with the proposed coding scheme of Sec. V-A. The motion signal in each of the 3D directions is coded separately.

(a) $\mathcal{I}_t + \mathcal{I}_{t+1}$ (b) Correspondence between $\mathcal{I}_t$ and $\mathcal{I}_{t+1}$ (c) $\mathcal{I}_{t,mc} + \mathcal{I}_{t+1}$

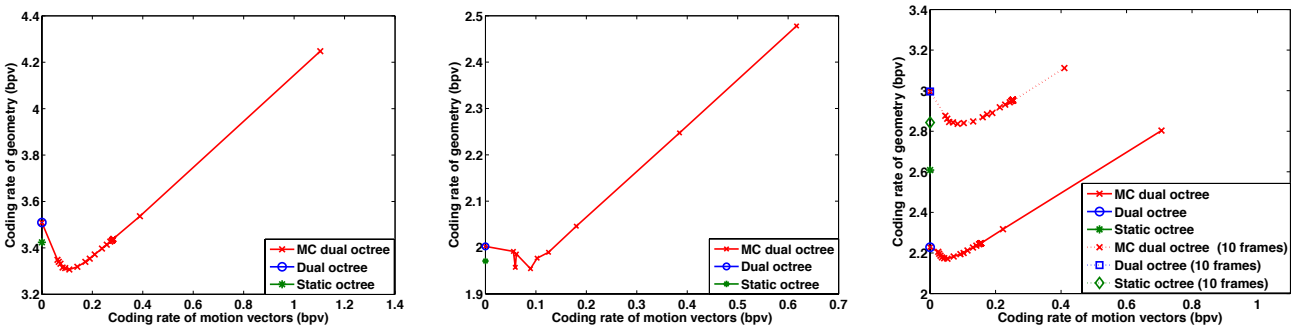
(d) $\mathcal{I}_t + \mathcal{I}_{t+1}$ (e) Correspondence between $\mathcal{I}_t$ and $\mathcal{I}_{t+1}$ (f) $\mathcal{I}_{t,mc} + \mathcal{I}_{t+1}$

(g) $\mathcal{I}_t + \mathcal{I}_{t+1}$ (h) Correspondence between $\mathcal{I}_t$ and $\mathcal{I}_{t+1}$ (i) $\mathcal{I}_{t,mc} + \mathcal{I}_{t+1}$

Fig. 9. Example of motion estimation and compensation in the yellow dress, man and upper body motion sequences. The superimposition of the reference ($\mathcal{I}_t$) and target frame ($\mathcal{I}_{t+1}$) is shown in (a), (d), and (g) while in (b), (e), (h) we show the correspondences between the target (red) and the reference frame (green). The superposition of the motion compensated reference frame ($\mathcal{I}_{t,mc}$) and the target frame ($\mathcal{I}_t$) is shown in (c), (f), and (i). Each small cube corresponds to a voxel in the motion compensated frame.

In Fig. 10 we first show the advantage of transforming the motion vectors in the graph Fourier domain, in comparison to coding directly in the signal domain, for the man sequence. Different stepsizes for uniform quantization are used to obtain different coding rates, hence different accuracies of the motion vectors. The performance is measured in terms of the signal-to-quantization noise ratio (SQNR) for a fixed number of bits per vertex. The SQNR is computed on pairs of frames. Each point in the rate distortion curve corresponds to the average over 64 frame pairs. The results confirm that coding the motion vectors in the graph Fourier domain results in an efficient spatial decorrelation of the motion signals, which brings significant gain in terms of coding rate. Similar results hold for the other two sequences, but we omit them due to space constraints.

We study next the effect of motion compensation in the coding rate of the 3D positions. We recall that for a particular depth of the tree, the coding of the geometry is lossless. There exists however a tradeoff between the overall coding rate of the geometry and the coding rate of the motion vectors as we illustrate next. In particular, we compare the motion compensated dual octree scheme as described in Sec. V-B, to the dual octree scheme of [6], and the static octree compression algorithm [52]. In Fig. 11, we illustrate the



(a) Man sequence (b) Upper body sequence (c) Yellow dress sequence

Fig. 11. Effect of the coding rate of the motion vectors on the overall coding rate of the geometry for the motion compensated dual octree algorithm. By sending the motion vectors at low bit rate (≈ 0.1 bpv), the motion compensated dual octree scheme performs slightly better than the static octree and the dual octree compression algorithm.

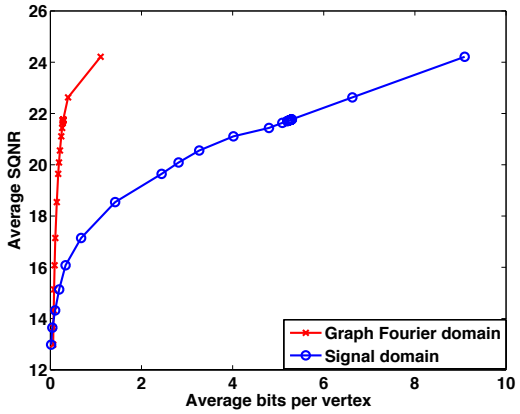


Fig. 10. Performance comparison of the average signal-to-quantization noise ratio (SQNR) versus bits per vertex (bpv) for coding the motion vectors in the graph Fourier domain and in the signal domain.

coding rate of the geometry with respect to the coding rate of the motion vectors, measured in terms of the average number of bits per vertex (bpv) over all the frames, for each of the three competitive schemes. The coding rate of the geometry includes the coding rate of the motion vectors. In Fig. 11(a), the smallest coding rate of the geometry (3.3 bpv) for the man sequence is achieved for a coding rate of the motion vectors of only 0.1 bpv. The latter indicates that coarse quantization of the motion vectors is enough for an efficient geometry compression. A smaller number of bits per vertex however tends to penalize the effect of motion compensation, giving an overall coding rate that approaches the one of the dual octree compression scheme. Of course, a finer coding of the motion vectors increases the overhead in the total coding rate of the geometry. The corresponding numbers for the static octree and the dual octree compression scheme are approximately 3.42 and 3.5 respectively. These results indicate that the temporal structure captured by the dual octree compression scheme is not sufficient to improve the coding rate with respect to the static octree compression algorithm. Motion compensation is thus needed to remove the temporal correlation. However, the overall gain that we obtain is small and corresponds to 3.5% bpv and 5.7% bpv with respect to the static octree and the dual octree compression algorithm respectively. Moreover, motion compensation does not seem to bring a significant gain in the coding of the geometry of the upper body sequence in Fig. 11(b). As we already mentioned before this sequence contains frames that are quite noisy. As a result, the performance of motion compensation seems to deteriorate, especially in the case of consecutive frames with appearing or disappearing nodes.

In order to study the effect of the motion in the compression performance, we perform two different tests in the yellow dress sequence. In the first test, we compress the entire yellow dress sequence, which is a low motion sequence. In the second test, we sample the sequence by keeping only 10 frames that are characterized by higher motion between consecutive frames. We then compress the geometry for this new smaller sequence. In Fig. 11(c), we observe that when the motion is low, the motion compensated dual octree and the dual octree compression algorithms are much more efficient in coding the geometry in comparison to the static octree compression algorithm. Moreover, the motion compensated dual octree scheme requires a slightly smaller number of bits per vertex (2.2 bpv), for a coding rate of the motion vectors of 0.1 bpv. The coding rate for the dual octree and the static octree compression algorithm are respectively 2.24 and 2.6 bpv. On the other hand, the static octree compression scheme outperforms the dual octree compression algorithm in the higher motion sequence of 10 frames, with coding rates of 3 and 2.8 bpv respectively. The motion compensated dual octree compression algorithm can close the gap between these two methods by achieving a coding rate of 2.8 bpv. We note that this performance is achieved for an overhead of 0.15 bpv for coding the motion vectors. Due to this overhead, the performance of the static octree and the motion compensated dual octree compression algorithm are relatively close.

C. Color compression

In the next set of experiments, we use motion compensation for color prediction, as described in Section V-C. That is, using the smoothed motion field, we warp the reference frame $\mathcal{I}_t$ to the target frame $\mathcal{I}_{t+1}$, and predict the color of each point in $\mathcal{I}_{t+1}$ as the average of the three nearest points in the warped

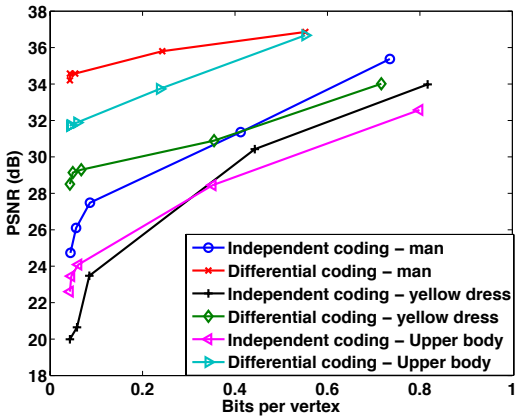


Fig. 12. Color compression performance (dB) vr. bits per vertex for independent and differential coding on the three datasets for a quantization stepsize of $\Delta = [32,\ 64,\ 256,\ 512,\ 1024]$.

(a) (b) (c)

Fig. 13. Rendering results of a point cloud frame from the yellow dress sequence compressed at a quantization stepsize of $\Delta = 1024$, and $\Delta = 32$; (a) original point cloud, (b) voxalized and decoded frame for $\Delta = 1024$, and (c) voxalized and decoded frame for $\Delta = 32$.

frame $\mathcal{I}_{t,mc}$. We fix the coding rate of the motion vectors to 0.1 bpv and, for the sake of comparison, we compute the signal-to-noise ratio (SNR) after predicting the color in the following three different ways: (i) the colors of points in the target frame are predicted from their nearest neighbors in the warped frame $\mathcal{I}_{t,mc}$, (SNR$_{mc}$) (ii) the colors of points in the target frame are predicted from their nearest neighbors in $\mathcal{I}_t$ (SNR$_{previous}$), and (iii) the colors of points in the target frame are predicted as the average color of all the points in $\mathcal{I}_t$ (SNR$_{avg}$). The SNR for frame $\mathcal{I}_{t+1}$ is defined as $\text{SNR} = 20\log_{10}\frac{\|c_{t+1}\|}{\|c_{t+1}-\widetilde{c_{t+1}}\|}$, where we recall that c_{t+1} and $\widetilde{c_{t+1}}$ are the actual color and the color prediction respectively. The prediction error is measured by taking pairs of frames in the sequence and computing the average over all the pairs. The obtained values are shown in Table I. We notice that for three sequences motion compensation can significantly reduce the prediction error, by obtaining an average gain in the color prediction of 2.5 dB and 8-10 dB with respect to simple prediction based on the color of the nearest neighbors in the reference frame, and the average color of the reference frame respectively.

TABLE I
COLOR PREDICTION ERROR (SNR IN dB)

Sequence	SNR$_{mc}$	SNR$_{previous}$	SNR$_{avg}$
Yellow dress	17	15	6.5
Man	13	10.5	4
Upper body	9.8	7.5	1.2

We finally use the prediction obtained from our motion estimation and compensation scheme to build a full scheme for color compression, that is based on a prediction path of a series of frames. Compression of color attributes is obtained by coding the residual of the target frame with respect to the color prediction obtained with the scheme described in Section V-C. In our experiments, we code the color in small blocks of $16\times16\times16$ voxels. We measure the PSNR obtained for different levels of the quantization stepsize in the coding of the color information, hence different coding rates, for both independent [7] and differential coding. The results for the three datasets are shown in Fig. 12. Each point on the curve corresponds to the average PSNR of the RGB components across the first ten frames of each sequence, obtained for a quantization stepsize of $\Delta = [32, 64, 256, 512, 1024]$ respectively. We observe that at low bit rate ($\Delta = 1024$), differential coding provides a gain with respect to independent coding of approximately 10 dB for all the three sequences. On the other hand, at high bit rate, the difference between independent and differential coding tends to become smaller, as both methods can code the color quite accurately. Examples of the decoded frames of the yellow dress sequence for $\Delta = 32, 1024$ are shown in Fig. 13. Finally, we note that the gain in the coding performance is highly dependent on the length of the prediction path. As the number of predicted frames increases, the accumulated quantization error from the previously coded frames is expected to lead to a gradual PSNR degradation that is more significant at low bit rate. This can be mitigated by periodic insertion of reference frames, and by optimizing the number of predicted frames between consecutive reference frames.

D. Discussion

Our experimental results have shown that motion compensation is beneficial overall in the compression of 3D point cloud sequences. The main benefit though is observed in the coding of the color attributes, providing a gain of up to 10 dB with respect to coding each frame independently. The gain in the compression of the 3D geometry is only marginal due to the overhead for coding the motion vectors. Moreover, from the experimental validation in our datasets, we observe that the proposed motion compensated geometry compression framework that is based on the differential coding of consecutive octree graph structures is the most expensive part of the overall compression system. Only a very coarse quantization of the motion vectors is sufficient to achieve an overall good compression rate. We expect however the bit rate to increase with the level of the motion. Empirically, for each

vertex in the man sequence, we need 0.1-0.2 bits to code the motion vectors, 0.1-0.3 bits for the color residual, and 3.3 bits for the geometry compression. Similar observations hold for the other datasets.

VII. Conclusions

In this paper, we have proposed a novel compression framework for 3D point cloud sequences that is based on exploiting temporal correlation between consecutive point clouds. We have first proposed an algorithm for motion estimation and compensation. The algorithm is based on the assumption that 3D models are representable by a sequence of weighted and undirected graphs where the geometry and the color of each model can be considered as graph signals. Correspondence between a sparse set of nodes in each graph is first determined by matching descriptors based on spectral features that are localized on the graph. The motion on the rest of the nodes is interpolated by exploiting the smoothness of the motion vectors on the graph. Motion compensation is then used to perform geometry and color prediction. Finally, these predictions are used to differentially encode both the geometry and the color attributes. Experimental results have shown that the proposed method is efficient in estimating the motion and it eventually provides significant gain in the overall compression performance of the system.

There are a few directions that can be explored in the future. First, it has been shown in our experimental section that a significant part of the bit budget is spent for the compression of the 3D geometry, which given a particular depth of the octree, is lossless. A lossy compression scheme that permits some errors in the reconstruction of the geometry could bring non-negligible benefits in terms of the overall rate-distortion performance. Second, the optimal bit allocation between geometry, color and motion vector data stays an interesting and open research problem, due mainly to the lack of a suitable metric that balances geometry and color visual quality. Third, the estimation of the motion is done by computing features based on the spectral graph wavelet transform. Features based on data-driven dictionaries, such as the ones proposed in [55], are expected to increase significantly the matching, and consequently the compression performance.

Acknowledgments

The authors would like to thank Cha Zhang and Dinei Florêncio for providing the code for the color encoder and help in the color compression experiments, Alvaro Collet et al. for providing the Yellow Dress and Man sequences, and Charles Loop, Qin Cai, Mingsong Dou, and Ricardo L. de Queiroz for providing the upper body sequence.

References

[1] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *High-Performance Graphics Conf.*, 2013, pp. 73–79.

[2] Y. Huang, J. Peng, C. C. J. Kuo, and M. Gopi, "A generic scheme for progressive point cloud coding," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 2, pp. 440–453, 2008.

[3] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Symposium on Point-Based Graphics*, Jul. 2006.

[4] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based motion estimation and compensation for dynamic 3D point cloud compression," in *IEEE Int. Conf. on Image Process.*, Sep. 2015.

[5] D. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2010.

[6] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *IEEE Int. Conf. on Robotics and Automation*, May 2012.

[7] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *IEEE Int. Conf. on Image Process.*, Sep. 2014, pp. 2066 – 2070.

[8] T. Ochotta and D. Saupe, "Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields," in *Eurographics Conf. on Point-Based Graphics*, 2004, pp. 103–112.

[9] O. Devillers and P-M. Gandoin, "Geometric compression for interactive transmission," in *IEEE Visualization*, 2000, pp. 319–326.

[10] J. Digne, R. Chaine, and S. Valette, "Self-similarity for accurate compression of point sampled surfaces," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 155–164, 2014.

[11] J-Y. Sim, C-S. Kim, and S-U. Lee, "Lossless compression of 3D point data in QSplat representation," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1191–1195, Dec. 2005.

[12] J-Y. Sim and S-U. Lee, "Compression of 3D point visual data using vector quantization and rate-distortion optimization," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 305–315, Apr. 2008.

[13] J. Ahn, K. Lee, J. Sim, and C. Kim, "Large-scale 3D point cloud compression using adaptive radial distance prediction in hybrid coordinate domains," *IEEE Journal Selec. Topics Signal Process.*, vol. 9, no. 3, pp. 422–434, 2015.

[14] M. Irani and P. Anandan, "About direct methods," in *Inter. Workshop on Vision Algorithms: Theory and Practice*, 2000, pp. 267–277.

[15] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Inter. Workshop on Vision Algorithms: Theory and Practice*, 2000, pp. 278–294.

[16] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Inter. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European Conf. on Computer Vision*, 2006, pp. 404–417.

[18] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," in *British Machine Vision Conf.*, 2002, pp. 384–393.

[19] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Inter. Conf. on Multimedia*, 2007, pp. 357–360.

[20] F. Tombari, S. Salti, and L. di Stefano, "Unique signatures of histograms for local surface description," in *European Conf. on Computer Vision*, Sept. 2010, pp. 356–369.

[21] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching," in *IEEE Conf. on Comp. Vision and Pattern Recogn.*, Jun. 2009, pp. 373–380.

[22] F. Tombari, S. Salti, and L. di Stefano, "A combined texture-shape descriptor for enhanced 3D feature matching," in *IEEE Int. Conf. on Image Process.*, 2011, pp. 809–812.

[23] H. Chen and B. Bhanu, "3D free-form object recognition in range images using local surface patches," *Pattern Recogn. Lett.*, vol. 28, no. 10, pp. 1252–1262, Jul. 2007.

[24] I. Sipiran and B. Bustos, "A robust 3D interest points detector based on harris operator," in *Eurographics Conf. on 3D Object Retrieval*, 2010, pp. 7–14.

[25] L. A. Alexandre, "3D descriptors for object and category recognition: a comparative evaluation," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems*, 2012.

[26] J. Peng, Ch. Kim, and C. Jay Kuo, "Technologies for 3D mesh compression: A survey," *Journal of Vis. Comun. and Image Represent.*, vol. 16, no. 6, pp. 688–733, Dec. 2005.

[27] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Trans. on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 47–61, Jan. 1999.

[28] M. Alexa and W. Müller, "Representing animations by principal components," *Comput. Graph. Forum*, vol. 19, no. 3, pp. 411–418, Sep. 2000.

[29] L. Váša and V. Skala, "Geometry driven local neighborhood based predictors for dynamic mesh compression," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1921–1933, 2010.

[30] H. Q. Nguyen, P. A. Chou, and Y. Chen, "Compression of human body sequences using graph wavelet filter banks," in *IEEE Int. Conf. Acoustic, Speech, and Signal Process.*, May 2014, pp. 6152–6156.

[31] S.-R. Han, T. Yamasaki, and K. Aizawa, "Time-varying mesh compression using an extended block matching algorithm," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 17, no. 11, pp. 1506–1518, Nov. 2007.

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

[33] A. Doumanoglou, D. S. Alexiadis, D. Zarpalas, and P. Daras, "Toward real-time and efficient compression of human time-varying meshes," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 24, no. 12, pp. 2099–2116, 2014.

[34] X. Gu, S. J. Gortler., and H. Hoppe, "Geometry images," in *Annual Conf. on Computer Graphics and Interactive Techniques*, 2002, pp. 355–361.

[35] H. M. Briceno, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry Videos: A New Representation for 3D Animations ," in *ACM Symposium on Computer Animation*, 2003, pp. 136–146.

[36] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality streamable free-viewpoint video," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 69:1–69:13, Aug. 2015.

[37] H. Habe, Y. Katsura, and T. Matsuyama, "Skin-off: Representation and compression scheme for 3D video," in *Picture Coding Symp.*, 2004, pp. 301–306.

[38] J. Hou, L. Chau, N. Magnenat-Thalmann, and Y. He, "Compressing 3D human motions via keyframe-based geometry videos," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 25, no. 1, pp. 51–62, 2015.

[39] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *14th Annual Conf. on Comp. Graphics and Interactive Techniques*, 1987, pp. 163–169.

[40] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[41] F. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

[42] C. Zhang and D. Florêncio, "Analyzing the optimality of predictive transform coding using graph-based models," *IEEE Signal Process. Lett.*, vol. 20, no. 1, pp. 106–109, 2013.

[43] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Symposium on Geometry Process.*, 2009, pp. 1383–1392.

[44] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *IEEE Conf. on Comp. Vision and Pattern Recogn.*, 2010, pp. 1704–1711.

[45] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *IEEE Int. Conf. Comp. Vision*, 2011, pp. 1626–1633.

[46] R. Litman and A. M. Bronstein, "Learning spectral descriptors for deformable shape correspondence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 171–180, 2014.

[47] N. Hu, R. M. Rustamov, and L. Guibas, "Stable and informative spectral signatures for graph matching," in *IEEE Conf. on Comp. Vision and Pattern Recogn.*, Jun. 2014.

[48] W. H. Kim, M. K. Chung, and V. Singh, "Multi-resolution shape analysis via non-euclidean wavelets: Applications to mesh segmentation and surface alignment problems," in *IEEE Conf. on Comp. Vision and Pattern Recogn.*, 2013, pp. 2139–2146.

[49] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Inter. Conf. on Machine Learning*, 2003, vol. 13, pp. 912–919.

[50] D. Zhou, O. Bousquet, T .N. Lal, J. Weston, and B. Scholkopf, "Learning with local and global consistency," in *Advances in Neural Information Process. Systems*, 2004, vol. 16, pp. 321–328.

[51] S.-C. T. Choi and M. A. Saunders, "MINRES-QLP for symmetric and hermitian linear equations and least-squares problems," *ACM Trans. Math. Softw.*, vol. 40, no. 2, 2014.

[52] J. Katajainen and E. Mäkinen, "Tree compression and optimization with applications," *Int. Journ. Found. Comput. Science*, vol. 1, no. 4, pp. 425–448, 1990.

[53] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *IEEE Int. Conf. Acoustic, Speech, and Signal Process.*, Jul. 2012, pp. 3921–3924.

[54] H. S. Malvar, "Adaptive run-length / golomb-rice encoding of quantized generalized gaussian sources with unknown statistics," in *Data Compression Conf.*, Mar. 2006.

[55] D. Thanou, D. I Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849–3862, Aug. 2014.