SCHOOL OF ENGINEERING - STI
SIGNAL PROCESSING INSTITUTE
Ivana Tosic, Pascal Frossard and Pierre Vandergheynst

CH-1015 LAUSANNE

Telephone: +4121 6935615
Telefax: +4121 6937600
e-mail: ivana.tosic@epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PROGRESSIVE CODING OF 3D OBJECTS BASED ON OVERCOMPLETE DECOMPOSITIONS.

Ivana Tosic, Pascal Frossard and Pierre Vandergheynst

Swiss Federal Institute of Technology Lausanne (EPFL)

Signal Processing Institute Technical Report

TR-ITS-2005.026

October 10th, 2005

Part of this work has been submitted to the IEEE Transactions on Circuits and Systems for Video Technology.

This work has been supported by the Swiss National Science Foundation, under grants PP-002-68737 and 200021-101880/1.

Progressive coding of 3D objects based on overcomplete decompositions.

Ivana Tosic, Pascal Frossard and Pierre Vandergheynst
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute, Lausanne 1015, Switzerland
{ivana.tosic, pascal.frossard, pierre.vandergheynst}@epfl.ch
Fax: +41 21 693 7600, Phone: +41 21 693 2601

***Abstract*— This paper presents a progressive coding scheme for 3D objects, based on an overcomplete decomposition of the 3D model on a sphere. Due to increased freedom in the bases construction, redundant expansions have shown interesting approximation properties in the decomposition of signals with multidimensional singularities organized along embedded submanifolds. We propose to map simple 3D models on 2D spheres and then to decompose the signal over a redundant dictionary of oriented and anisotropic atoms that live on the sphere. The signal expansion is computed iteratively with a Matching Pursuit algorithm, which greedily selects the most prominent components of the 3D model. The decomposition therefore inherently represents a progressive stream of atoms, which is advantageously used in the design of scalable representations. An encoder is proposed that compresses the stream of atoms by adaptive coefficient quantization, and entropy coding of atom indexes. Experimental results show that the novel coding strategy outperforms state-of-the-art progressive coders in terms of distortion, mostly at low bit rate. Furthermore, since the dictionary is built on structured atoms, the representation simultaneously offers an increased flexibility. This enables easy stream manipulations, and we finally illustrate this advantage in the design of a view-dependent transmission scheme.**

I. Introduction

The widespread use of 3D data in many areas like gaming or entertainment, architecture, robotics, medical imaging, geographic information systems, has created an essential need for efficient compression of 3D models. On the other side, the increasingly large variety of decoding engines, with heterogeneous capabilities and connectivity, imposes a need for multi-resolution representation, as well as low-complexity decoders, based on generic purpose hardware. The most common approaches for 3D data representation are based on polygonal meshes, which are described by both geometry (i.e., the position of vertices in space) and connectivity information, as well as optional information about normals, colors and textures. It generally results in models built on arbitrarily defined and non-uniform grids, which lead to efficient decoding performance on dedicated hardware. Such representations stay however quite voluminous, and do not provide a lot of flexibility for adaptation to the requirements of specific applications, or to the constraints imposed by the decoding engine.

The aim of this paper is to propose a novel coding scheme for 3D objects, which can provide a progressive representation with flexibility in the stream manipulation, whilst achieving good compression performance. A progressive representation enables the decoder to construct a model at different resolutions simply by proper stream truncation to meet a well-chosen rate-distortion trade-off. In the same time, a flexible representation provides the possibility to manipulate the model in the compressed domain, to decode the model at different sizes, or from different viewpoints, for example. Towards this objective, we first propose to move away from restrictive representation techniques on non-uniform grids, by resampling 3D data on a regular spherical grid, thus reducing the dimension of the input data into a 2D data set. A 3D surface which can be represented as a function on a 2-D sphere is a genus-zero[1] surface which has only one intersection point with each radial line from the center of the point cloud. We will reference to these models as *simple genus-zero*, or *star-shape* models. Next, we show that the representation of more complex models is feasible by decomposition into several spherical mappings. A representation defined on a regular grid excludes the need for coding samples positions, and enables the usage of 2D signal transform coding techniques for 3D models.

Inspired by the good efficiency of discrete wavelet transform for image compression, the first choice would be the use of the spherical wavelet transform for the representation of 3D models. However, similarly to contours in natural images, 3D objects often present numerous multi-dimensional singularities that are organized along embedded submanifolds. It has been shown that wavelets are not optimal at representing such features like contours because they cannot deal with the geometrical regularity of these characteristics. We therefore propose to represent the 3D model as a series of oriented, and anisotropically refined functions taken from a redundant dictionary of atoms. These atoms are edge-like functions living on the 2D sphere, which can take arbitrary positions, shapes and orientations. In order to capture the low-frequency components of the 3D model, low frequency atoms built on 2-dimensional gaussian functions finally complement the dictionary. We propose to use the iterative Matching Pursuit algorithm to greedily build the signal approximation. Matching Pursuit (MP) inherently produces a progressive stream of atoms, which can be decoded with a reduced complexity. A coefficient quantization step, as well as an entropy coding stage for atom parameters are proposed to generate a progressive and flexible compressed

[1] A mesh has a genus g, iff one can cut the mesh along 2g closed loops without disconnecting the mesh

representation of the 3D model. The proposed encoder offers better compression performance at low rate, compared to classical shape compression methods, while still providing interesting scalability properties. Experimental results show that the PSNR gain of the proposed coding scheme over the state-of-the-art schemes even reaches 3dB at low bit rates. In the same time, the MP encoder produces a completely progressive stream, which can be efficiently truncated at any arbitrary rate. We finally demonstrate the increased flexibility of the proposed representation by presenting a view-dependent coding algorithm, typically useful in interactive applications with scarce bandwidth resources.

This paper is organized as follows. In section II we give an overview of related work on 3D model compression schemes and redundant expansions. Section III focuses on the construction of the over-complete dictionary adapted to 3D object properties, while in section IV each step of the proposed 3D object coding scheme is described in more detail. Section V presents experimental results, and comparisons with state-of-the-art algorithms. In section VI we explain how the proposed scheme is used for view-dependent applications and present the obtained results. Finally, Section VII concludes the paper.

II. Related Work

Numerous works have addressed the coding of 3D models, and we just mention here the most relevant ones in the context of the present paper. The first mesh geometry compression scheme, introduced by Deering [1], was based on *triangle strips* and *triangle fans*, and implemented in GL [2] and OpenGL [3]. In GL, triangles are ordered to form strips, whose connectivity is defined with a *marching bit* per triangle; it specifies which of the two free edges of the current triangle the next triangle has to be attached to. In OpenGL, triangles are attached alternatively on left and right edges, and no connectivity information is transmitted. The drawback of this technique is that most meshes have twice as many faces as vertices: each vertex has to be transmitted twice, in average.

Taubin and Rossignac later introduced the Topological Surgery (TS) scheme [4], which is a single-resolution manifold triangular mesh compression scheme that preserves the connectivity. After extensions to arbitrary manifold meshes, TS has become part of MPEG-4 standard. In TS, Faces are interconnected by a *face forest*, spanning the dual graph of the mesh [2]. The edges that do not belong to the face forest, then define a *vertex graph* and interconnect all the vertices of the mesh. A simple polygon connectivity mesh is obtained by cutting a mesh through a vertex graph, and is eventually encoded along with the vertex graph.

In order to obtain a multi-resolution representation with mesh-based coding schemes, several works proposed mesh decimation techniques, which reduce the number of triangles, vertices and edges. They provides initially a coarse mesh model that progressively refines by insertion of more detailed information. One of the first progressive transmission schemes for multi-resolution triangular manifold meshes has been introduced by Hoppe in [5]. A triangular manifold mesh is represented by a base mesh followed by a sequence of successive vertex split refinements. Taubin has introduced the Progressive Forest Split (PFS) scheme [6], which highly reduces the number of levels of detail, and thus unnecessary information. A forest split operation is in essence described by a group of consecutive edge split operations. Together with Topological Surgery (TS), PFS represent the core of 3D mesh coding in the MPEG-4 standard.

A common characteristic of multi-resolution mesh-based compression schemes mentioned above is that most of the geometry information of a coarse mesh is embedded within a finer mesh, except for a set of vertices or edges that result from vertex or edge split operations. This kind of surface sampling does not necessarily lead to the best approximation at a given resolution. On the other side, by representing a 3D model as a continuous function on a 2-d surface, positions of vertices are determined by uniform sampling of this function so they are different from one resolution to another. This results in equal approximation enhancement over the 3D object surface, which is an important advantage of 2-d surface methods versus mesh-based methods. Moreover, the translation of a 3D object into a continuous space offers a possibility of employing various signal transformation techniques towards building fully progressive representations.

Amongst the alternatives to mesh-based approaches, and perhaps closer to the approach proposed in this paper, Schröder and Sweldens [7] proposed one of the earliest works that represent 3D models as functions defined on the surface of a sphere. They introduced a lifting scheme to construct bi-orthogonal spherical wavelets with customized properties. Shape compression using spherical wavelets has become recently an active area of research. The progressive coding scheme introduced by Khodakovsky et al. [8] uses wavelet transform, zerotree coding and subdivision-based reconstruction to improve the compression ratio. Hoppe and Praun [9] describe a shape compression technique using spherical geometry images, which represent the surface remeshed into a regular 2D grid. In comparison to ordinary image wavelets, spherical wavelets are shown to provide better compression performance for surfaces that can be nicely parametrized on the sphere. However, the related compression techniques suffer from rippling artifacts for surfaces with long extremities.

III. Redundant representations on the 2-D sphere

A. Preliminaries

Redundant expansions have shown interesting approximation properties in the decomposition of signals with multidimensional singularities organized along embedded submanifolds, like images [10, 11]. Redundant expansions provide a lot of freedom in the design of the bases or dictionaries. In particular, it is possible to include rotation or anisotropy in the basis functions. These two properties are keys to the development of efficient algorithms for the approximation of multi-dimensional signals. In such a context, separable orthogonal bases like wavelets, have shown their limitations in terms of approximation rate, whilst they stay optimal for 1-D continuous signals with point-like singularities.

[2]The dual graph of a polygonal mesh is the graph composed of the mesh faces as dual graph nodes, and the internal mesh edges as dual graph edges

Since 3D models are signals composed of multi-dimensional features, we propose to represent them as a series of atoms, taken from a redundant dictionary of functions. Dictionaries are in general constructed as a set of different waveforms, where each waveform is defined by a generating function. Each generating function can serve as a base for building the overcomplete dictionary, simply by changing the function parameters or indexes (e.g., position or scale indexes). While there is a priori no restriction on the construction of the dictionary, its construction from one or several generating functions advantageously leads to structured dictionaries, whose indexes directly correspond to atom characteristics. Furthermore, the storage or transmission of the dictionary becomes unnecessary, since atoms can be reconstructed only from their indexes.

The construction of the dictionary is certainly the most important step for efficient approximation algorithms. Increasing the number of functions generally increases the redundancy of the dictionary, and thus the approximation performance; there is an increasingly high probability that prominent signal features can be efficiently captured by a single atom. In the same time, it also increases the size of the dictionary, and most probably augments the coding rate, and the search complexity. We now discuss more in detail the dictionary construction for the overcomplete expansion of simple 3D models on the 2-D sphere. It involves the three following steps:

- definition of the generating function(s) on the sphere,
- definition of atoms motion on the sphere, and their rotation around their axis,
- implementation of the anisotropic scaling of atoms.

Since the signal to be approximated is defined in the space of square-integrable functions on a unit 2-sphere S^2, denoted as $L^2(S^2)$ (i.e., $f(\theta,\varphi) \in L^2(S^2)$), the atoms have obviously to live in the same space. Let g denote a generating function on the 2D sphere. By combining motion, rotation and scaling, we form an overcomplete set of atoms g_γ, where $\gamma = (\theta, \varphi, \psi, a_1, a_2) \in \Gamma$ is the atom index. This index is described by 5 parameters that respectively represent the position of the atom on the sphere (θ, φ), its orientation (ψ), and the scaling parameters (a_1, a_2). In order to finally map the atoms on the sphere, we use an inverse stereographic projection from the complex plane, to the 2D sphere. The stereographic projection [12] at the North pole, can be expressed as $\Phi : S^2 \rightarrow \mathbb{C}$, where $\mathbb{C}$ represents the complex plane (see Figure 1). It can be written as :

$$\Phi(\omega) = \vec{v} = \rho e^{j\varphi} = 2tan\left(\frac{\theta}{2}\right) e^{j\varphi} \ , \qquad (1)$$

with $\omega \equiv (\theta, \varphi)$ and $0 \leq \theta \leq \pi, -\pi \leq \varphi \leq \pi$. Since the stereographic projection is bijective, any point with polar coordinates (ρ, φ) and represented by a vector $\vec{v} = (\rho\, cos\varphi, \rho\, sin\varphi)$ on the tangent plane, can be uniquely mapped back to the 2D sphere. We use that property in the design of the dictionary, as presented below.

B. Generating functions

Under the assumption that simple 3D models are mostly composed of smooth surfaces, and singularities aligned on pieces of great circles, we propose to build the dictionary over two generating functions. First, in order to efficiently capture the singularities, we use a generating function that resembles to a piece of contour on the sphere. In the space $L^2(\mathbb{R}^2)$, it can be written as:

$$g_{rect}(\vec{v}) = -\frac{1}{K}\left(4x^2 - 2\right)\exp\left(-\frac{\left(x^2+y^2\right)}{4}\right), \qquad (2)$$

where $\vec{v} = (x, y)$ is a vector in $\mathbb{R}^2$, and K is a normalization factor. Note that this function is very similar to the one that has been efficiently used for images coding in [10, 11], which is Gaussian in one direction and its second derivative in the other direction:

$$g_{image}(\vec{v}) = \frac{1}{\tilde{K}}\left(4x^2 - 2\right)\exp\left(-\left(x^2+y^2\right)\right). \qquad (3)$$

The function defined in Eq. (2) differs however from Eq. (3), in the sense that it generates longer atoms (slower decay) in the direction of Gaussian, but with the same sharp decay in the direction of its derivative. This leads to improved approximation of singularities on the 2D sphere. The generating function from Eq. (2) can further be expressed in polar coordinates, as:

$$g_{rect}(\rho, \varphi) = -\frac{1}{K}\left(4\rho^2 cos^2\varphi - 2\right)\exp\left(-\frac{\rho^2}{4}\right). \qquad (4)$$

By inverse stereographic projection $\Phi^{-1} : \mathbb{R}^2 \rightarrow S^2$, the generating function is mapped on the sphere, and can be written as:

$$g_{HF}(\theta,\varphi) = -\frac{1}{K_1}\left(16\tan^2\left(\frac{\theta}{2}cos^2\varphi - 2\right)\right) \cdot \exp\left(-\tan^2\left(\frac{\theta}{2}\right)\right), \qquad (5)$$

where K_1 is a normalization constant. The generating function g_{HF} defines an edge-like atom that is centered exactly on the North pole.

Then, in order to also efficiently represent the smooth areas in the 3D models, we propose to use a second generating function for the construction of the dictionary. The second function is built on a two-dimensional Gaussian function in $L^2(S^2)$:

$$g_{LF}(\theta,\varphi) = \exp\left(-\tan^2\left(\frac{\theta}{2}\right)\right). \qquad (6)$$

Eq. (6) represents an isotropic function, centered at the North Pole. The extension of the dictionary to contain atoms built on two generating functions actually improves the approximation rate, but does not increase the search complexity. In our implementation, the dictionary is indeed divided into two

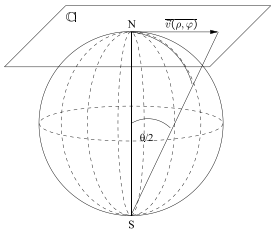


Fig. 1. Stereographic Projection

distinct parts, one with LF atoms (LF part) and the other of oscillating or high-frequency atoms (main part). We then successively use one sub-dictionary, and then the other, but not both at the same time, so that the search complexity is not augmented.

C. Motion on the sphere

Now that the generating functions have been defined, we form the redundant dictionary by applying geometrical transformations to these functions, on the 2d sphere. In other words, the dictionary is constructed by moving the generating functions on the sphere, by rotation of the functions around their axis, and by anisotropic scaling.

Motion and rotation belong to the group of affine transformations of the unit 2-sphere S^2. They are both realized by a single rotation $\varrho \in SO(3)$, where $SO(3)$ is the rotation group in $\mathbb{R}^3$. It is equivalent to apply the unitary operator Π_ϱ on the matrix of cartesian coordinates (x, y, z) of the unit sphere, denoted as P:

$$P_r = \Pi_\varrho P = R(\psi)U(\theta)R(\varphi)P, \qquad \varrho \in SO(3), \tag{7}$$

where $\{P\}_{3\times N}$ is the matrix of (x, y, z) coordinates of the non-transformed unit sphere, and $\{P_r\}_{3\times N}$ is the matrix of (x, y, z) coordinates of the transformed unit sphere. The generating function, as defined in Eq. (5) on the non-transformed unit sphere, is therefore transformed into a rotated atom, displaced on the particular point (θ,φ) on the sphere. The rotation matrix $R(\psi)$ is given by:

$$R(\psi) = \begin{pmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and is responsible for the rotation of the atom around its axis by a desired angle ψ. The matrices $U(\theta)$ and $R(\varphi)$ introduce motion of the atom over the sphere, by angles θ and φ. The matrix R is already defined above, and the matrix $U(\theta)$ is given by :

$$U(\theta) = \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix}$$

It is important to note that the order of these matrices is strictly defined. The atom first rotates on its axis, on the North pole, and then moves by (θ,φ) on the sphere.

D. Anisotropic refinement of atoms on the sphere

In order to well approximate the elongated characteristics of 3D models, we further deform atoms by anisotropic refinement that scales the generating function differently in each direction. We perform the scaling operation on the plane tangent to the North pole and then map the resulting atom on the sphere S^2, by inverse stereographic projection as described before. Let $\vec{v} = (x, y)$ denote a vector in the tangent plane; the anisotropic scaling operator is then expressed as:

$$D(a_1, a_2)g(\vec{v}) = Cg(a_1x, a_2y), \tag{8}$$

where the constant C is a normalization factor. The coordinates of the vector after scaling, $\vec{v_s}$, become :

$$\begin{aligned} x_s &= a_1 x = a_1 \rho \cos\varphi \\ y_s &= a_2 y = a_2 \rho \sin\varphi \,. \end{aligned} \tag{9}$$

In polar coordinates, it translates to :

$$\begin{aligned} \rho_s &= \sqrt{x_s^2 + y_s^2} = \rho\sqrt{a_1^2\cos^2\varphi + a_2^2\sin^2\varphi} \\ \varphi_s &= \arctan\frac{y_s}{x_s} = \arctan\frac{a_2 sin\varphi}{a_1 cos\varphi}. \end{aligned} \tag{10}$$

Anisotropic refinement of high frequency atoms, as given in Eq. (2), is obtained by substitution of the polar coordinates with the ones obtained after scaling. They can be written as :

$$\begin{aligned} g_{rect}(\rho, \varphi) = &-\frac{1}{K}\left(4a_1^2\rho^2\cos^2\varphi - 2\right) \\ &\cdot \exp\left(-\frac{\rho^2\left(a_1^2\cos^2\varphi + a_2^2\sin^2\varphi\right)}{4}\right). \end{aligned} \tag{11}$$

By inverse stereographic projection $\Phi^{-1} : \mathbb{R}^2 \rightarrow S^2$, the reshaped atom is mapped on the sphere, and can be written as :

$$\begin{aligned} g_{HF}(\theta, \varphi) = &-\frac{1}{K_2}\left(16a_1^2\tan^2\frac{\theta}{2}\cos^2\varphi - 2\right) \\ &\cdot \exp\left(-\tan^2\frac{\theta}{2}\left(a_1^2\cos^2\varphi + a_2^2\sin^2\varphi\right)\right), \end{aligned} \tag{12}$$

where K_2 is a normalization factor. On the other hand, the low-frequency atoms after anisotropic refinement, can be written as :

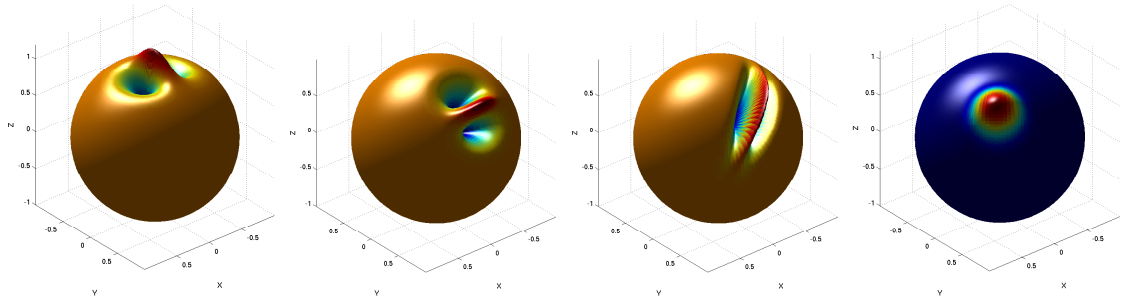
$$g_{LF}(\theta, \varphi) = \exp\left(-\tan^2\frac{\theta}{2}\left(a_1^2\cos^2\varphi + a_2^2\sin^2\varphi\right)\right). \tag{13}$$

In summary, the dictionary is obtained by anisotropic scaling, rotation and displacement of atoms, which is implemented by first building the generating function on a unit sphere previously transformed due to rotations. Then anisotropic scaling is applied on the displaced atom, as explained above. Such a process is performed for both oscillating atoms as expressed in Eq. (12)), and for low-frequency ones (see Eq. (13)). By a proper choice of the transformation parameters, one finally obtains a overcomplete dictionary of functions, that is used to represent simple 3D models. Sample atoms are illustrated in Figure 2.

IV. Matching Pursuit encoder for 3D objects

A. Matching Pursuit overview

Finding the sparsest representation of a simple 3D model, with functions taken from a redundant dictionary, as the one described before, is in general an NP-hard problem. Matching Pursuit [13] provides a solution to this approximation problem, with a tractable computational complexity. Under its generic form, Matching Pursuit (MP) is an algorithm that iteratively decomposes a signal into a linear combination of waveforms, or atoms. Interestingly, very few restrictions are imposed on the dictionary construction, besides the fact that it should at least span the space of the signal to represent. In other words,



(a) **(b)** **(c)** **(d)**

Fig. 2. Anisotropic atoms: a) on the North pole ($\theta = 0, \varphi = 0$), $\psi = 0, a_1 = 8, a_2 = 8$; b) $\theta = \frac{\pi}{4}, \varphi = \frac{\pi}{2}, \psi = 0, a_1 = 8, a_2 = 8$; c) $\theta = \frac{\pi}{4}, \varphi = \frac{\pi}{2}, \psi = \frac{\pi}{4}, a_1 = 16, a_2 = 4$; d) Low frequency atom: $\theta = \frac{\pi}{4}, \varphi = \frac{\pi}{4}, \psi = \frac{\pi}{4}, a_1 = 8, a_2 = 8$

the dictionary is defined as a set of vectors $D = (g_\gamma)_{\gamma \in \Gamma}$ in a Hilbert space H. In order to be able to represent each vector in H as a linear combination of unit norm vectors in D, the dictionary must satisfy the completeness property (i.e., $\overline{Span(D)} = H$).

Let $f \in H$ denote a function which we want to approximate with a linear expansion over D. With MP, an N-terms linear expansion is obtained by successive approximations of $R^n f$ through orthogonal projections on dictionary vectors:

$$f = \sum_{n=0}^{N} \langle R^n f, g_{\gamma_n} \rangle \, g_{\gamma_n} + R^{N+1} f, \tag{14}$$

where $R^n f$ is the residue after $n-1$ iterations of the algorithm ($R^0 f = f$). One must choose, at each iteration, the atom that best approximates $R^n f$, with the maximal projection $|\langle R^n f, g_{\gamma_n} \rangle|$ over the dictionary:

$$|\langle R^n f, g_{\gamma_n} \rangle| = sup_{\gamma \in \Gamma} |\langle R^n f, g_\gamma \rangle|. \tag{15}$$

When $N \to \infty$, under assumption that the dictionary is complete, it can be shown that:

$$f = \sum_{n=0}^{\infty} \langle R^n f, g_{\gamma_n} \rangle \, g_{\gamma_n}. \tag{16}$$

Interestingly, it has been shown that the residue decays exponentially in a finite dimensional space, assuming complete dictionary [13]. Under the same assumption, the signal can be exactly recovered after a possibly very large number of iterations, i.e., $R^{N+1} f \to 0$ when $N \to \infty$. The decay rate depends on the correlation between the residue and the dictionary elements, so that the construction of an efficient dictionary, adapted to the very structure of the signal f, is a crucial step.

Overall, MP offers a sub-optimal solution to the optimal (sparsest) signal representation problem, since it iteratively approximates the signal, in a totally greedy manner. However, it allows for an efficient approximation of the signal by rapidly capturing its most important components, which is an interesting property in the design of a scalable coder. In the same time, it does not impose any condition on the dictionary design, and the complexity at decoder is kept small.

B. 3D-SMP encoder

Our objective is to build on the nice approximation properties offered by redundant expansions to obtain compressed versions of 3D models. The block diagram of the proposed encoder, called 3D-SMP (3D model encoder based on Spherical Matching Pursuit), is represented in Figure 3. Matching Pursuit selects a series of atoms from the dictionary described above, with their relative coefficients. Atoms are then sorted along the decreasing magnitude of their coefficients. The coefficients are then uniformly quantized, with a decaying quantization range. This takes advantage of the property that the energy of Matching Pursuit coefficients is limited by an exponentially decaying upper-bound. A piecewise linear approximation of that upper-bound is used for quantization, inspired from the scheme proposed in [14]. Quantized coefficients, and discrete atom indexes are finally encoded with an arithmetic coder [15], in order to obtain a compact representation. Interested readers are referred to [11] for more details about quantization, and entropy coding of MP atoms.

Note that an initial resampling step may be required in order to map the input model on a 2D sphere, with a regular (θ, φ) grid. Since the proposed scheme requires the input 3D data to be a 2D set of radius values representing a function $f : S^2 \to \mathbb{R}$ defined on 2-sphere S^2 as $R_{int} = f(\theta, \varphi)$, a resampling step may need to be introduced, depending on the format of the input model. Starting from any common format 3D model, the initial block of the codec extracts a set of vertices $p_i = (x_i, y_i, z_i) \in I$, $I \subset \mathbb{R}^3$ which represent a point cloud of a 3D model. It then calculates the samples values on the sphere by performing an interpolation within neighboring radiuses from the point cloud. This method is valid only for simple genus-zero 3D models, but it can be extended to more complex models by splitting the model into a combination of spherical surfaces, as we will show in the Section V.

C. 3D-SMP decoder

On the other end, the decoder, as represented in Figure 4, first performs the entropy decoding and inverse quantization. It then reconstructs the approximated 3D model as a spherical function, by linear combination of atoms whose relative weights are given by the MP coefficients. The reconstruction

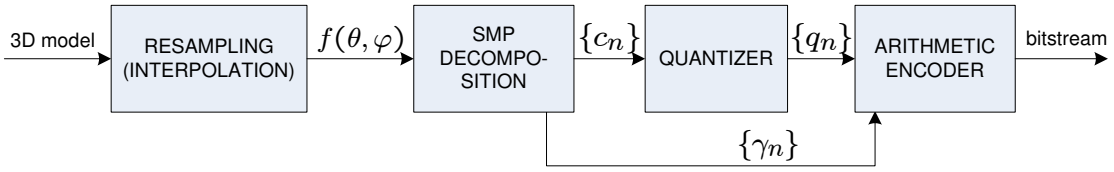


Fig. 3. 3D-SMP encoding scheme

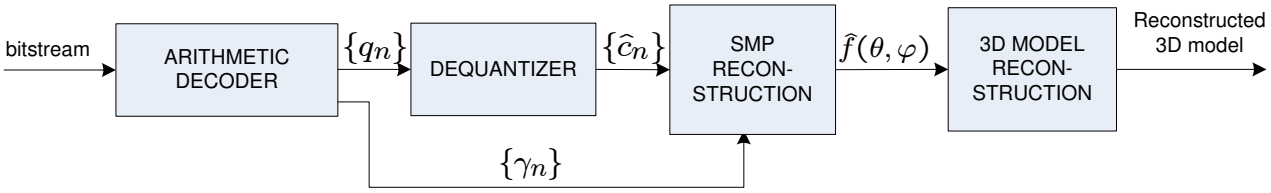


Fig. 4. 3D-SMP decoding scheme

step at the decoder side has a quite low computational complexity, roughly proportional to the number of atoms, and thus the coding rate.

The decoder can finally generate the decoded 3D object in the form of a standard polygonal mesh, as accepted by all modern computer graphics application and hardware. Since the encoder has completely discarded the mesh connectivity information of the original 3D model, the decoder has to generate new connectivity. This problem can be formulated as a surface reconstruction problem from an unorganized point cloud, which is still an active area of research, and many surface reconstruction algorithms already exist (for example [16]). Since we are primarily dealing with simple models parameterized as one spherical function, we can use the a-priori knowledge of the (θ, φ) coordinates for each vertex on the spherical grid and construct a *semi-regular* connectivity structure. A mesh with semi-regular connectivity has almost all vertices of valence 6 (i.e., six incident edges), except for a few isolated extra-ordinary vertices with valence$\neq$6. The connectivity matrix is defined with indexes of three incident vertices for each face. We target here the construction of a triangular mesh, since every polygonal mesh can be reduced to triangular. In order to obtain a semi-regular mesh, we can divide the spherical grid into rings limited with two successive values of θ, and then triangulate each ring to produce a triangular strip. Such a mesh construction is illustrated in Figure 5 (a), which shows the triangular subdivision of the sphere. The Figure 5 (b) represents the same grid, but applied to the Venus model. All vertices are of valence 6, except the two poles, so that the resulting mesh is indeed semi-regular.

For more complex models whose representation requires multiple spheres, the method explained above is not directly applicable, since the boundary between two neighboring spheres does not necessarily coincide with a great circle on the sphere. In these cases, a simpler solution would be to use a more generic surface reconstruction algorithm. The proposed 3D-SMP scheme uses the algorithm explained in [17] [3].

V. Experimental coding results

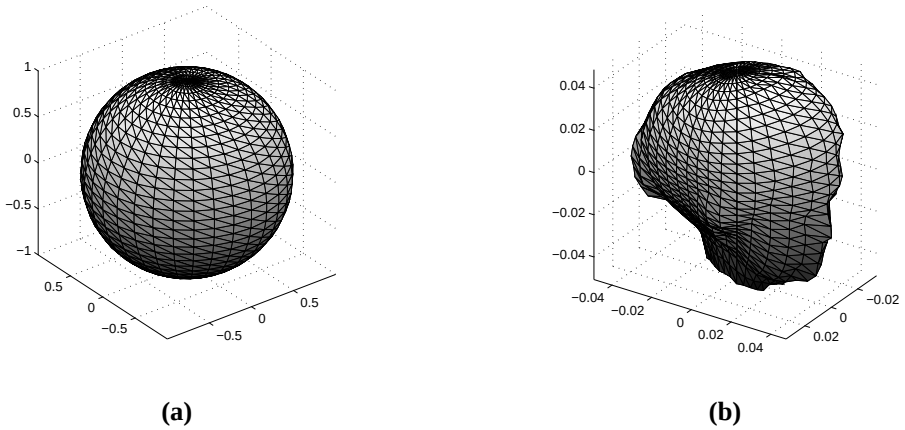
A. Preliminaries

As most of the models available are represented as meshes, or point clouds, we have implemented a simple method to map the input model on a regular spherical grid. The value of the signal at each point (θ, φ) on the spherical grid is obtained by performing a nearest neighbor interpolation between four points from the original point cloud, whose projections are maximal on the desired direction of the radius. Furthermore, we use a grid that is uniform in θ and φ, in order to be compliant with fast algorithms for spherical harmonic transforms that are used in the signal decomposition.

Two models are used in our experiments: Venus and Rabbit[4]. Venus satisfies the assumption of a simple genus-zero model, thus it is represented via one spherical function. Since Rabbit is not a simple model, we have chosen to decompose it into three spheres separated by two parallel planes, one below the head and the other below the arms of the Rabbit. Each spherical function is obtained by interpolation within the point cloud on the corresponding part of the model, while the other parts are appropriately smoothed in order to avoid wasting atoms. Afterwards, SMP is independently run on each of these three spheres and finally gathered into a single

[3]Reconstruction server is available at `http://cgal.inria.fr/Reconstruction/submit.html`.

[4]The models have been downloaded from `http://www.cyberware.com`.



(a) **(b)**

Fig. 5. Generating connectivity matrix: a) Sphere connectivity b) Connectivity on the Venus model.

decomposition by ordering the atoms in a decaying order of their coefficient values. One additional atom parameter is introduced to denote the sphere that the atom belongs to. The quantization and entropy coding steps are the same as for the one-sphere decompositions. Finally, three spherical functions are reconstructed at the decoder, and point clouds are merged using a surface reconstruction algorithm, as explained in IV-C. The original and interpolated models are shown in Figure 6. The interpolation error is expressed with relative L^2 error and in PSNR[dB], as computed with the MESH software [5]. The relative L^2 error is actually a ratio of RMS (Root Mean Square Error), which measures the squared symmetric distance between two surfaces averaged over the first surface, relative to a bounding box diagonal D. The PSNR (Peak Signal To Noise Ratio) for 3D meshes is thus expressed as:

$$PSNR[dB] = 20log\left(\frac{D}{RMS}\right) = 20log\left(\frac{1}{L^2}\right) \qquad (17)$$

B. Implementation

In the dictionary presented in Section III, the atom parameters obviously take discrete values. In general, a fine granularity in atom indexes leads to high redundancy, and likely to high approximation rate. In the same time, it leads to a large dictionary, and therefore high coding cost. The design of an optimal dictionary is still an open problem, beyond the scope of this paper. Here, we propose to use a dictionary, mostly built on empirical choices for atom parameter values. First we use the equiangular spherical grid to drive the values of the position parameters, θ and φ; both parameters are uniformly distributed on the interval $[0, \pi]$, and $[-\pi, \pi)$, respectively, with a resolution that is identical to the input signal. The rotation parameter ψ is uniformly sampled on the interval $[-\pi, \pi)$, with the same resolution as θ and φ. This choice is mostly due to the use of spherical convolution in the Matching Pursuit algorithm, as explained below. Finally, scaling parameters are distributed in a logarithmic manner, from 1 to half of the size of the input signal, with a granularity of one third of octave. The largest atom thus covers half of the sphere. For low pass atoms, the maximal scale is chosen to be $1/16$ of the signal size. Motions and rotations are discretized in the same way as for anisotropic atoms.

In our implementation, the full dictionary is divided into low-frequency atoms g_{LF}, and high-frequency ones. During the first iterations, the Matching Pursuit algorithm uses the low-frequency sub-dictionary, and later switches to the anisotropic sub-dictionary when the energy of the coefficients starts to saturate, or more precisely when :

$$\frac{|C_n|}{\|R_n\|_2} \rightarrow const, \qquad (18)$$

where C_n denotes a projection after $n-1$ iterations. In each of these sub-dictionaries, the Matching Pursuit algorithm performs a full search to determine the highest energy atom. Our implementation uses the Fast Spherical Transform [18] to compute the convolution of atoms on the sphere. In particular, we used in our implementation the *SpharmonicKit* library[6], which is part of the *YAW toolbox* [7]. Such a transform allows to identify the position and rotation of an atom on the sphere, which has the best correlation with the signal. One spherical convolution allows to determine the parameters (θ, φ, ψ) for each atom with given scale parameters. Therefore, our implementation iterates over the scale parameters: for each couple (a_1, a_2), it computes the spherical convolution between the corresponding atom on the North Pole, and the residual signal to code, $R^n f$. The indexes of the convolution coefficient with the largest magnitude correspond to the position and rotation parameters (θ, φ, ψ) of the best matching atom for that pair of scales. The coefficient of that best matching atom is then computed with the inner product of two functions defined on the sphere, given by :

$$\langle R^n f, g\rangle = \int_\theta \int_\varphi R^n f(\theta, \varphi) g(\theta, \varphi) sin\theta d\theta d\varphi, \qquad (19)$$

Finally, the algorithm selects, among all pairs of scales, the atom with largest coefficient, removes its contribution from the residual signal, and repeats the whole procedure until a

[5]MESH is available at `http://mesh.epfl.ch`.

[6]`http://www.cs.dartmouth.edu/~geelong/sphere/`

[7]`http://fyma.fyma.ucl.ac.be/projects/yawtb/`

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

Fig. 6. (a) Original Venus model; (b) Interpolated Venus model at resolution $N_\theta \times N_\varphi = 128 \times 128$, PSNR=65.7983dB, $L^2 = 5.1296 \cdot 10^{-4}$; (c) Interpolated Venus model at resolution $N_\theta \times N_\varphi = 256 \times 256$, PSNR=70.1930dB, $L^2 = 3.09278 \cdot 10^{-4}$; d) Original Rabbit model; e) Interpolated Rabbit model on three spheres at resolution $N_\theta \times N_\varphi = 128 \times 128$, PSNR=64.1790dB, $L^2 = 6.18091 \cdot 10^{-4}$.

stopping criteria is met (e.g., a pre-defined number of atoms, or an energy threshold). The search algorithm is described in Algorithm 1.

Algorithm 1 Full search of the dictionary

for all scale couples $(a_1(j), a_2(k))$ **do**
$C = \text{convsph}(R^n f, g(0,0,0,a_1(j), a_2(k)))$
$C_{max} = \max_{1<m<N_\theta, 1<n<N_\varphi, 1<p<N_\psi} C(m,n,p)$
$\theta = \theta(m); \varphi = \varphi(n); \psi = \psi(p)$
$P(j,k) = \langle R^n f, g(\theta, \varphi, \psi, a_1(j), a_2(k)) \rangle$
end for
$P_{max} = \max_{j,k}(P(j,k))$
$a_1 = a_1(j); a_2 = a_2(k)$

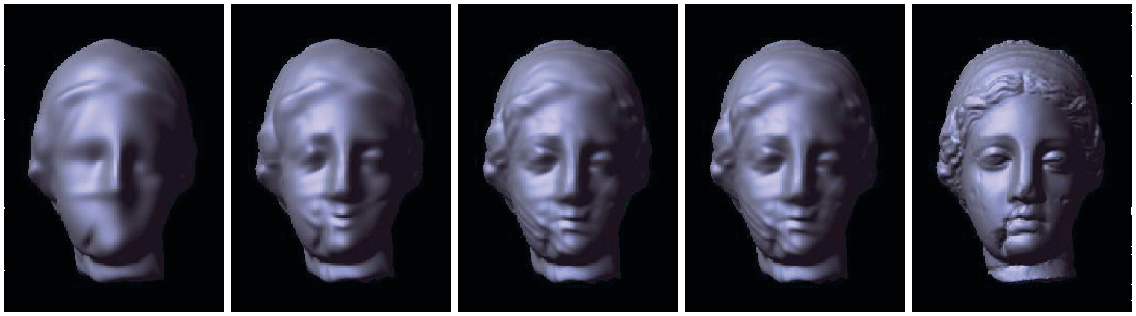
C. Numerical results

Venus and Rabbit models, as reconstructed by the decoder, are represented on Figure 7 and Figure 8, respectively, for different numbers of atoms. It can be seen that Matching Pursuit rapidly captures the most important features of the 3D model and progressively refines the representation with finer details. The type of coding artifacts is quite different than the degradations observed in mesh-based coders, and visually less annoying at low rate. It can be seen also that the gain in representation accuracy is less important when the number of iterations increases; as expected Matching Pursuit is mostly efficient for low bit rate representation of 3D models.

Figures 9 and 10 present the rate-distortion performance of the proposed 3D-SMP algorithm for Venus and Rabbit models, in terms of both PSNR (a) and L^2 error (b)). They compare the 3D-SMP encoder performance with the following state-of-the-art encoders: (i) TG: Tauma-Gotsman non-progressive coding [19], (ii) Alliez-Desbrun progressive coding [20], and (iii) PGC: Progressive coding scheme by Khodakovsky et al. [8]. Due to the differences in input formats, and coding approaches, we use the following approach to obtain fair performance comparisons between these four very different approaches. The input models for TG and Alliez-Desbrun methods are the models that have been used to obtain the interpolated version for 3D-SMP encoder[8], but decimated to 1400 faces (using the Qslim software[9]), in order to have a comparison in the same rate region. Different rates for the TG algorithm are obtained by changing the number of bits per vertex for encoding. As PGC uses its own mesh format, the input models are downloaded from the PGC website[10]. The base mesh for the PGC is encoded using TG with 8 bit per vertice. Note that the rate is actually given by the filesize, or the total number of bits, rather than in bits per vertex, since the proposed 3D-SMP coding scheme uses one single mesh (256x256 or 128x128 vertices in the current implementation) throughout progressive compression.

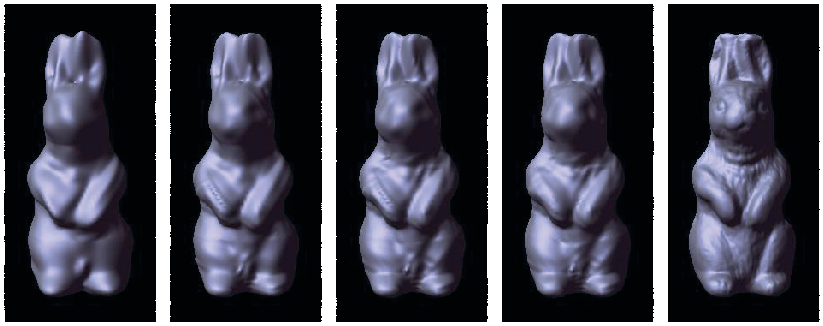
It can be seen that 3D-SMP significantly outperforms the state-of-the-art compression methods TG and Alliez-Desbrun, as well as the PGC wavelet-based coder at low bit rate. MP then tends to saturate towards high bite rate, as observed earlier. For the Venus model the performance is slightly better than for the Rabbit model. This behavior is actually expected since the resolution of the employed SMP for Rabbit is smaller. Therefore, we can certainly expect better performance for the Rabbit model at higher resolutions. It has to be noted also that the input model for the SMP is an interpolated version of the original model, and that this also introduces a distortion, that is independent of the coding method. Figure 11 shows a visual comparison of Venus encoded with 3D-SMP using 250 coefficients and resolution 256x256, and encoded with the PGC algorithm, for the same filesize of 1287B. It can be seen that both coders offer similar performance, but the coding artifacts are quite different. The 3D-SMP coder generally provides a smoother approximation of the model, but fails in capturing the highly textured regions like the hair, for example. The proposed encoder offer an interesting alternatives to classical approaches, with excellent compression performance at low bit rate, whilst it still brings an inherent progressive representation. Additionally, it offers a great flexibility in the stream construction, that can be advantageously exploited in adaptive applications, like view-dependent rendering as

[8] http://www.cyberware.com/samples/index.html
[9] http://graphics.cs.uiuc.edu/~garland/software/qslim.html
[10] http://www.multires.caltech.edu/software/pgc/



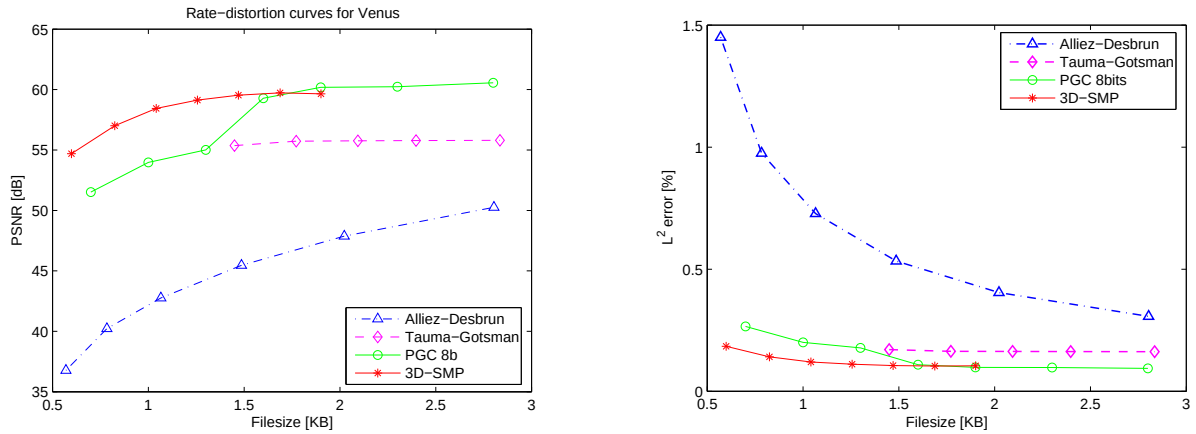
(a) **(b)** **(c)** **(d)** **(e)**

Fig. 7. Venus reconstructed after decoding (resolution 256x256): a) 100 coefficients; b) 200 coefficients; c) 300 coefficients; d) 400 coefficients; e) Input model.



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

Fig. 8. Rabbit reconstructed after decoding (three spheres, resolution 128x128): a) 100 coefficients; b) 200 coefficients; c) 300 coefficients; d) 400 coefficients; e) Input model.



(a) **(b)**

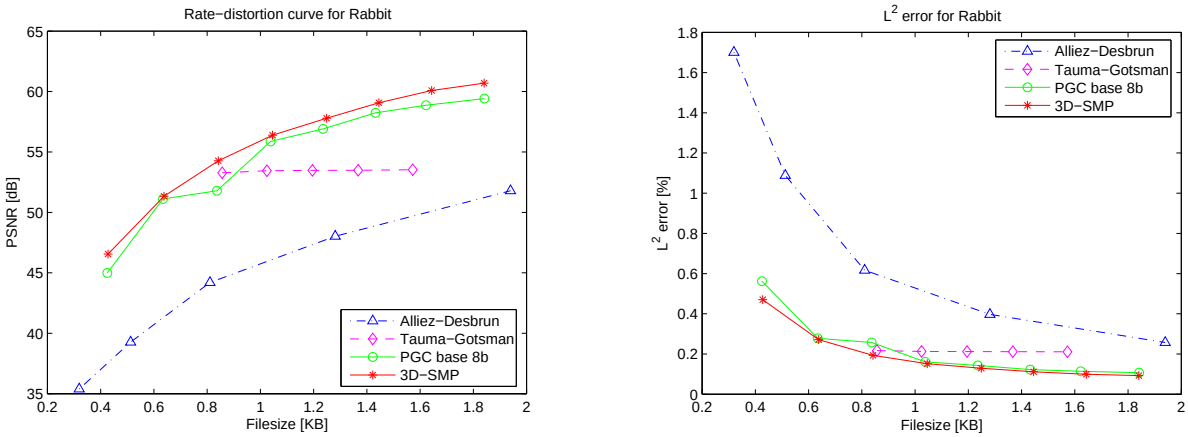
Fig. 9. Rate-distortion performance for the Venus model (resolution 256×256) (a) PSNR, (b) L^2 error.

presented in the next section.

VI. View-dependent rendering

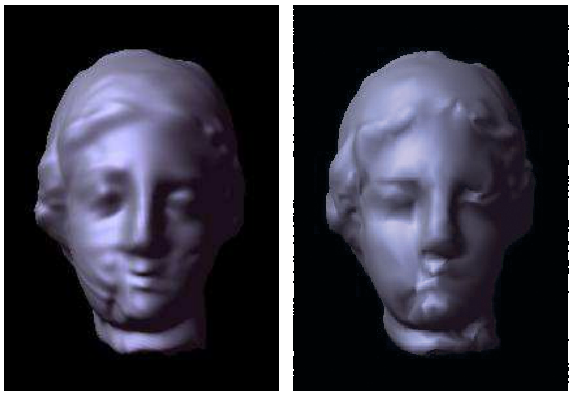
This section proposes a simple application that uses the increased flexibility offered by the use of a structured dictionary of functions, in addition to the progressive nature of the stream generated by the Matching Pursuit decomposition. In scenarios where very high compression is necessary, or when the transmission channel represents an important bottleneck, it becomes interesting to use view-dependent rendering algorithms. In such cases, the 3D model does not need to be completely decoded before being displayed, but the rendering can be made dependent on the viewpoint. The server can transmit in priority the parts of the model that are visible, while cutting off the invisible parts. View dependent progressive transmission and rendering can therefore significantly improve the performance of 3D graphics streaming applications.

All the previous work done in this area uses classical 3D



(a) **(b)**

Fig. 10. Rate-distortion performance for the Rabbit model (resolution 128×128) (a) PSNR, (b) L^2 error.



(a) **(b)**

Fig. 11. Venus with a filesize of 1287B: a) MP; b) PGC.

multi-resolution coding techniques based on mesh simplification algorithms by vertex split operations. For example, Yang et al. [21] split the 3D model into progressively encoded partitions which are reorganized so that the visible ones are transmitted with higher priority. Zach et al. [22] propose a view-dependent mesh connectivity encoding scheme which reorganizes the vertex tree, while Wang and Li [23] employ an absolute path coding of vertices. Recently, Bischoff and Rossignac introduced the *TetStreamer* [24] which performs a view-dependent front-to-back (in the visibility order) encoding of 3D models. However, we note that none of the mesh geometry compression schemes based on wavelets ([7], [9], [8]) does address this issue, even if some solutions inspired from Region-of-Interest coding in images could be envisaged.

The use of a structured dictionary in the proposed Matching Pursuit encoder presents a great advantage in regards to manipulation of the compressed stream. In particular, it allows for a quite simple extension towards a view-dependent transmission and rendering algorithm. A 3D model is decomposed into a linear combination of atoms specified by their position and scale parameters: it can be partially transmitted by selecting the atoms visible only from any given viewpoint. Assume that the viewpoint can be associated with a window function defined on the unit sphere (i.e., $w(\theta, \varphi) \in L^2(S^2)$). Such a window function on the sphere is obtained by an inverse stereographic projection of a circle on the tangent plane, with the center at the North Pole and the radius ρ. The window function can be expressed as :

$$w(\theta, \varphi) = \begin{cases} 1 & \text{if } 2tan\left(\frac{\theta}{2}\right) \leq \rho, -\pi \leq \varphi \leq \pi; \\ 0 & \text{otherwise.} \end{cases}$$

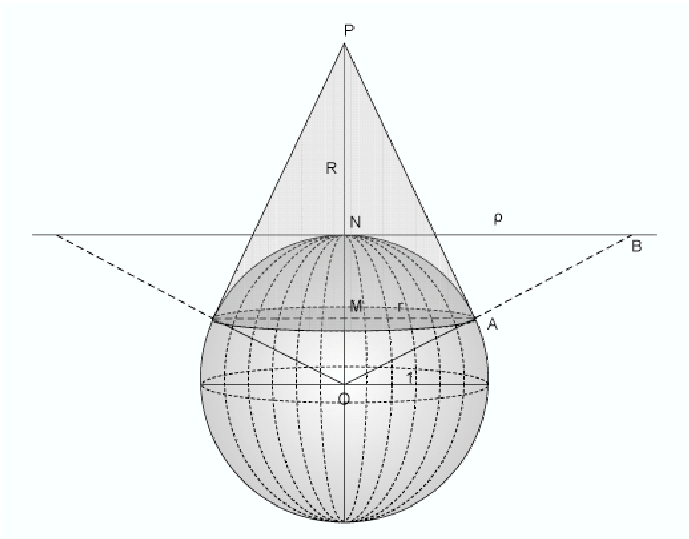


Fig. 12. The intersection of the unit sphere and the conical surface originating from the viewpoint P.

This is illustrated in Figure(12), where the tangent plane is shown as a line passing through the North pole N. The part of the sphere that is visible from the viewpoint is delimited by a conical surface formed by the union of all the straight lines that pass through the viewpoint, and are tangent to the unit sphere. This part is shown on the Figure(12) as the darker shaded part of the sphere. Let r denote the radius of the circle obtained

by intersecting the cone and the sphere ($r = \overline{MA}$). From the similarity of triangles $\triangle OPA$ and $\triangle OAM$, it follows:

$$r = \frac{\sqrt{R_p^2 - 1}}{R_p}, \tag{20}$$

where $R_p = \overline{OP}$. From the similarity of triangles $\triangle OAM$ and $\triangle OBN$, we obtain :

$$\rho = \frac{r}{\sqrt{1 - r^2}} = \sqrt{R_p^2 - 1} \tag{21}$$

The radius ρ is therefore directly determined by the distance between the viewpoint, and the 3D model.

To generate a view-dependent reconstruction of the 3D model, each atom $g_{\gamma_i}, i = 1, ..., N$ in the signal decomposition is simply multiplied with the window function. It results in a windowed atom gw_{γ_i}, which represents that part of the atom, visible from the viewpoint P :

$$gw_{\gamma_i}(\theta, \varphi) = g_{\gamma_i}(\theta, \varphi) \cdot w(\theta, \varphi), \qquad i = 1, ..., N. \tag{22}$$

Only the atoms that keep a significant contribution after windowing are finally considered for view-dependent transmission and rendering. The selection of atoms is based on the comparison of their maximal value, with a predefined relative threshold δ (%). If $c_\gamma \cdot \ gw_{\gamma_i}(\theta, \varphi) > \delta$ for any pair (θ, φ), the atom g_{γ_i} is transmitted.

It has to be noted that the uniform window proposed above obviously attributes the same importance to all the atoms that are visible from the viewpoint P. Other windows are however straightforward to implement. For example, if one wants to emphasize the importance of atoms closer to the view direction ($\overline{OP}$), a Gaussian window could be used, as given by :

$$w_g(\theta, \varphi) = \begin{cases} exp\left(\frac{-4tan^2\frac{\theta}{2}}{2\sigma^2}\right) & \text{if } 2tan\left(\frac{\theta}{2}\right) \leq \rho, \\ & -\pi \leq \varphi \leq \pi; \\ 0 & \text{otherwise.} \end{cases}$$

The Figure 13 presents the results of view-dependent coding and transmission, applied to the Venus model. The threshold δ is set to 1% of the non-windowed atom peak value, and the variance of the Gaussian window function is set to $\sigma = r$. It can be seen that emphasis is correctly put on the part of the model that is visible, as expected. Furthermore, the view-dependent transmission of atoms allows for a significant decrease in required bit rate, since only 42.8% of the model has been used.

By increasing σ, more precise view dependent encoded models are obtained as more atoms are transmitted. This of course implies that the bit rate is also increased. Change of σ can be therefore used for the coarse tuning of the rate-distortion trade-off. For a finer adaptation, the threshold value δ can be changed. As the viewpoint may change during the interactive 3D model rendering, the rest of the atoms can be progressively transmitted in lower priority, until the complete scene is received. Finally, the view-dependent rendering by atom windowing is a simple algorithm that allows to prioritize atoms from the Matching Pursuit decomposition. Obviously, more sophisticated methods could be proposed with application-specific requirements, yet taking advantage of the structured nature of the redundant dictionary used in the proposed Matching Pursuit encoder.

VII. Conclusions and Future work

This paper has proposed a novel approach for the coding of 3D objects. The models are mapped on a 2D sphere and decomposed over a redundant dictionary of multi-dimensional atoms, which is built to efficiently capture the most prominent features of the signal. An encoding algorithm based on Matching Pursuit has been used to generate progressive representations of the 3D objects. The proposed encoder has been shown to outperform state-of-the-art progressive coders, especially at low bit rate. In the same time, it offers a truly progressive representation, with increased flexibility in the stream manipulation. The structured nature of the dictionary has been advantageously used in the design of a simple view-dependent rendering application, which may prove to be useful in service with important transmission bottlenecks. The proposed algorithm still leaves numerous possibilities for future improvements. The interpolation step, which uses the very basic nearest neighbor method, has an important influence that penalizes the compression performance. We are also currently working on better parametrization of input models, and on generalizing the multiple spheres decomposition approach for the coding of more complex 3D models. Finally, since it has been shown that redundant decompositions are mostly beneficial at low rate, the proposed scheme can offer a an efficient coding solution for base layer in scalable applications. Similarly to hybrid image coding schemes [25], enhancement layers based on spherical wavelets, for example, could nicely complement the proposed scheme for high bit rate coding.

VIII. Acknowledgments

The authors would like to thank Dr. Nicolas Aspert for his valuable help in implementation of the 3D-SMP, and Dr. Rosa Maria Figueras i Ventura for helpful discussions on the Matching Pursuit coding scheme.

References

[1] M. Deering, "Geometric compression," *Siggraph '95 Conference Proceedings*, pp. 13–20, August 1995.
[2] Silicon Graphics Inc., "GL programming guide," 1991.
[3] J. Neider, T. Davis and M. Woo, "OpenGL programming guide," *Addison-Wesley, MA, USA*, 1997.
[4] G. Taubin and J. Rossignac, "Geometry compression through topological surgery," *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 84–115, April 1998.
[5] H. Hoppe, "Progressive meshes," *Siggraph '96 Conference Proceedings*, pp. 99–108, August 1996.
[6] G. Taubin, A. Guéziec and W. Horn, "Progressive forest split compression," *Siggraph '98 Conference Proceedings*, pp. 123–132, July 1998.
[7] P. Schröder and W. Sweldens, "Spherical wavelets: Efficiently representing functions on the sphere," *Siggraph '95 Conference Proceedings*, pp. 161–172, August 1995.
[8] A. Khodakovsky, P. Schröder and W. Sweldens, "Progressive geometry compression," *Siggraph '00 Conference Proceedings*, pp. 271–278, July 2000.
[9] H. Hoppe and E. Praun, "Shape compression using spherical geometry images," *Symposium on Multiresolution in Geometric Modeling. Cambridge*, September 2003.

[10] P. Vandergheynst and P. Frossard, "Efficient image representation by anisotropic refinement in matching pursuit," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Salt Lake City*, vol. 3, pp. 1757–1760, May 2001.
[11] R. Figueras i Ventura, P. Vandergheynst, and P. Frossard, "Low rate and flexible image coding with redundant representations." *IEEE Transactions on Image Processing*, February 2005.
[12] J. Antoine and P. Vandergheynst, "Wavelets on the 2-sphere : a group theoretical approach," *Applied and Computational Harmonic Analysis*, vol. 7, pp. 1–30, November 1999.
[13] S.G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, December 1993.
[14] P. Frossard, P. Vandergheynst, R.M. Figueras i Ventura and M. Kunt, "A posteriori quantization of progressive matching pursuit streams," *IEEE Transactions on Signal Processing*, vol. 52, no. 2, pp. 525–535, February 2004.
[15] I.H. Witten, R.M. Neal and J.G. Cleary, "Artihmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.
[16] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Surface reconstruction from unorganized points," *Siggraph '92 Conference Proceedings*, June 1992.
[17] D. Cohen-Steiner, F. Da, "A Greedy Delaunay Based Surface Reconstruction Algorithm," INRIA, ECG Technical Report ECG-TR-124202-01, December 2001.
[18] J.R. Driscoll and D.M. Healy, Jr., "Asympthotically fast algorithms for spherical and related transforms," *30th Annual Symposium on Foundations of Computer Science*, pp. 344 – 349, October 1989.
[19] C. Touma and C. Gotsman, "Triangle mesh compression," *Graphics Interface Conference Proceedings, Vancouver*, June 1998.
[20] P. Alliez and M. Desbrun, "Progressive compression for lossless transmission of triangle meshes," *Siggraph '01 Conference Proceedings*, pp. 195–202, August 2001.
[21] S. Yang, C.-S. Kim, C.-C. Jay Kuo, "View-dependent progressive mesh coding for graphic streaming," *Proceedings of SPIE ITCOM*, August 2001.
[22] C. Zach, M. Grabner, K. Karner, "Improved compression of topology for view-dependent rendering," *Technical Report, VRVis*, 2004.
[23] H. Wang, J. Li, "Octmesh - interactive mesh browsing over the internet," *The International Conference on Information Technology: Coding and Computing (ITCC'00)*, March 2000.
[24] U. Bischoff and J. Rossignac, "TetStreamer: Compressed Back-to-Front Transmission of Delaunay Tetrahedra Meshes," in *Data Compression Conference*, March 2005, pp. 93–102.
[25] L. Peotta, L. Granai and P. Vandergheynst, "Image compression using an edge adapted redundant dictionary and wavelets," *Eurasip Journal on Signal Processing , special issue on Sparse Approximations in Signal and Image Processing.*, 2005, to appear.

(a) **(b)** **(c)** **(d)**

Fig. 13. View dependent coding of Venus, $\sigma = r$; $\delta = 1\%$; 42.8% of the model is transmitted (214 out of 500 atoms): a) visible part of the decoded model; b) difference between the original model and the decoded model for the visible part; c) invisible part of the decoded model; d) difference between the original model and the decoded model for the invisible part.