Sparse Approximations of 3D Mesh Geometry Using Frames as Overcomplete Dictionaries

Maja Krivokuća
The University of Auckland
mkri012@aucklanduni.ac.nz

Waleed H. Abdulla
The University of Auckland
w.abdulla@auckland.ac.nz

Burkhard C. Wünsche
The University of Auckland
burkhard@cs.auckland.ac.nz

Abstract

This paper presents a novel method for creating a frame, to be used as an overcomplete dictionary for the progressive compression of 3D mesh geometry. The frame is computed from redundant linear combinations of the eigenvectors of a mesh Laplacian matrix, and atoms are selected by a Matching Pursuit algorithm. Experimental results show that a sparser representation of a given mesh geometry can be obtained with the frame than by decomposition of the mesh geometry onto an orthogonal basis. The proposed frame also has other desirable properties, including directionality and orientability of the atoms, and the ability to be applied directly to a manifold mesh with arbitrary topology and connectivity type.

1. Introduction

The raw data size of many 3D models created today for applications such as engineering design and manufacturing, architectural walkthroughs, biometrics, cultural heritage, entertainment, and e-commerce, often exceeds what can be stored or downloaded in a reasonable time by the users of these models. This has created an essential need for efficient algorithms for 3D model compression. Research on 3D model compression mainly started to take off in the mid- to late-90s, and has been heavily focused on the compression of triangular meshes as this is the most common representation for 3D objects. Traditionally, the research on mesh compression was on *single-rate* compression methods, where the entire geometry and connectivity are compressed, decompressed and rendered as a whole. But with the rise in popularity of the Internet and the increasing needs of consumers to share and download large 3D models over low-bandwidth network links, the idea of *progressive* compression started to emerge. Progressive compression allows a mesh to be transmitted and reconstructed progressively, from coarse to fine levels of detail, so that intermediate states of the model can be rendered while the data is still being received.

The focus of early progressive compression techniques [1-3] was the compression of mesh connectivity information. However, more recent algorithms [4-8] have striven to give priority to the compact encoding of the geometry data, because this often makes up a larger portion of the total mesh size than the connectivity data. Amongst these latter algorithms, the ones that seem to offer the best compression performance are those based on spectral analysis techniques borrowed from signal processing, namely the Fourier Transform [5] and the Wavelet Transform [6-8]. The fundamental assumption behind such techniques is that the input data can be represented as some sparse linear combination of *atoms* taken from a representative dictionary of atoms that constitute the new representation domain. Traditionally, these dictionaries have been chosen to be orthogonal bases; however, the recent explosion of research into alternative signal representations has shown that *overcomplete* dictionaries, or *frames* [9], where the representative vectors are not necessarily linearly independent, may be able to offer even sparser representations than orthogonal bases can.

In the field of 3D mesh compression, the idea of redundant representations has only recently started to emerge [10, 11]. The main difficulty in implementing a redundant representation system is currently the creation of a good overcomplete dictionary. In this paper we attempt to move this research a step forward, by offering a new method for constructing a frame to use for 3D mesh geometry compression. The results show that our frame indeed produces a sparser approximation of a given mesh geometry than the decomposition of the geometry on an orthogonal basis.

Section 2 of this paper covers background information on sparse approximations from redundant dictionaries, and reviews previous work in this area relevant to this paper; Section 3 presents our proposed frame; Section 4 explains how we use the proposed frame to achieve progressive mesh geometry compression; Section 5 details the experimental procedure that we followed to evaluate our frame in the context of mesh geometry compression; Section 6 presents some key results of our investigation; Section 7 examines the benefits of the proposed frame as well as the areas that still need further work; and Section 8 concludes the paper.

2. Sparse approximations from redundant dictionaries

The goal of sparse approximations is to express a given signal of dimension n (belonging to an n-dimensional vector space $\mathbb{V}^n$) as a linear combination of a small number of vectors taken from a *dictionary* that is representative of that class of input signals. The elements of the dictionary are typically unit-norm vectors called *atoms* [12]. The dictionary atoms may form a *basis*, or a *frame*, or neither, depending on their properties.

The dictionary forms a *basis* if: (a) its atoms are all *linearly independent* (no atom in the dictionary can be represented by a linear combination of any of the other dictionary atoms), (b) the set of atoms is *complete* (the atoms in the dictionary can be used to represent *any* other vector that belongs to the same vector space), and (c) any other vector in the same vector space can be represented by a *unique* linear combination of the basis vectors. In this sense, basis representations are non-redundant.

The redundant counterpart of a basis is called a *frame* [9]. Similarly to a basis, a frame is a representative set of vectors that span a given vector space. The difference is that the vectors in a frame do not necessarily need to be linearly independent. Thus for the same n-dimensional vector space $\mathbb{V}^n$, the frame may contain m vectors, where $m > n$. This makes a frame an *overcomplete* set of representative vectors for a given vector space, because some of the vectors in this set are *redundant*.

Formally, a family of vectors $\Phi = \{\varphi_i\}_{i\epsilon I}$ (where $I = \{1, \dots, m\}$) in a Hilbert space $\mathbb{H}$ of dimension n (where $m > n$) is said to be a *frame* for that vector space if there exist two constants $0 < A \leq B < \infty$, such that for all x in $\mathbb{H}$:

$$A\|x\|^2 \leq \sum_{i\epsilon I} |\langle x, \varphi_i \rangle|^2 \leq B\|x\|^2 \quad (1)$$

Here A and B are called the *frame bounds*. If $A = B$, the frame becomes a *tight frame* and equation (1) becomes:

$$\sum_{i\epsilon I} |\langle x, \varphi_i \rangle|^2 = A\|x\|^2 \quad (2)$$

If all the frame vectors also have a norm of 1, the frame is called a *unit-norm tight frame* and the constant A gives the redundancy ratio. Tight frames are convenient to use, because a tight frame and its dual are the same. This means that the same frame can be used for analysis and synthesis of the input signal because $\Phi\Phi^T x = x$ for all x, thus no inversion of the frame matrix is necessary. However, for more general frames, the frame and its dual are usually not the same, which means that a pseudo-inversion of the frame matrix is necessary. The solution for the pseudo-inverse is not unique and can be very time-consuming for a high-dimensional frame matrix.

The two main challenges of working with frames are: (i) the creation of a good frame, which can represent the input signal in a compact manner, and (ii) the method by which the "best" atoms are selected from this frame, in order to obtain the sparsest possible representation. The following sub-sections review some of the most important work to date in these two areas, which is relevant to the work in this paper.

2.1. Survey on existing overcomplete dictionaries

The variety of dictionaries proposed in the literature for various applications so far have been created by using one of three methods [13]: (i) analytic dictionaries, created from a mathematical model of the data; (ii) dictionaries learned from a training set of the data; or (iii) a combination of the methods in (i) and (ii). The work in this paper mainly fits under the umbrella of analytic dictionaries, and so the following literature review will be focused on that class of dictionaries only.

The overcomplete analytic dictionaries proposed to date have mainly been extensions of the well-known Wavelet Transform and have typically been formulated as *tight frames*. The difference between the different frames is mainly in the "shape" of their atoms and their localization properties. In [14], Candès and Donoho introduced tight frames made of *curvelets*, whose atoms are obtained by anisotropic dilations, rotations, and translations of a collection of unit-scale "oscillatory blobs" [14], which are localized in position, scale and orientation. This was extended to a 3D version in [15], where the curvelet atoms resemble flattened ellipsoids, and an alternative to the 2D curvelets was proposed in [16] as *contourlets*. The contourlet atoms look similar to the curvelets, but are easier to compute and have been extended to a multi-dimensional version called *surfacelets* [17]. In [18], Le Pennec and Mallat introduced *bandelets* for sparse image representations, with a second version by Peyré and Mallat [10] being introduced soon after. Bandelets on an image are edge-like elements that are achieved by a 2D wavelet transform followed by a *bandeletization*. Other analytic dictionaries that have been proposed, which follow different approaches to those described above, include mainly: the idea of merging two or more dictionaries to create *mega-dictionaries* [19]; the use of *wavelet packets* [20]; and forming unions of orthonormal bases [21].

2.2. Survey on methods for selecting atoms from overcomplete dictionaries

Selecting the "best" set of atoms from an overcomplete dictionary in order to obtain the sparsest possible approximation of the input signal is, in general, an NP-

hard problem. However, some good sub-optimal solutions have been proposed to date, which are solvable in polynomial time. Amongst these solutions, the most widely-used include: *Matching Pursuit* (MP) [22], which iteratively decomposes a signal into a linear combination of dictionary atoms by greedily selecting the best-matching atom at each iteration; *Basis Pursuit* (BP) [19], which starts with a representation of a signal in a basis and iteratively "improves" the basis by swapping out relatively insignificant atoms with new, more useful atoms; and *Orthogonal Matching Pursuit* (OMP) [23], which is similar to MP except that OMP updates *all* the coefficients extracted after each iteration by computing the orthogonal projection of the signal onto the set of atoms selected so far. These algorithms have been designed for *general* overcomplete dictionaries, making them widely applicable.

2.3. Survey on the use of redundant representations for 3D mesh compression

In the context of compression of 3D mesh data, the idea of using redundant representations still appears to be very much in its infancy. Indeed, we have only been able to find two papers [10, 11] that directly deal with this issue. In [10], the authors apply their *bandelet* frames to the compression of *spherical geometry images* [7] and normal maps of 3D mesh models. In [11], the authors resample the input mesh onto a regular spherical grid, and decompose the resampled model onto a redundant dictionary of oriented and anisotropic atoms living on the sphere. The redundant dictionary in [11] is created by applying translations, rotations and anisotropic scaling on two Gaussian generating functions on the sphere.

While the redundant representations in [10] and [11] demonstrate promising compression results over some existing, non-redundant techniques, there still exist avenues for improvement, mainly: (i) both of these algorithms are limited to models that can be mapped well on the sphere; (ii) both of these methods change the input mesh connectivity, so they cannot recover the original connectivity at the end; and (iii) the redundant dictionaries presented in these papers have only been tested on a very few select models, with certain properties, and so the design of an optimal frame for sparse approximations of 3D mesh geometry still remains an open problem.

To this end, in this paper we present a new idea for the creation of a frame, for the purpose of achieving a sparse approximation of 3D mesh geometry. Our objective here is not to demonstrate the creation of an *optimal* dictionary (indeed, that is an area that still needs much further work), but simply to (i) propose a new way of creating an overcomplete dictionary that can be used for 3D mesh geometry compression, and (ii) thereby confirm that redundant representations do indeed have promising capabilities for sparse approximations of 3D mesh data and should be researched further. The details of the proposed frame are presented in the following section.

3. The proposed algorithm for frame creation

Our frame is created from redundant linear combinations of the eigenvectors of the mesh Laplacian matrix, and a sparse approximation of the input mesh geometry is then obtained by using a Matching Pursuit algorithm. The procedure is outlined in the following sub-sections.

3.1. Basis creation

Firstly, the basis for a given input mesh is computed as the set of eigenvectors of the combinatorial Laplacian matrix of the input mesh, similarly to the *spectral decomposition* algorithm of Karni and Gotsman [5]. The Laplacian $\mathbf{L} \coloneqq (\ell_{i,j})_{n\,x\,n}$ is computed from the mesh connectivity as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{3}$$

where $\mathbf{A}$ is the $n \times n$ adjacency matrix of the input connectivity and $\mathbf{D}$ is the $n \times n$ matrix of vertex degrees (number of edges incident to each vertex), so that:

$$(\ell_{i,j}) \coloneqq \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $deg(v_i)$ is the degree of the vertex i. The resulting eigenvectors of $\mathbf{L}$ are orthogonal to each other and they are also normalized to have a length (*norm*) of 1, which makes this set an orthonormal basis for the input mesh. The eigenvectors are ordered in ascending order of their corresponding eigenvalues, so that the eigenvectors at the lower end effectively represent the low-frequency basis vectors and the ones at the higher end represent the high-frequency basis vectors.

We chose to use the eigenvectors of the mesh Laplacian for several reasons: (i) The eigenvectors represent the curvature directions of all the independent curvatures ("frequencies") making up the input mesh. This means that they inherently have *directionality* and *orientability*, which are desirable properties in dictionary design [13]; (ii) The eigenvectors have a meaningful physical interpretation, which relates to the structure of the input model: they essentially represent the directions in which the mesh has been "stretched" to give it the shape it currently has, and the eigenvalues represent the *amount* of "stretch" in the associated eigenvector direction; (iii) The computation of the eigenvectors is independent of the input mesh geometry, which is useful when we wish to use

this dictionary for mesh geometry compression; (iv) The eigenvectors can be computed for a manifold mesh with arbitrary connectivity and topology type; (v) It has already been shown in [5] that decomposition of mesh geometry onto the eigenvector basis produces a sparse set of coefficients. We wished to investigate if it was possible to improve this sparsity by extending the basis to a frame.

3.2. Frame creation

We create redundant dictionary vectors by computing different linear combinations of the eigenvectors in the basis described in section 3.1. We do this by selecting a certain number of eigenvectors from the ordered eigenvector set (e.g., 2 eigenvectors in the simplest case) and computing every possible linear combination of these eigenvectors, using all the scalar multiplying values in the range [-0.9, 0.9] with a step size of 0.1 between successive values. This range of scalars effectively represents all the possible coefficients that can be used in a linear combination of the vectors, when they have been quantized by rounding to a 1 decimal point precision. We normalize each created redundant vector to have a norm of 1 and store the normalized vector in the dictionary. Each new group of eigenvectors is selected by shifting only 1 place in the dictionary, so each group has an overlap with at least one of the eigenvectors of the previous group, e.g., when using *pairs* of eigenvectors, the second pair will consist of the second eigenvector used in the first pair and the next eigenvector in the list. We continue this process until all the eigenvectors have been considered. In the end, we have a frame that consists of the original eigenvectors of the input model and also the redundant vectors that have been created as linear combinations of these eigenvectors. The size of the entire dictionary (the number of atoms inside) is then equal to:

$$n + (n - (r - 1)) \times s^r, \tag{5}$$

where n is the number of original eigenvectors computed for a given input model with n vertices; r is the number of eigenvectors in a group that is used to create every new linear combination (e.g., $r = 2$ when using *pairs* of eigenvectors); and s is the number of scalar values considered for creating linear combinations of eigenvectors (19 in our case).

We chose to use the range [-0.9, 0.9] for the coefficient values, because we normalize all the dictionary atoms and the input vectors to have a length of 1, so any resulting inner product between an input vector and a dictionary atom will have a value between -1 and 1. We exclude the values -1 and 1 from the range because, if an input vector has an inner product of -1 or 1 with a dictionary vector, this means that this input vector already only needs just that one dictionary vector to represent it.

3.3. Matching pursuit for atom selection

To select atoms from our frame, we implemented Mallat and Zhang's Matching Pursuit (MP) algorithm [22]. We chose to use MP because it allows us to reconstruct our signal as a linear combination of the frame atoms, in an iterative manner, enabling us to easily apply our frame to the progressive compression of mesh geometry. Since our frame is not *tight* (see section 2), we cannot simply use the same frame for analysis and synthesis of the input, and pseudo-inversion of this frame would be very time-consuming (and probably sub-optimal) for large input models. MP is attractive because it is able to rapidly capture the most important components of the input signal. Furthermore, MP imposes very few restrictions on the dictionary construction, namely: (i) the dictionary must be at least *complete* for the input vector space; and (ii) the dictionary vectors should have unit norm. Our frame satisfies both of these requirements.

Formally, if the dictionary is defined as a family of vectors $\mathcal{D} = \{g_\gamma\}_{\gamma \in \Gamma}$ in a Hilbert Space $\mathbb{H}$, such that $\|g_\gamma\| = 1$ and $\mathcal{D}$ spans $\mathbb{H}$: For a given function $f \in \mathbb{H}$ we can obtain an M-term linear expansion of f using MP, by successive approximations of $R^n f$ through orthogonal projections onto the elements of $\mathcal{D}$ [11, 22]:

$$f = \sum_{n=0}^{M} \langle R^n f, g_{\gamma_n} \rangle \, g_{\gamma_n} + R^{M+1} f \tag{6}$$

where $R^n f$ is the residue after $n - 1$ iterations of the algorithm and $R^0 f = f$. At each iteration, the entire dictionary is searched to find the atom that has the largest absolute inner product with $R^n f$. For signals in finite-dimensional vector spaces, it has been shown [22] that the norm of the residues decays exponentially. The rate of decay depends on how closely the dictionary atoms match the input signal.

The authors in [22] claim that the largest possible dictionary $\mathcal{D}$ for representing a signal in a Hilbert space $\mathbb{H}$ would be the *set of all unit vectors in* $\mathbb{H}$, in which case the MP algorithm would converge in one iteration. This is the same theory that (independently) motivated us to construct a frame by computing many possible linear combinations of the vectors of an existing basis for a given vector space.

4. Using the proposed frame for progressive mesh geometry compression

The progressive compression algorithm that we propose is focused on minimizing the amount of data that must be transmitted between two users in a network. Because our focus is on mesh geometry compression, we assume that the connectivity will be compressed separately and sent to the decoder first. The decoder can then use the

connectivity information to compute the basis and frame, as described in sections 3.1 and 3.2, respectively. The encoder computes the basis and frame in the same way. The input mesh geometry in our compression algorithm is treated as three separate vectors, each corresponding to one set of coordinates, so that vector $\mathbf{X} = \{x_i\}_{i\epsilon I}$, vector $\mathbf{Y} = \{y_i\}_{i\epsilon I}$, and vector $\mathbf{Z} = \{z_i\}_{i\epsilon I}$, with $I = \{1, \dots, n\}$ (where n is the number of vertices in the input mesh). We transmit the norm of each of these vectors to the decoder. We then normalize each input vector to have *unit norm*: let us refer to these normalized vectors as $\hat{\mathbf{X}}$, $\hat{\mathbf{Y}}$, and $\hat{\mathbf{Z}}$. This is done so that when we are computing inner products between the input vectors and the dictionary atoms (which also have unit norm), the inner products will always be in the range [-1, 1]. This makes the inner products easier to interpret and compare, because they effectively represent a correlation value. We pass each of the normalized geometry vectors to the Matching Pursuit algorithm separately, because the input model may have different curvature properties in the x, y, and z directions. We transmit the coefficients and indices of the unique atoms chosen by MP, to the decoder in a progressive manner, according to the magnitude of the coefficients (largest-magnitude coefficients first). The decoder progressively uses the decoded coefficients, together with the frame elements corresponding to the received atom indices, to reconstruct the input mesh geometry with an increasingly better accuracy. Because the original geometry vectors are normalized to have unit norm at the encoder, and the best-matching dictionary atoms are found based on this, the geometry reconstructed from the received coefficients will not be immediately correct. To correct this, each reconstructed vector is multiplied by the corresponding original vector norm, which was sent by the encoder.

5. Experimental procedure

The objective in our experiments was to determine whether the frame that we create for a given input mesh is capable of producing a sparser representation for this mesh's geometry than the corresponding eigenvector basis can. To obtain a fair comparison between the number of coefficients (and atoms) required by a basis-based reconstruction versus a frame-based reconstruction, we selected the "significant" dictionary atoms in the same way in each case, using an own implementation of the Matching Pursuit algorithm (see section 3.3). For a basis-based reconstruction, the "dictionary" that we passed to the MP process was just the original eigenvector set for the given input mesh. For a frame-based reconstruction, the dictionary was the frame described in section 3.2, using firstly $r = 2$ (refer to equation (5)), then $r = 3$, and finally $r = 4$. For each case, we let the MP process continue running until the norm of the residual vector dropped to a "very small number" (1×10^{-4} is the value we used). We did *not* quantize or entropy code the produced coefficients (nor the atom indices), because we wished to isolate the compression achieved just by considering the number of coefficients required for a certain reconstruction quality, rather than including the compression effects of quantization and entropy coding as well. We then allowed the decoder to reconstruct the model progressively, building the mesh geometry up with one coefficient at a time. During reconstruction, we measured the error between each new mesh approximation and the original mesh geometry, by using three different error metrics: the *root mean square error* (RMSE), the *signal-to-noise ratio* (SNR), and the *Hausdorff distance* (dH), and plotted a *rate-distortion* (R-D) curve using each of these metrics. The *rate* in each R-D curve was represented as the number of coefficients used in the reconstruction so far. The results of our experiments for two test models are shown in the following section.

6. Results and discussion

We show here the results for the Torus model (50 vertices) and the Venus model (711 vertices), obtained from AIM@SHAPE and Gabriel Peyré, respectively (see Figure 1).

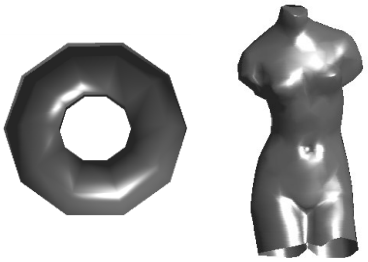

Figure 1: Torus (left) and Venus (right).

6.1. Basis vs frame reconstructions

In this section we present, separately, the R-D curves for a basis-based reconstruction versus a frame-based reconstruction, for the Torus and Venus models. In these tests the frames were created by considering *pairs* of eigenvectors ($r = 2$).

Figure 2 presents the R-D curves for the Torus. These plots show that the frame-based reconstructions have a better rate-distortion performance than the basis-based reconstructions, especially at the lower rates (numbers of coefficients used for reconstruction). This indicates that selecting atoms from the frame allows us to more rapidly capture the important features of a mesh than by just selecting atoms from the basis. Figure 3 shows the reconstructed Torus models obtained from using the first (largest) 14 coefficients, from both the basis and frame dictionaries. The superiority of the redundant representations is evident here: the Torus reconstructed by using the frame is almost visually identical to the original

model (see Figure 1), while the reconstruction with the orthogonal basis is of a much lower visual quality. Indeed, this is confirmed in the R-D plots in Figure 2: the RMSE and dH values for the frame-based reconstruction with 14 coefficients drop to almost 0, while the basis-based reconstructions still have higher error values. The SNR difference between the frame- and basis-based reconstructions is also quite significant here, with the frame SNR being around 30dB higher than the basis SNR at the 14 coefficient mark.

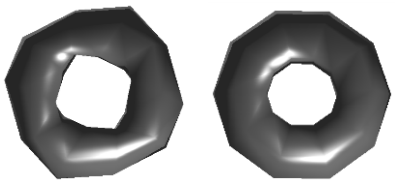

Figure 3: Torus models reconstructed with the largest 14 coefficients: Basis reconstruction (left) and frame reconstruction (right).

The rate-distortion results for the Venus model are shown in Figure 4. Similarly to the Torus, we can observe that for Venus the frame-based representation generally achieves a better mesh reconstruction quality at a lower rate (fewer coefficients) than the basis. Figure 5 shows the basis- and frame-based reconstructions of Venus, using the largest 50 coefficients. We can observe that the frame-based reconstruction with 50 coefficients captures not only

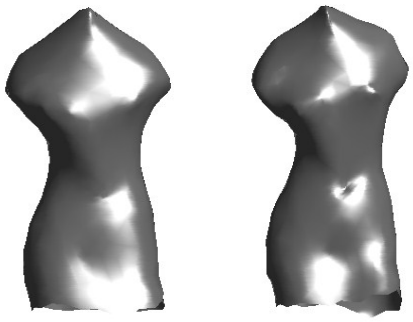

Figure 5: Venus models reconstructed with the largest 50 coefficients: Basis reconstruction (left) and frame reconstruction (right).

the basic shape of the model, but also begins to capture some of the finer details such as the belly button, the legs and chest area. This is in contast to the basis reconstruction with 50 coefficients, which manages to capture the basic shape of the model but does not really capture the finer detail.

6.2. Reconstructions from different frame constructions

In this section, we demonstrate the effects of increasing the redundancy in the frame for a given input model. We show here the R-D results for the Torus model using

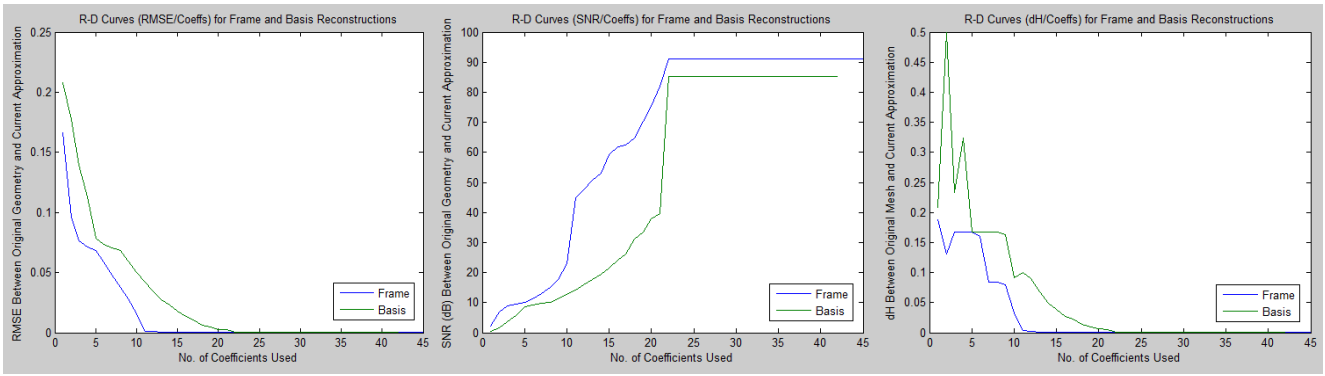

Figure 2: R-D curves for Torus.

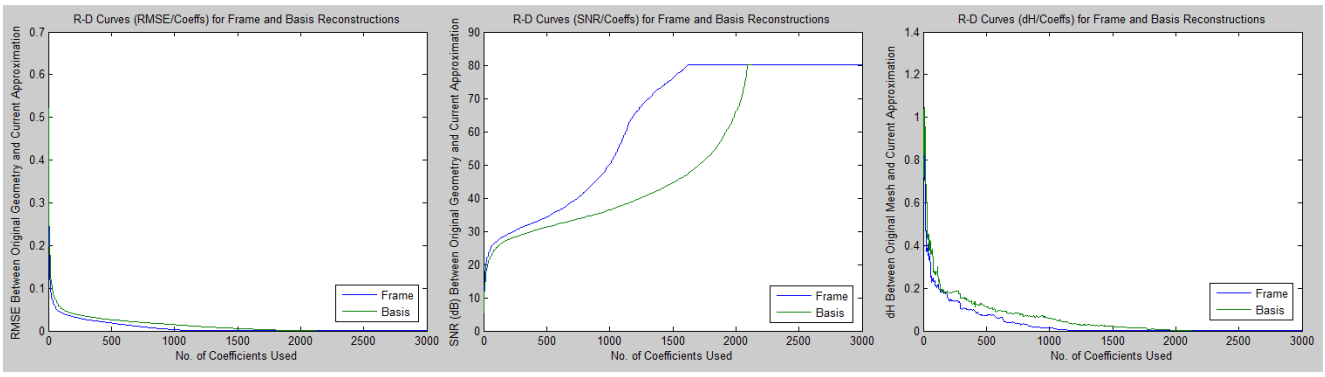

Figure 4: R-D curves for Venus.

$r = 3$ (refer to equation (5)) for the frame vector creation ("triplets" in Figure 6) and $r = 4$ ("quadruplets"), and compare these to the R-D curves for $r = 2$ ("pairs"). We include the basis R-D curve in each plot as well.

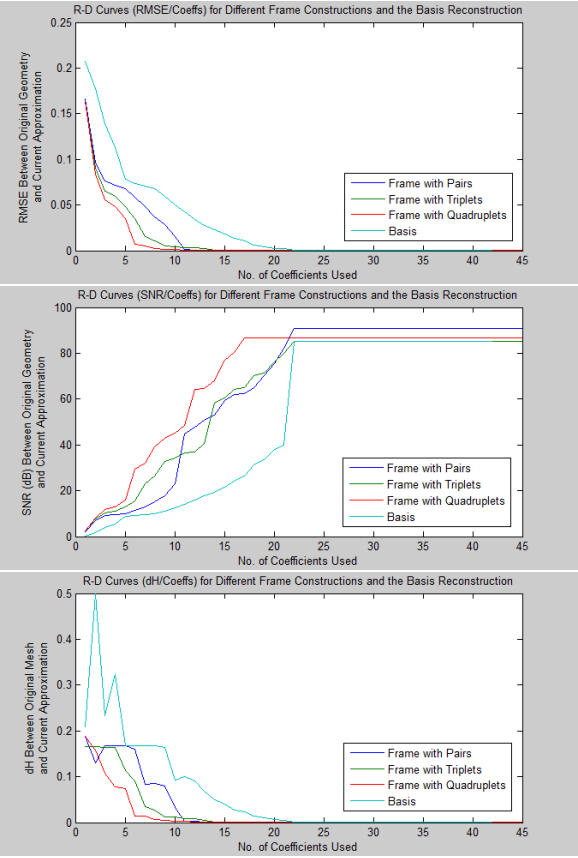

Figure 6: R-D curves for Torus when using different frame constructions.

The R-D curves for all three metrics in Figure 6 show a clear trend that, generally, the more eigenvectors we use in the linear combinations that produce the redundant vectors, the smaller the number of coefficients we need to transmit to achieve the same mesh approximation quality. We extrapolate from these results that the sparsest possible solution using this method would be obtained by computing each redundant vector as a linear combination of *all* the eigenvectors in the original basis (i.e., $r = n$), and computing every possible linear combination in this way. This is because the original eigenvector set is a basis for the finite vector space $\mathbb{R}^n$, which means that there exists a unique linear combination to represent *every* possible n-dimensional input vector. However, the trade-offs here include the time required to compute the larger redundant dictionary, the time taken to search through the larger dictionary, and the amount of storage space that would be required. From equation (5), we can deduce that increasing r by 1 increases the total dictionary size approximately 19 times. For example, for the Torus model, with $r = 2$ the total number of frame elements is 17,739; with $r = 3$ this number increases to 329,282; and with $r = 4$ it is 6,125,137. With $r = n$, the total number of frame elements would be around 9×10^{63}. The increase in computation time is theoretically also linear, increasing by a factor of 19 for each increase by 1 in r, but in practice this depends on the computational resources at hand. For example, in our implementation in MATLAB R2012b, on a machine with a 64-bit OS, 16GB RAM and a 3.30GHz CPU, the time taken to compute the frame for the Torus with $r = 2$ is only around 0.04 seconds and with $r = 4$ it is around 16.4 seconds. The time taken to perform one full dictionary search is also linear, increasing by a factor of 19 for each increase by 1 in r. Storing the frame for a given model with double-precision floating-point elements requires $8dn$ bytes of storage space, where d is the total number of atoms in the dictionary. For large dictionaries, in order to avoid overloading the main system memory, the dictionary may be saved on disk (provided that enough disk space is available) and loaded into main memory in chunks. Based on the results presented in this paper, our recommendation would be to use as high an r value as possible, given the available storage space and the acceptable computation time as determined by the user.

7. Benefits of the proposed frame and potential areas for improvement

The following sub-sections summarise the main benefits of our frame for 3D mesh geometry compression, and identify some areas that still require further work.

7.1. Benefits

Our frame enables a progressively better reconstruction of mesh geometry, requiring a smaller number of coefficients for reconstruction than when using an orthogonal basis, for the same reconstruction quality. This is especially useful in applications where rapid previewing of the model is required, e.g., in product visualization for e-commerce or a first prototype evaluation in collaborative product design. The frame is computed from the eigenvectors of the mesh Laplacian matrix, which have a meaningful physical interpretation in relation to the structure of the input model, and offer *directionality* and *orientability* of the atoms. The frame elements do not need to be transmitted between the encoder and decoder, because they can be generated independently at each end. The frame is also able to be applied directly on a given input mesh, regardless of its connectivity or topology type.

7.2. Areas for improvement

The main area for improvement is in optimizing our algorithm and implementation to be able to compute larger frames, for larger models, in a computationally efficient

manner. One way to do this might be to partition the input model into several smaller sub-meshes as was done in [5], and compute the eigenvectors and their linear combinations separately for each sub-mesh. Our frame is also currently computed separately for each input model; we wish to extend this to construct a larger frame that can work for an entire *class* of input models.

8. Conclusion

In this paper, we proposed a new method for creating a frame to be used for the progressive compression of 3D mesh geometry. The frame consists of the eigenvectors of the mesh Laplacian matrix, which form an orthogonal basis for the given mesh, plus a large number of redundant vectors created from linear combinations of these eigenvectors. Experimental results show that the frame-based representation of a given input mesh geometry has a better rate-distortion performance than representation of the mesh geometry in the eigenvector basis, as measured independently by three objective error metrics (the RMSE, the SNR, and the Hausdorff distance), in addition to visual inspection of the reconstructed models. The most pronounced differences between the basis and frame reconstructions are at the lower ends of the R-D curves, which indicates that the frame can more rapidly capture the important features of the input model than the orthogonal basis can. Our results also show that increasing the redundancy of the frame further improves the R-D performance of the algorithm. Other desirable properties of our frame include directionality and orientability of the atoms, as well as the ability to be applied directly to a manifold mesh with arbitrary topology and connectivity type. In future work, we will concentrate on finding a computationally efficient method for constructing larger frames with greater redundancy, which can be applied to an entire class of input models. We will also further investigate the parameters of our frame constructions, in order to optimize them to achieve the best possible rate-distortion trade-offs.

References

[1] H. Hoppe, "Progressive meshes," in *Proc. of SIGGRAPH 1996*, New Orleans, LA, USA, 1996, pp. 99-108.

[2] J. Popović and H. Hoppe, "Progressive Simplicial Complexes," in *Proc. of SIGGRAPH 1997*, Los Angeles, CA, USA, 1997, pp. 217-224.

[3] G. Taubin, *et al.*, "Progressive forest split compression," in *Proc. of SIGGRAPH 1998*, Orlando, FL, USA, 1998, pp. 123-132.

[4] P.-M. Gandoin and O. Devillers, "Progressive lossless compression of arbitrary simplicial complexes," *ACM Trans. Graph.,* 21(3):372-379, 2002.

[5] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proc. of SIGGRAPH 2000*, New Orleans, LA, USA, 2000, pp. 279-286.

[6] A. Khodakovsky, *et al.*, "Progressive Geometry Compression," in *Proc. of SIGGRAPH 2000*, New Orleans, LA, USA, 2000, pp. 271-278.

[7] H. Hoppe and E. Praun, "Shape Compression using Spherical Geometry Images," in *Advances in Multiresolution for Geometric Modelling*, N. A. Dodgson, *et al.*, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 27-46.

[8] A. Khodakovsky and I. Guskov, "Compression of Normal Meshes," in *Geometric Modeling for Scientific Visualization*, G. Brunnett, *et al.*, Eds., ed: Springer Verlag, 2002, pp. 189–206.

[9] J. Kovačević and A. Chebira, "An introduction to frames," *Signal Processing,* 2(1):1-94, 2008.

[10] G. Peyré and S. Mallat, "Surface compression with geometric bandelets," *ACM Trans. Graph.,* 24(3):601-608, 2005.

[11] I. Tošić, *et al.*, "Progressive Coding of 3-D Objects Based on Overcomplete Decompositions," *Circuits and Systems for Video Technology, IEEE Transactions on,* 16(11):1338-1349, 2006.

[12] I. Tošić and P. Frossard, "Dictionary Learning," *Signal Processing Magazine, IEEE,* 28(2):27-38, 2011.

[13] R. Rubinstein, *et al.*, "Dictionaries for Sparse Representation Modeling," *Proceedings of the IEEE,* 98(6):1045-1057, 2010.

[14] E. J. Candès and D. L. Donoho, "New tight frames of curvelets and optimal representations of objects with piecewise C2 singularities," *Communications on Pure and Applied Mathematics,* 57(2):219-266, 2004.

[15] L. Ying, *et al.*, "3D discrete curvelet transform," in *Optics & Photonics. Proc. of SPIE*, San Diego, CA, USA, 2005, pp. 591413-591413-11, vol. 5914.

[16] M. N. Do and M. Vetterli, "Contourlets: a new directional multiresolution image representation," in *Proc. of Signals, Systems and Computers 2002*, Pacific Grove, CA, USA, 2002, pp. 497-501, vol.1.

[17] Y. M. Lu and M. N. Do, "Multidimensional Directional Filter Banks and Surfacelets," *Image Processing, IEEE Transactions on,* 16(4):918-931, 2007.

[18] E. Le Pennec and S. Mallat, "Sparse geometric image representations with bandelets," *Image Processing, IEEE Transactions on,* 14(4):423-438, 2005.

[19] S. S. Chen, *et al.*, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing,* 20(1):33-61, 1998.

[20] R. Coifman, *et al.*, "Signal processing and compression with wavelet packets," in *Wavelets and Their Applications*. vol. 442, J. S. Byrnes, *et al.*, Eds., ed: Springer Netherlands, 1994, pp. 363-379.

[21] R. Gribonval and M. Nielsen, "Sparse decompositions in "incoherent" dictionaries," in *Proc. of ICIP 2003*, Barcelona, Spain, 2003, pp. I-33-6, vol.1.

[22] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on,* 41(12):3397-3415, 1993.

[23] Y. C. Pati, *et al.*, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proc. of Signals, Systems and Computers 1993*, Pacific Grove, CA, USA, 1993, pp. 40-44, vol.1.