# Compression of Normal Meshes

Andrei Khodakovsky[1] and Igor Guskov[2]

[1] Caltech *akh@cs.caltech.edu*
[2] University of Michigan *guskov@eecs.umich.edu*

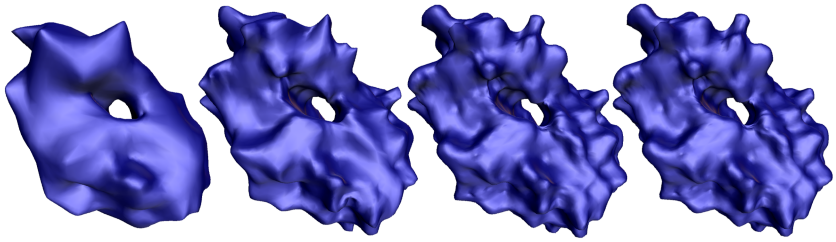


**Fig. 1.** Partial reconstructions from a progressive encoding of the molecule model. File sizes are given in bytes, errors in multiples of $10^{-4}$ and PSNR in dB (model courtesy of The Scripps Research Institute).

**Summary.** Normal meshes were recently introduced as a new way to represent geometry. A normal mesh is a multiresolution representation which has the property that all details lie in a known normal direction and hence the mesh depends only on a *single scalar per vertex*. Such meshes are ideally suited for progressive compression. We demonstrate such a compression algorithm for normal meshes representing complex, arbitrary topology surfaces as they appear in 3D scanning and scientific visualization. The resulting coder is shown to exhibit gains of an additional 2-5dB over the previous state of the art.

## 1 Introduction

The growth of computing power of personal computers and recent progress in shape acquisition technology facilitate the wide use of highly detailed meshes in industry and entertainment. Similarly, scientific visualization applications tend to produce ever finer meshes, such as iso-surfaces. In their raw, irregular form, acquired meshes are complex and often unmanageable due to their sheer size and irregularity [23]. It is therefore important to find more efficient and compact representations. Algorithms for efficient encoding of such meshes have been described in both single rate [37, 15, 35, 30, 29, 2] and progressive settings [27, 28, 24, 18, 6, 34, 3, 1]. For an overview of 3D geometry compression see [36].

One should recognize the fact that compression is always a trade-off between size and accuracy. That is especially true for meshes that come from shape acqui-

sition or iso-surface extraction, and which always carry sampling error and acquisition noise. Compression methods for such meshes can be lossy as long as the approximation error of the reconstruction stage is comparable with the sampling error. In the past eight years a number of efficient "remeshing" techniques have appeared that replace the original mesh with a mesh consisting of a number of "regular" pieces, such as B-spline [12], NURBS [20], or subdivision connectivity [22, 11, 21] patches. Recently, even more regular resampling techniques were introduced that use a single regular patch for the whole model [14]. Naturally, one should expect that the remeshed model should behave much better with regards to compression algorithms. This expectation was confirmed in [19], where the MAPS algorithm [22] was included as part of a progressive geometry coder. In particular, the paper [19] makes it clear that any mesh representation can be considered as having three components: geometry, connectivity, and parameterization; moreover, the last two components are not relevant for the representation of the geometry. For semi-regular mesh hierarchies, one can make a reasonable assumption that the normal component of the detail coefficients stores the geometric information, whereas the tangential components carry the parametric information (see also [16, 9]).

Most of the existing remeshing algorithms purposefully remove almost all of the connectivity information from the mesh, and also reduce the parametric information. One may wonder: *how much of the parametric information can be removed from the surface representation?* The answer is *almost all of it* as demonstrated in [17]. In fact, by fixing the transform algorithm to use unlifted Butterfly wavelets it is possible to build a semi-regular mesh whose details lie almost exclusively in the local normal direction. Consequently, the geometry of "normal" meshes is fully represented by a single scalar per vertex, instead of the usual three. Therefore it is natural to use normal meshes for compression as we do in this paper.

*Contribution* The goal of this paper is to demonstrate the additional coding gains possible when employing normal semiregular meshes rather than standard semiregular remeshes (such as those produced by MAPS [22]).

## 2 Compression Algorithms

The progressive geometry coding described in [19] requires three necessary components: a remeshing algorithm, a wavelet transform, and a zerotree coder. In [19] the MAPS remesher [22] was used, followed by the Loop or Butterfly wavelet transform. In this paper, we are using the normal remesher of [17] which was specifically designed to produce detail coefficients with no tangential components when an *unlifted Butterfly* wavelet transform is applied to the produced semi-regular mesh. We use the same zerotree coder as in [19]. The following sections will briefly overview the algorithms used for compression. For a more detailed exposition, the reader is referred to [19] and [17].

### 2.1 Normal Remesher

The normal remesher starts with a closed irregular mesh, and proceeds to build a semi-regular mesh hierarchy approximating the original model. The algorithm is described in [17] and consists of two stages. Using mesh simplification, a base mesh is chosen that is topologically equivalent to the original mesh. The connectivity of this base mesh will eventually become the connectivity of the coarsest level in the semi-regular hierarchy. Also at this stage a net of surface curves is initialized that splits the irregular mesh into a number of non-intersecting regions (these regions are in a one-to-one correspondence with the coarsest level faces of the semi-regular mesh being constructed). To complete the first stage of the algorithm, the net of surface curves is propagated to the finest level of the original irregular mesh, and a relaxation of global vertex positions within the surface is performed to even out their distribution and improve aspect ratios of the base mesh triangles.

In the second stage of the algorithm, a "piercing procedure" is applied recursively to obtain positions of finer level points of the semi-regular mesh. Thus, the semi-regular mesh is refined and, to maintain the status quo, the corresponding regions of the irregular mesh are split into smaller subregions. A global adaptive parameterization is maintained on the original mesh in order to keep the piercing process under control and to enable fast construction of surface curves. Surface patches are parameterized with Floater's parameterization scheme [13]. Every time a patch is split into smaller ones, a new parameterization is computed for the resulting subpatches.

The described two-stage process produces a semi-regular mesh that has mostly normal detail coefficients except for a small number of locations where the piercing did not find any "valid" intersection point with the original surface. For such exceptional vertices, the recorded detail is not scalar, and will have three components. The percentage of non-normal coefficients varies depending on the geometric properties of a given mesh and the corresponding coarse level points chosen in the first stage of the algorithm. Typically, the number of non-normal coefficients is below 10% for adaptive meshes that have the same number of vertices as the original irregular mesh. It may be possible to decrease the percentage of the non-normal details by a non-linear optimization approach, however we observed that the impact of non-normal coefficients on the compression results is not significant.

The interpolating character of the Butterfly subdivision scheme makes the construction of the normal remesh relatively straightforward since once a vertex is introduced during refinement its position stays the same on finer levels of the hierarchy. In contrast, the construction of normal meshes corresponding to any non-interpolating subdivision scheme (such as Loop) would have to be much more involved both algorithmically and computationally. In both interpolating and non-interpolating cases the normality condition introduces a system of constraints on the surface point sampling. However, for a non-interpolating scheme in order to satisfy these constraints one would need to use a global nonlinear optimization procedure. The metamesh approach to the normal mesh construction used for Butterfly normal meshes would become unwieldy for such an optimization, and a different,

possibly parameterization-based approach needs to be designed for Loop normal mesh construction. We leave it as a direction for future work.

*Displaced subdivision surfaces* A similar goal of achieving a "scalar" geometry description was addressed in the work on displaced subdivision surfaces [21]. That approach represents the original shape as a single resolution normal displacement from a base Loop subdivision surface, which is achieved by a careful construction of the base mesh. While the DSS representation is purely scalar, the typical sizes of the DSS base meshes are on the order of magnitude higher than the ones obtained in the normal remesher even for geometrically simple models [17][21].

### 2.2 Wavelet Transform

A semi-regular surface representation is a sequence of approximations at different levels of resolution: the corresponding nested sequence of linear spaces is given as $V_0 \subset V_1 \subset \ldots \subset V_n$. Here $V_0$ is the space of coarse base meshes, and $V_n$ is where the finest level meshes live. The wavelet transform maps this mesh hierarchy into a representation that consists of the base mesh and the sequence of wavelet coefficients that express differences between successive levels of the semi-regular mesh hierarchy. Thus, the wavelet transform decomposes the surface with respect to the hierarchy $V_n = V_0 + W_1 + \ldots + W_n$, where $V_j = V_{j-1} + W_j$ for $j = 1, \ldots, n$. [26]

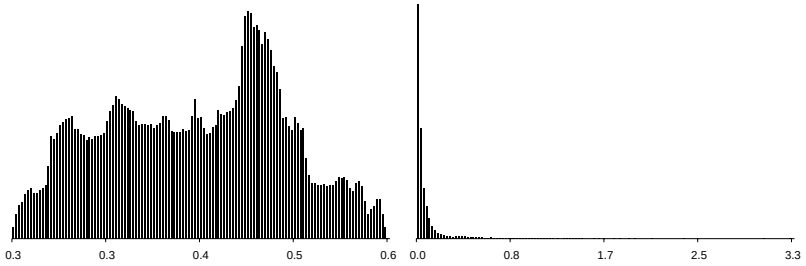


**Fig. 2.** Histograms of distribution of the original geometry and Loop wavelet coefficients for Venus head model.

Typically, an appropriate wavelet transform achieves a better statistical distribution of the coefficients in the surface representation. This works well because vertex coordinates of a smooth surface are correlated. This correlation can be exploited through the prediction of finer level geometry based on the coarser level geometry (low-pass filtering). The difference between the prediction and the actual geometry is represented by wavelet coefficients (high-pass filtering). For smooth surfaces we expect to achieve good prediction, which leads to smaller wavelet coefficients. Figure 2 shows the histograms of the distribution of original geometry coordinates as well as the distribution of the corresponding wavelet coefficients. Note that there are very few large coefficients (these actually contain important information) while the remaining majority of the coefficients are almost zero. Therefore, this representation is much more suitable for compression.

We start our wavelet construction by fixing a predictor, *i.e.*the subdivision scheme. The subdivision defines an embedding $V_j \subset V_{j+1}$.

$$\begin{pmatrix} s_e^{j+1} \\ s_o^{j+1} \end{pmatrix} = \begin{pmatrix} P_e \\ P_o \end{pmatrix} \left( s^j \right).$$

$P_o$ block computes positions of of new (*odd*) vertices and $P_e$ is an update on the old (*even*) points. There are two classes of subdivision schemes: *interpolating*(e.g. Butterfly) and *approximating*(e.g. Loop or Catmull-Clark) [38]. Interpolating schemes insert odd points as weighted averages of even points but never update the positions of even points ($P_e$ is the identity matrix). Approximating schemes not only compute odd points but also update positions of even points.

The wavelet space expresses the difference between finer and coarser spaces. The problem of designing the wavelet transform can be stated as a matrix completion problem: find such two matrices $Q_e$ and $Q_o$ that the whole transform $V_j + W_{j+1} \mapsto V_{j+1}$ is invertible.

$$\begin{pmatrix} s_e^{j+1} \\ s_o^{j+1} \end{pmatrix} = \begin{pmatrix} P_e \; Q_e \\ P_o \; Q_o \end{pmatrix} \begin{pmatrix} s^j \\ w^{j+1} \end{pmatrix}. \tag{1}$$

Of course, the properties of the wavelet transform greatly depend on the choice of the completion. In signal processing, the wavelets are typically required to define a stable transform, to have some number of vanishing moments, and to satisfy some orthogonality conditions. Additionally, the transforms with sparse $Q_e$ and $Q_o$ are preferable, since they can be implemented more efficiently. For surfaces, the wavelet theory is far from being fully developed, and the importance of some of the above mentioned qualities still has to be established.

In the remainder of this section we describe the particular wavelet transforms that we use for the geometric compression of semi-regular meshes.

*Linear subdivision.* First, we consider a simple case of the linear prediction (subdivision) scheme and the corresponding wavelet transform. The linear subdivision scheme computes the predicted geometry position of an odd point as the average of the two endpoints of the edge associated with the new point (Figure 4). The details (wavelet coefficients) are the differences between predicted and original positions of the odd points. Note, that this is an interpolating scheme, that is the even vertices do not change their values between levels of the hierarchy. Therefore, details are associated only with odd points. The wavelet transform preserves the number of degrees of freedom. This is true in general, though for approximating schemes the construction gets more complex [19].

After the above transform we adjust the scaling between the coarser coefficients and the details. Namely, the details on level $j$ are divided by $2^j$. This scaling arises from the $L_2$ normalization of the subdivision basis functions. Linear subdivision basis functions are hat functions centered at a vertex and supported on the one-ring of that vertex. In the pure subdivision setting, basis functions are normalized to have the same maximal value at all the levels of the hierarchy. However, for compression

purposes it is beneficial to decompose the surface with respect to an $L_2$ normalized basis. Since the area of the support shrinks by the factor of $4$ between successive levels, the finer basis functions must be multiplied by $2$, which accumulates so that the detail coefficients on level $j$ must be divided by $2^j$. Note that because of such a normalization the coarser details will be encoded with higher precision than the finer ones. The same scaling strategy applies not only for linear subdivision wavelets but also for all higher order schemes described below.

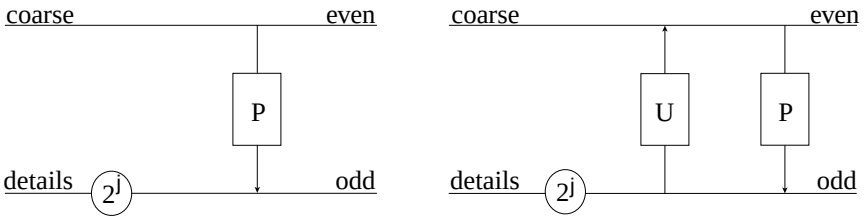


**Fig. 3.** The diagram illustrates the wavelet transform in the lifting framework. On the left the initial construction, on the right an update step is added.

A two-level transform can be illustrated with the following diagram (Figure 3). The wires on the left correspond to the coarse model (top) and details (bottom). The right hand side corresponds to the finer level geometry. The forward transform (analysis) is a transition from right to left: the predictor is evaluated on even vertices and subtracted from the odd vertex positions. Finally, the top wire is divided by $2^j$. Reconstruction is a transition in the opposite direction: the top wire is rescaled back, then the predictor, computed using coarser coefficients (bottom wire) is added to the details. Equivalently, the transforms are expressed by the following formulas:

$$\begin{pmatrix} s^{j-1} \\ w^j \end{pmatrix} = \begin{pmatrix} I & 0 \\ -2^{-j}\, P & 2^{-j}\, I \end{pmatrix} \begin{pmatrix} s^j_e \\ s^j_o \end{pmatrix}, \qquad \begin{pmatrix} s^j_e \\ s^j_o \end{pmatrix} = \begin{pmatrix} I & 0 \\ P & 2^j\, I \end{pmatrix} \begin{pmatrix} s^{j-1} \\ w^j \end{pmatrix}.$$

For stability and good approximation properties of a wavelet transform, it is important that the wavelet basis has some number of vanishing moments. In our construction, primal wavelets do not have even the zeroth vanishing moment. This problem can be fixed using the lifting construction: During the analysis step we modify coarse coefficients by subtracting a specially defined linear combination of just computed details (Figure 3). For example, using only the closest details with weight $-0.125$ enforces a zero moment of the primal wavelets. The same approach can be used to improve other properties of the wavelets. For instance, instead of enforcing moments, one can improve an angle between the wavelet space $W_i$ and the coarser space $V_{i-1}$.

*Higher order schemes.* The linear scheme is very simple and easy to implement but it does not produce smooth surfaces. Several other methods for building wavelet transforms on semi-regular meshes exist [26, 32, 4]. For compression of smooth surfaces it is advantageous to use higher order schemes. Such schemes can achieve better approximation leading to smaller detail coefficients and therefore to better

compression performance. In this work we used two particular wavelet transforms: Loop wavelets and non-lifted Butterfly wavelets.

The Loop [25] wavelets we use were first described in [19]. These have the advantage of using the Loop subdivision scheme in its inverse transform and perform quite well. However, a more natural wavelet transform to use in this work is the unlifted version of Butterfly wavelets [10, 39] because the *exact* same transform is used to produce normal meshes. A detailed description of the construction of Butterfly wavelets can be found in [32].

Figure 4 shows the Butterfly subdivision stencils in the regular case. The scheme uses an eight-point stencil and produces smooth surfaces. Similar to the linear case, an update step can be added to improve the wavelet transform. In general, such an update step can improve coding performance, but it is not suitable for normal mesh compression. The reason is that the normal mesh is constructed with respect to "pure" butterfly transform. Therefore, the wavelet coefficients will not be normal to the surface if any update step is added.

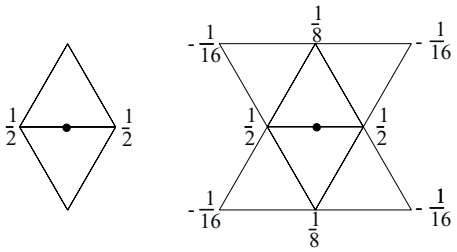


**Fig. 4.** Stencils for the linear (left) and Butterfly (right) subdivision in a regular case.

Note that the Butterfly wavelet transform uses finite filters for both analysis and reconstruction, and is therefore faster than the forward Loop transform which requires the solution of a sparse linear system. The Loop reconstruction filter has support of the same size as the Butterfly filter. On the other hand, we found that for non-normal meshes the Loop wavelet transform typically yields better visual appearance than the Butterfly transform, with comparable error measures (Figure 9).

*Local frames.* Typically, the $x$, $y$, and $z$ wavelet components are correlated [19]. We exploit this correlation by expressing wavelet coefficients in local frames induced by the coarser level. This is especially true for normal meshes. Almost all wavelet coefficients computed for normal meshes have only a normal component. Figure 5 shows histograms of the latitude angles $\theta$ (the angle from the normal axis) of the Butterfly and Loop wavelet coefficients in a local coordinate frame. Since the normal mesh is built using Butterfly subdivision almost all Butterfly coefficients have only normal components (those are seen as two peaks at $0$ and $180$ degrees of the histogram).

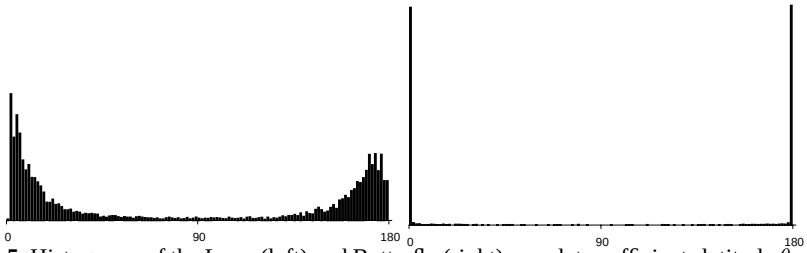


**Fig. 5.** Histograms of the Loop (left) and Butterfly (right) wavelet coefficients latitude $\theta$ angle for the Venus head model in a local frame.

### 2.3 Zerotree Coding

We encode components of wavelet coefficients separately, that is, our coder essentially consists of three independent zerotree coders. The bits from the three coders are interleaved to maintain progressivity.

A general principle of wavelet coefficient encoding is to send the highest order bits of the largest magnitude coefficients first. They will make the most significant contributions towards reducing error. The zerotree algorithm [33, 31, 8] groups all coefficients in hierarchical zerotree sets such that with high probability all coefficients in a given set are simultaneously below threshold.

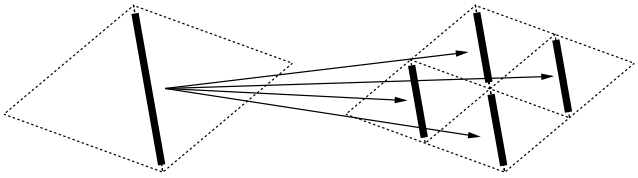

**Fig. 6.** A coarse edge (left) is parent to four finer edges of the same orientation (right).

The main distinction of our setting from the image case is the construction of the zerotrees. For images, one associates the coefficients with a quadrilateral face and the trees follow immediately from the face quadtree. For semi-regular mesh hierarchies, the main insight is that while scale coefficients are associated with vertices, wavelet coefficients have a one to one association with edges of the coarser mesh. Vertices do not have a tree structure, but edges do. Each edge is the parent of four edges of the same orientation in the finer mesh as indicated in Figure 6. Hence, each edge of the base domain forms the root of a zerotree; it groups all the wavelet coefficients of a fixed wavelet subband from its two incident base domain triangles. We found that the hierarchies, which have such a subband separation property lead to better compression performance (up to 0.5dB in some examples)

As was established in [19], the normal components of wavelet coefficients have more effect on geometric error than the tangential components. We therefore scale the normal components by four before quantization, so that they are effectively encoded with two more bits of precision than their tangential counterparts.

The scale coefficients from the coarsest level are uniformly quantized and progressively encoded. Each bitplane is sent as the zerotree descends another bit plane. Finally, the output of the zerotree algorithm is encoded using an arithmetic coder leading to about 10% additional reduction in the file size.

## 3 Adaptive Reconstruction

There is a trade-off between the remeshing error and the size of the resulting semi-regular mesh. With adaptive remeshing this trade-off is local. We refine the mesh where the error is maximal and leave it coarse where the remeshing error is small (Figure 7). Note, that the coarse remesh and its refinement give the same compression performance at low bit-rates, since refinement usually introduces small details at finer levels. These details are ignored by the zerotree coder at low bit-rates.

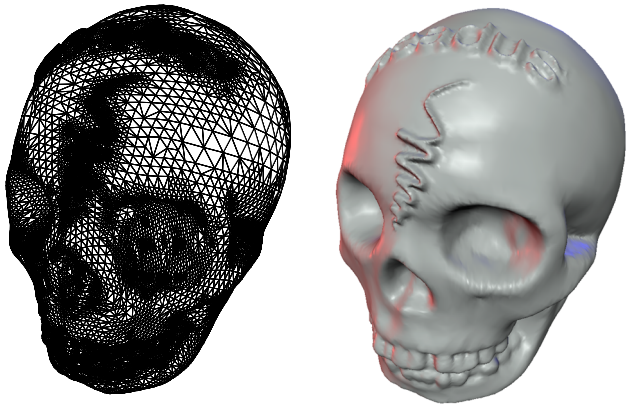

**Fig. 7.** Adaptive normal mesh for the skull (19138 vertices) with relative $L^2$ error of 0.02% and relative $L^\infty$ error of 0.09%. The base mesh is a tetrahedron (4 vertices) while the original mesh has 20002 vertices (model courtesy of Headus). For compression results see Figure 11.

The Butterfly based wavelet transform is fully adaptive: both analysis and reconstruction filters are finite. Therefore, it can take any adaptively remeshed semi-regular mesh as input. All regions that are not subdivided to the finest level define zero wavelet coefficients. These coefficients "virtually" exist as an extension of non-uniform zerotrees. The situation is more complicated for Loop wavelets: the analysis step requires solving a sparse linear system and its extension to adaptively refined meshes is not currently supported.

The zerotree coder step is a separate procedure that is naturally adaptive: there is no difference between non-uniform and uniform zerotrees with zeros attached to extra nodes of the latter.

During the reconstruction (for both Butterfly and Loop compression) we subdivide faces only when we decode wavelet coefficients that belong to those faces.

After we decode all the coefficients we subdivide all the faces until we meet an adaptive flatness threshold [40]. This step is important since we want to produce smooth surfaces even at low bit-rates. The flatness threshold is controlled by the user and can be chosen depending on the reconstruction bit-rate and the performance of the end-user graphics system.

## 4 Results and Discussion

We measured the performance of our coder [19] with normal meshes using both Butterfly and Loop wavelet transforms. For comparison we use the CPM coder of Pajarola and Rossignac [27] and the single-rate coder of Touma and Gotsman [37]. We plotted rate-distortion curves for the Touma-Gotsman coder by changing the coordinate quantization between 8 and 12 bits. We also compare our results with the performance of the coder of [19] for the Venus, horse and rabbit models.

Error was measured using the publicly available METRO tool [5]. All graphs show error between the reconstruction and the irregular original model as a function of the number of bits per vertex of the original mesh. All errors are given as PSNR (PSNR$= 20 \log_{10}($ BBoxDiag$/L^2$-error$)$).

| | Venus | rabbit | horse | dinosaur | skull | molecule | David |
|---|---|---|---|---|---|---|---|
| $V_{or}$ | 50K | 67K | 48K | 14K | 20K | 10K | 275K |
| $V_b$ | 42 | 71 | 112 | 128 | 4 | 53 | 283 |
| $E_r, 10^{-4}$ | 0.47 | 0.47 | 0.51 | 1.7 | 2.3 | 6.3 | 1.04 |
| $r_n$ | 1580 | 759 | 754 | 2973 | 1861 | 794 | 7500 |

**Table 1.** *Number of vertices in original models ($V_{or}$), base domain of remeshes ($V_b$), relative $L^2$ error ($E_r$) in units of $10^{-4}$, and the number of non-normal coefficients ($r_n$).*

We found that the coding with normal meshes has better performance compared to MAPS meshes. Even using a Loop wavelet transform we observe an improvement. For example, for the horse model (Figure 10) Loop wavelets on the normal mesh allow 1.5 times (3.5dB) smaller distortion than on the MAPS mesh. Note, that an improvement in the high bit-rate limit is due to the smaller remeshing error. More important is that we have better distortion for all bit-rates which happens because of the normality of the mesh. The additional improvement of about a factor of 1.2 (1.5dB) comes from using Butterfly wavelets summing up to a total of 5dB improvement compared to [19]. For the rabbit and Venus models the coders using MAPS and normal meshes have closer performance. However we still observe an improvement of at least 2dB.

Better compression rate of normal meshes obtained using Butterfly wavelets is not surprising given the fact that almost all the wavelet coefficients have a single nonzero component instead of three. A more remarkable result is a better compression rate of the normal meshes versus other remeshes when the Loop wavelets are used. Indeed, in that case one does not see scalar wavelet coefficients, and the reason for better performance is less explicit. We believe that this better performance

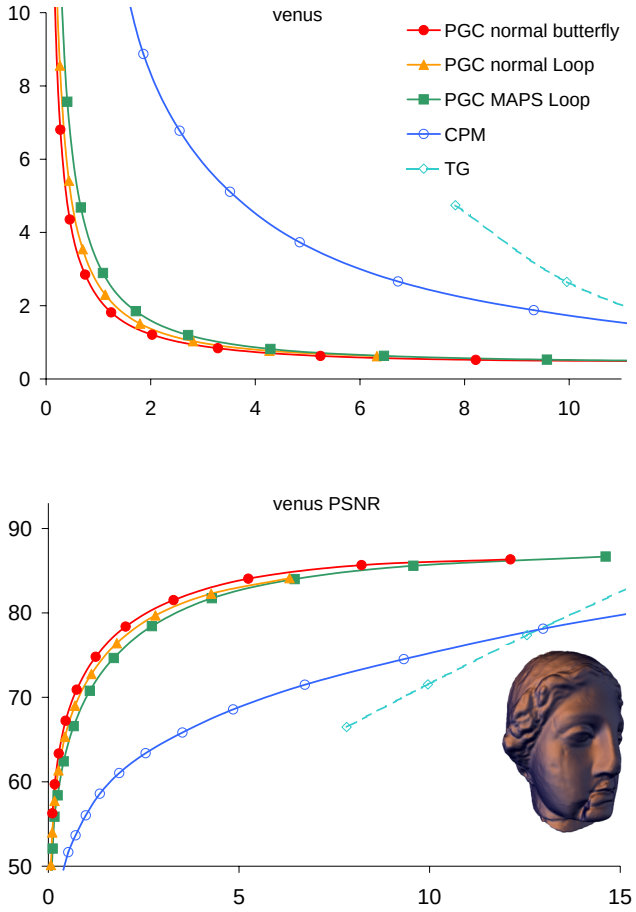


**Fig. 8.** Rate distortion for our coder with Butterfly and Loop wavelets using MAPS and normal meshes, TG, and CPM coders for the Venus model. At the top relative $L^2$ error in multiples of $10^{-4}$ as a function of bits/vertex. At the bottom the PSNR in dB.

can be explained by the smoothness of normal parameterization across the patch boundaries. One indication why this can be true is given by the following invariance property of the normal remesher: *given a Butterfly subdivision mesh as the input (together with extraordinary vertices as base vertices), the normal remesher will exactly reproduce the original mesh.* In practice, we need to also assume that all the normal piercing steps of the remesher succeeded which is usually true for meshes with good triangle aspect ratios. It is clear then that given a mesh well approximated by a subdivision surface one can hope to produce the normal mesh with small well compressible wavelet details. Since the Butterfly subdivision surface parameterization is smooth the normal mesh parameterization can be expected to be smooth as

**Fig. 9.** Comparison of partial reconstructions using butterfly (left) and Loop (right) wavelets for approximately the same PSNR value (compressed files are around 10KB). Note the bumps on the butterfly surface.

well. The corresponding result for the one-dimensional case of normal curves has been recently proven in [7].

The above discussion makes explicit the fact that the Floater parameterization is employed in the normal remesher in an auxiliary role – it is only used when the normal piercing fails. Otherwise, the local point sampling is fully independent of the particular parameterization scheme.

The fact that encoding with Loop wavelets benefits from Butterfly normal meshes is remarkable. Experimentally we found that if we treat wavelet components completely separately, Loop normal components compress better than Butterfly normal components. Given this evidence we should expect further improvement with Loop based normal meshes.

Figure 11 shows rate-distortion curves for the dinosaur, skull, and molecule models. These have coarser original meshes, therefore we allow a larger remeshing error (see the discussion on the natural choice of remeshing error in [19]). Note, that the adaptive remesh of the skull model has less vertices than the original, and has a reasonably small remeshing error. The base domain for the skull is a tetrahedron. In order to perform comparison with Loop based compression we uniformly subdivided the adaptive remesh with the Butterfly scheme.

For all the meshes mentioned in this paper (except for the David model), the remeshing step of our algorithm takes less than 10 minutes to complete. The forward Loop transform of the coder requires the solution of a sparse linear system, which takes about 30 seconds. Loop reconstruction and both Butterfly forward and backward transforms take 2-3 seconds. Zerotree encoding and decoding take about 1 second.

The largest model we were able to transform into the normal representation was a simplified model of the full David statue from the Digital Michelangelo project [23] (see Figure 12). Both the original and the remeshed model had on the order of 300,000 vertices. Our implementation of the normal remesher took

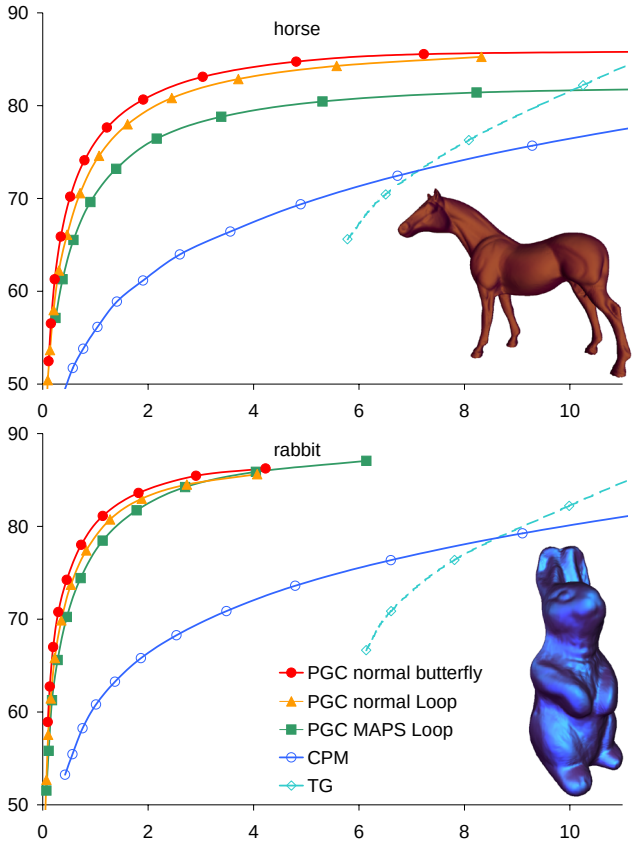


**Fig. 10.** Rate-distortion for rabbit and horse models showing PSNR as a function of bits/vertex.

thirty minutes to produce the normal mesh. It is clear that in order to remesh larger models, one needs to implement an out-of-core variant of the remeshing algorithm. Additionally, more efficient multigrid techniques for the patch parameterization can also be beneficial for better performance. We leave these improvements as future work.

## 5 Conclusions

In this paper we show that normal meshes improve performance of progressive geometry compression. We observe improvement of 2-5 dB depending on the model. We also describe adaptive compression which allows finer control on the number of vertices in the reconstructed model. The comparison between the Loop and Butterfly based compression methods is performed: in both cases compression is improved with normal meshes, and while the meshes compressed with Butterfly wavelets exhibit smaller mean-square error, the Loop based compression gives better visual quality of the reconstructed surfaces.

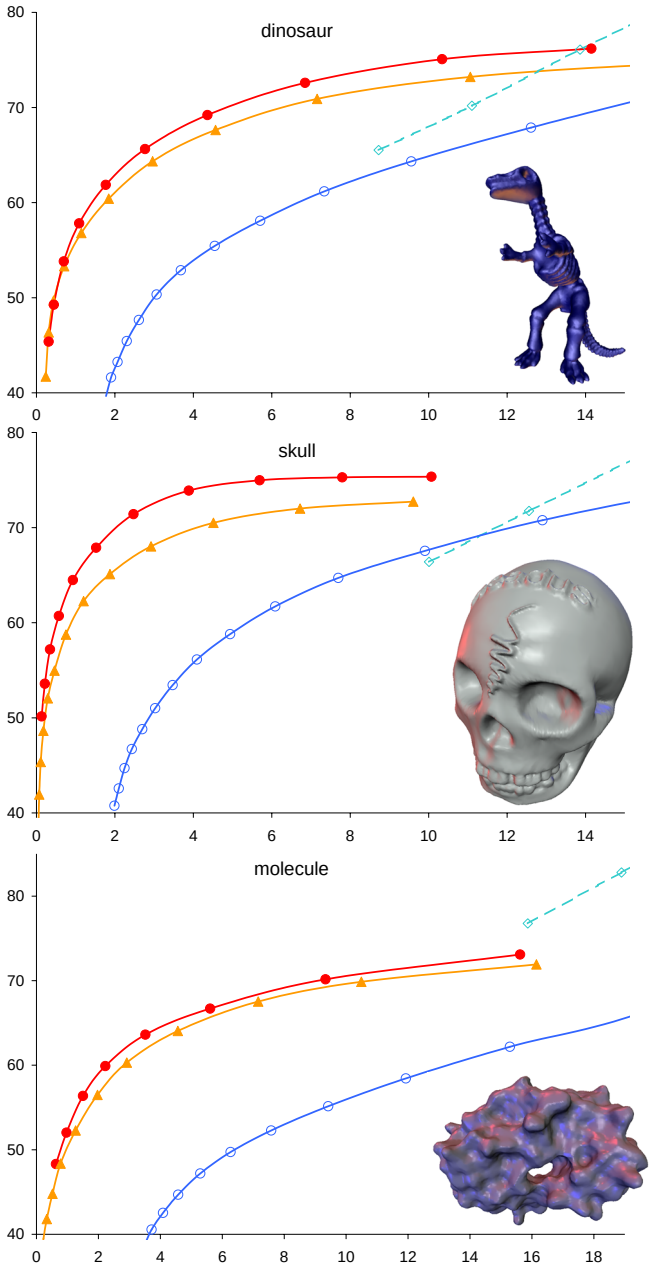


**Fig. 11.** Rate-distortion for our coder with Butterfly and Loop wavelets using normal meshes, TG and CPM coders for dinosaur, skull, and molecule models in PSNR as a function of bits/vertex.

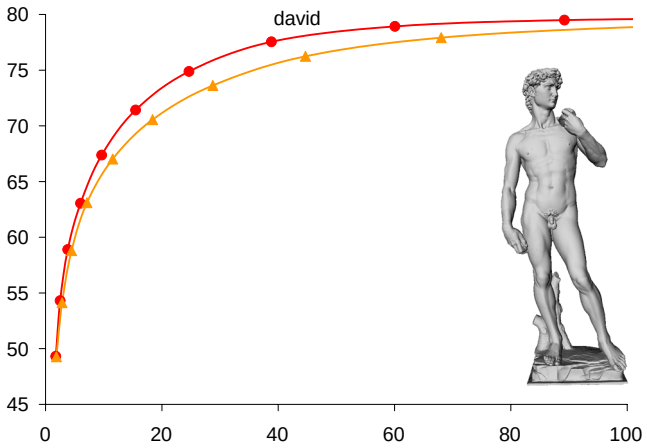


**Fig. 12.** Rate-distortion for our coder with Butterfly and Loop wavelets using normal meshes for a david model in PSNR as a function of file size (in $10^3$ Bytes).

Directions for future work include design of normal meshes using Loop wavelets, as well as generalization of the normal meshing to more topologically complex and dynamically changing surfaces.

*Acknowledgments* This work was supported in part by NSF (ACI-9624957, ACI-9721349, DMS-9874082, CCR-0133554). The paper was written with great encouragement and invaluable help from Wim Sweldens and Peter Schröder. Kiril Vidimče contributed to the current implementation of the normal remesher. Datasets are courtesy Cyberware, Headus, The Scripps Research Institute, University of Washington, and the Digital Michelangelo project at Stanford University [23].

## References

1. ALLIEZ, P., AND DESBRUN, M. Progressive compression for lossless transmission of triangle meshes. In *SIGGRAPH 2001 Conference Proceedings* (2001), pp. 198–205.
2. ALLIEZ, P., AND DESBRUN, M. Valence-driven connectivity encoding of 3d meshes. In *Eurographics 2001 Conference Proceedings* (Sept. 2001), pp. 480–489.
3. BAJAJ, C. L., PASCUCCI, V., AND ZHUANG, G. Progressive compression and transmission of arbitrary triangular meshes. *IEEE Visualization '99* (1999), 307–316.
4. BERTRAM, M., DUCHAINEAU, M. A., HAMANN, B., AND JOY, K. I. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization. In *Proceedings of IEEE Visualization 2000* (2000), pp. 389–396.
5. CIGNONI, P., ROCCHINI, C., AND SCOPIGNO, R. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.
6. COHEN-OR, D., LEVIN, D., AND REMEZ, O. Progressive compression of arbitrary triangular meshes. *IEEE Visualization '99* (1999), 67–72.
7. DAUBECHIES, I., RUNBORG, O., AND SWELDENS, W. Normal multiresolution approximation of curves. Preprint, Department of Mathematics, Princeton University, 2002.
8. DAVIS, G., AND CHAWLA, S. Image coding using optimized significance tree quantization. In *Prodeedings Data Compression Conference* (1997), IEEE, pp. 387–396.

9. DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. Implicit fairing of irregular meshes using diffusion and curvature flow. *Proceedings of SIGGRAPH 99* (1999), 317–324.
10. DYN, N., LEVIN, D., AND GREGORY, J. A. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics 9*, 2 (1990), 160–169.
11. ECK, M., DEROSE, T., DUCHAMP, T., HOPPE, H., LOUNSBERY, M., AND STUETZLE, W. Multiresolution analysis of arbitrary meshes. *Proceedings of SIGGRAPH 95* (1995), 173–182.
12. ECK, M., AND HOPPE, H. Automatic reconstruction of b-spline surfaces of arbitrary topological type. *Proceedings of SIGGRAPH 96* (1996), 325–334.
13. FLOATER, M. S. Parameterization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14* (1997), 231–250.
14. GU, X., GORTLER, S., AND HOPPE, H. Geometry images. *ACM SIGGRAPH 2002* (2002), 355–361.
15. GUMHOLD, S., AND STRASSER, W. Real time compression of triangle mesh connectivity. *Proceedings of SIGGRAPH 98* (1998), 133–140.
16. GUSKOV, I., SWELDENS, W., AND SCHRÖDER, P. Multiresolution signal processing for meshes. *Proceedings of SIGGRAPH 99* (1999), 325–334.
17. GUSKOV, I., VIDIMČE, K., SWELDENS, W., AND SCHRÖDER, P. Normal meshes. *Proceedings of SIGGRAPH 2000* (2000), 95–102.
18. HOPPE, H. Efficient implementation of progressive meshes. *Computers & Graphics 22*, 1 (1998), 27–36.
19. KHODAKOVSKY, A., SCHRÖDER, P., AND SWELDENS, W. Progressive geometry compression. *Proceedings of SIGGRAPH 2000* (2000), 271–278.
20. KRISHNAMURTHY, V., AND LEVOY, M. Fitting smooth surfaces to dense polygon meshes. *Proceedings of SIGGRAPH 96* (1996), 313–324.
21. LEE, A., MORETON, H., AND HOPPE, H. Displaced subdivision surfaces. In *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)* (2000), pp. 85–94.
22. LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. Maps: Multiresolution adaptive parameterization of surfaces. *Proceedings of SIGGRAPH 98* (1998), 95–104.
23. LEVOY, M. The digital michelangelo project. In *Proceedings of the 2nd International Conference on 3D Digital Imaging and Modeling* (Ottawa, October 1999).
24. LI, J., AND KUO, C. Progressive coding of 3-d graphic models. *Proceedings of the IEEE 86*, 6 (1998), 1052–1063.
25. LOOP, C. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.
26. LOUNSBERY, M., DEROSE, T. D., AND WARREN, J. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics 16*, 1 (1997), 34–73. Originally available as TR-93-10-05, October, 1993, Department of Computer Science and Engineering, University of Washington.
27. PAJAROLA, R., AND ROSSIGNAC, J. Compressed progressive meshes. Tech. Rep. GIT-GVU-99-05, Georgia Institute of Technology, 1999.
28. PAJAROLA, R., AND ROSSIGNAC, J. SQUEEZE: Fast and progressive decompression of triangle meshes. In *Proc. CGI* (2000).
29. ROSSIGNAC, J. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (1999), 47–61.
30. ROSSIGNAC, J., AND SZYMCZAK, A. Wrap&zip: Linear decoding of planar triangle graphs. Tech. Rep. GIT-GVU-99-08, Georgia Institute of Technology, 1999.

31. SAID, A., AND PEARLMAN, W. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transaction on Circuits and Systems for Video Technology 6*, 3 (1996), 243–250.
32. SCHRÖDER, P., AND SWELDENS, W. Spherical wavelets: Efficiently representing functions on the sphere. *Proceedings of SIGGRAPH 95* (1995), 161–172.
33. SHAPIRO, J. Embedded image-coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing 41*, 12 (1993), 3445–3462.
34. TAUBIN, G., GUEZIEC, A., HORN, W., AND LAZARUS, F. Progressive forest split compression. *Proceedings of SIGGRAPH 98* (1998), 123–132.
35. TAUBIN, G., AND ROSSIGNAC, J. Geometric compression through topological surgery. *ACM Transactions on Graphics 17*, 2 (1998), 84–115.
36. TAUBIN, G., AND ROSSIGNAC, J., Eds. *3D Geometry Compression*. No. 21 in Course Notes. ACM Siggraph, 1999.
37. TOUMA, C., AND GOTSMAN, C. Triangle mesh compression. *Graphics Interface '98* (1998), 26–34.
38. ZORIN, D., AND SCHRÖDER, P., Eds. *Subdivision for Modeling and Animation*. Course Notes. ACM SIGGRAPH, 1999.
39. ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interpolating subdivision for meshes with arbitrary topology. *Proceedings of SIGGRAPH 96* (1996), 189–192.
40. ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97* (1997), 259–268.