3D Mesh Geometry Filtering Algorithms for Progressive Transmission Schemes

Radu Balan *

Siemens Corporate Research

755 College Road East, Princeton, NJ 08540, USA

rvbalan@scr.siemens.com

Gabriel Taubin

IBM T.J.Watson Research Center

P.O.Box 704 Yorktown Heights , NY 10598 , USA

taubin@watson.ibm.com

March 9, 2000

Keywords: Compression , Geometry, Multi Resolution , Filter

*This paper was written while the first author was a postdoctoral associate at IBM (T.J.Watson Research Center) and IMA (University of Minnesota)

Abstract

A number of static and multi-resolution methods have been introduced in recent years to compress 3D meshes. In most of these methods the connectivity information is encoded without loss of information, but user-controllable loss of information is tolerated while compressing the geometry and property data. All these methods are very efficient at compressing the connectivity information, in some cases to a fraction of a bit per vertex, but the geometry and property data typically occupies much more room in the compressed bitstream than the compressed connectivity data. In this paper we investigate the use of polynomial linear filtering as studied in [Taubin95, TaZhGo96], as a global predictor for the geometry data of a 3D mesh in multi-resolution 3D geometry compression schemes. Rather than introducing a new method to encode the multi-resolution connectivity information, we choose one of the efficient existing schemes depending on the structure of the multi-resolution data. After encoding the geometry of the lowest level of detail with an existing scheme, the geometry of each subsequent level of detail is predicted by applying a polynomial filter to the geometry of its predecesor lifted to the connectivity of the current level. The polynomial filter is designed to minimize the l^2-norm of the approximation error but other norms can be used as well. Three properties of the filtered mesh are studied next: accuracy, robustness and compression ratio. The Zeroth Order Filter (unit polynomial) is found to have the best compression ratio. But higher order filters achieve better accuracy and robustness properties at the price of a slight decrease of the compression ratio.

1 Introduction

Polygonal models are the primary 3D representations for the manufacturing, architectural, and entertainment industries. They are also central to multimedia standards such as VRML and MPEG-4. In these standards, a polygonal model is defined by the position of its vertices (geometry); by the association between each face and its sustaining vertices (connectivity); and optional colors, normals and texture coordinates (properties).

Several single-resolution [TaRo98, LiKuo98] and multi-resolution methods [Hoppe96, PoHo97, TaGuHoLa98, Ross99] have been introduced in recent years to represent 3D meshes in compressed form for compact storage and transmission over networks and other communication channels. In most of these methods the connectivity information is encoded without loss of information, and user-controllable loss is tolerated while compressing the geometry and property data. In fact, some of these methods only addressed the encoding of the connectivity data [GuSt98]. Multi-resolution schemes reduce the burden of generating hierarchies of levels on the fly, which may be computationally expensive, and time consuming. In some of the multi-resolution schemes the levels of detail are organized in the compressed data in progressive fashion, from low to high resolution. This is a desirable property for applications which require transmission of large 3D data sets over low bandwith communication channels. Progressive schemes are more complex and typically not as efficient as single-resolution methods, but reduce quite significantly the latency in the decoder process.

In this paper we investigate the use of polynomial linear filtering [Taubin95, TaZhGo96], as a global predictor for the geometry data of a 3D mesh in multi-resolution 3D geometry compression schemes. As in other multi-resolution geometry compression schemes, the geometry of a certain level of detail is predicted as a function of the geometry of the next coarser level of detail. However, other 3D geometry compression schemes use simpler and more localized prediction schemes.

Although we concentrate on the compression of geometry data, property data may be treated similarly. The methods introduced in this paper apply to the large family of multi-resolution connectivity encoding schemes. These linear filters are defined by the connectivity of each level of detail and a few parameters, which in this paper are obtained by minimizing a global criterion related to certain desirable properties. Our simulations present the least square filters compared with some other standard filters.

In [PaRo99] the Butterfly subdivision scheme is used for the predictor. In [GuSwSc99], a special second order local criterion is minimized to refine the coarser resolution mesh. The latter class of algorithms have concentrated on mesh simplification procedures and efficient connectivity encoding schemes. For instance in the Progressive Forest Split scheme [TaGuHoLa98], the authors have used a technique where the sequence of splits is determined based on the local volume conservation criterion. Next, the connectivity can be efficiently compressed as presented in the aformentioned paper or as in [Ross99].

Mesh simplification has been also studied in a different context. Several works address the remeshing problem, usually for editing purposes. For instance in [Eck&all95] the harmonic mapping is used to resample the mesh. Thus the remashing is obtained by minimizing a global curvature-based energy criterion. A conformal map is used in [Lee&all98] for similar purposes, whereas in [MaYaVe93] again a global length based energy criterion is used to remesh.

The organization of the paper is the following: in section 2 we review the mesh topology based filtering and introduce the basic notions; in section 3 we present two geometry encoding algorithms; in section 4 we analyze three desirable properties, accuracy, robustness and compression ratio; in section 5 we present numerical and graphical results; finally, the conclusions are contained in section 6 and are followed by the bibliography.

2 Mesh Topology Based Filtering

Consider a mesh $(\mathcal{V},\mathcal{F})$ given by a list of vertex coordinates $\mathcal{V}$ (the *mesh geometry*) of the nV vertices, and a list of polygonal faces $\mathcal{F}$ (the *mesh connectivity*). The mesh geometry can be thought of as a collection of three vectors (x,y,z) of length nV containing, respectively, the three coordinates of each vertex; alternatively we can see $\mathcal{V}$ as representing a collection of nV vectors $(r_0,r_1,\ldots,r_{nV})$ of length 3, each of them being the position vector of some mesh vertex. To the list $\mathcal{F}$ we associate the symmetric $nV \times nV$ vertex to vertex incidence matrix M, and the $nV \times nV$ matrix K defined by:

$$K = I - DM \tag{1}$$

where D is the $nV \times nV$ diagonal matrix whose (i,i) element is the inverse of the number of first order neighbors the vertex i has. As shown in [Taubin95], K has nV real eigenvalues all in the interval $[0,2]$.

Consider now a collection $P = (P_x(X), P_y(X), P_z(X))$ of three polynomials each of degree d, for some positive integer d.

Definition We call P a *polynomial filter of length* $d+1$ (and *degree* or *order* d), where its action on the mesh $(\mathcal{V},\mathcal{F})$ is defined by a new mesh $(\mathcal{V}',\mathcal{F})$ of identical connectivity but of geometry $\mathcal{V}' = (x', y', z')$ given by:

$$x' = P_x(K)x \quad , \quad y' = P_y(K)y \quad , \quad z' = P_z(K)z \tag{2}$$

A *rational filter* (Q, P) of orders (m, n) is defined by two collections of polynomials (Q_x, Q_y, Q_z) and (P_x, P_y, P_z) of degrees m, respectively n, whose action on the mesh $(\mathcal{V},\mathcal{F})$ is defined by the new mesh $(\mathcal{V}',\mathcal{F})$ through:

$$Q_x(K)x' = P_x(K)x \quad , \quad Q_y(K)y' = P_y(K)y \quad , \quad Q_z(K)z' = P_z(K)z \tag{3}$$

To avoid possible confusions, we assume $Q_x(K), Q_y(K)$ and $Q_z(K)$ invertible. We point out the filtered mesh has the same connectivity as the original mesh; only the geometry changes. Note also the filter works for non-manifold connectivity as well.

In this report we consider only polynomial filters, i.e. rational filters of the form $(1, P)$. In [DeMeScBa99], the authors considered the case $(Q, 1)$. Note the distinction between polynomial and rational filters is artificial. Indeed, any rational filter is equivalent to a polynomial filter of length nV, in general, and in fact, any polynomial filter of degree larger than nV is equivalent to a polynomial filter of degree at most $nV - 1$. These facts are results of the Cayley-Hamilton theorem (see [FrInSp79], for instance) that says the characteristic polynomial of K vanishes when applied on K. Therefore:

$$Q(K)^{-1}P(K) = P_0(K) \tag{4}$$

for some polynomial P_0 of degree at most $nV - 1$. Hence the notion of IIR (Infinite Impulse Response) filter does not have any correspondence in the mesh topology based filtering, because any polynomial of higher order or rational filter is equivalent to a polynomial filter of degree at most $nV - 1$, thus a FIR (Finite Impulse Response) filter. However, the difference between polynomial and rational filters lays in their implementation. The polynomial filter is easily implemented by a forward iteration scheme. The rational filter can be implemented by a forward-backward iteration scheme:

$$\begin{aligned} w &= P_x(K)x \\ Q_x(K)x' &= w \end{aligned} \tag{5}$$

involving the solution of a linear system of size nV. For small degrees m, n compared to nV (when the rational form has an advantage), the backward iteration turns into a sparse linear system, and thus efficient methods can be applied to implement it.

Two particular filtering schemes are of special importance to us and are studied next. The first scheme is called the *Zeroth Order Filter* and is simply defined by the constant polynomial 1:

$$P_Z(X) = (1, 1, 1) \tag{6}$$

Technically speaking, with the order definition given before, this is *a* zero order filter, but the most general form of zero order filters would be constant polynomials, not necessary 1. However, throughout this paper we keep this convention to call the constant polynomial 1, the Zeroth Order Filter. Note its action is trivial: it does not change anything.

The second distinguished filtering scheme is called *Gaussian Smoothing* and it is a first order filter defined by:

$$P_G(X) = (1 - X, 1 - X, 1 - X) \tag{7}$$

Using the definition of K and the filter action on the mesh geometry, the geometry of the Gaussian filtered mesh is given by:

$$x' = DMx \quad , \quad y' = DMy \quad , \quad z' = DMz \tag{8}$$

which turns into the following explicit form (using the position vectors r_i and the first order neighborhood i^* of vertex i):

$$r_i^{'} = \frac{1}{|i^*|} \sum_{v \in i^*} r_v \tag{9}$$

In other words, the new mesh geometry is obtained by taking the average of the first order neighbors positions on the original mesh.

3 The Progressive Approximation Algorithms

In Progressive Transmission schemes, the original mesh is represented as a sequence of succesively simplified meshes obtained by edge collapsing and vertex removal. Many simplification techniques have been proposed in the literature. For instance in [TaGuHoLa98] the Progressive Forest Split method is used. It consists of partitioning the mesh into disjoint patches and in each patch a connected sequence of edge collapsing is performed.

The meshes we are using here have been simplified by a clustering procedure. First all the coordinates are normalized so that the mesh is included in a 3D unit cube. Next the cube is divided along each coordinate axis into 2^B segments (B is the *quantizing rate*, representing the number of bits per vertex and coordinate needed to encode the geometry), thus obtaining 2^{3B} smaller cubes. In each smaller cube all the edges are collapsed to one vertex placed in the center of the corresponding cube. The mesh such obtained represents the quantized mesh at the finest resolution level. The coarsening process proceeds now as follows: 2^{3K} smaller cubes are replaced by one of edge size 2^K times bigger, and all the vertices inside are removed and replaced by one placed in the middle of the bigger cube. Next, the procedure is repeated until we obtain a sufficiently small number of vertices (i.e. a sufficient coarse resolution).

At each level of resolution, the *collapsing ratio* (i.e. the number of vertices of the finer resolution, divided by the number of vertices of the coarser resolution) is not bigger than 2^{3K}. In practice, however, this number could be much smaller than this bound, in which case some levels may be skipped. After l steps, the number of bits needed to encode one coordinate of any such vertex is $B - lK$. Thus, if we consider all the levels of detail and a constant collapsing ratio R, the total number of bits per coordinate needed to encode the geometry becomes:

$$M_b = NB + \frac{N}{R}(B-K) + \frac{N}{R^2}(B-2K) + \cdots + \frac{N}{R^L}(B-LK)$$

where N is the initial number of vertices and L the numbers of levels. Assuming $\frac{1}{R^L} \ll 1$ we obtain $M_b = NB\frac{R}{R-1} + NK\frac{R}{(R-1)^2}$. Thus, the number of bits per vertex (of initial mesh) and coordinate turns into:

$$N_{bits} \simeq \frac{R}{R-1}(B + \frac{K}{R-1}) \;\; [bits/vertex \cdot coordinate] \tag{10}$$

Thus, if we quantize the unit cube using $B = 10$ bits and we resample at each level with a coarsening factor of $2^K = 2$ and a collapsing ratio $R = 2$, we obtain the sequence has $B/K = 10$ levels of details encoded using an average of 22 $bits/vertex \cdot coordinate$, or 66 $bits/vertex$ (including all three coordinates). Thus a single resolution encoding would require only B (10, in this example) bits per vertex and coordinate in an uncompressed encoding. Using the clustering decomposition algorithm, the encoding of all levels of details would require N_{bits} (given by (10)), about 22 in this example, which is more than twice the single resolution rate.

In this scenario no information about the coarser resolution mesh has been used to encode the finer resolution mesh. In a progressive transmission, the coarser approximation may be used to predict the finer approximation mesh and thus only the differences should be encoded and transmitted. Moreover, the previous computations did not take into account the internal redundancy of the bit stream. An entropic encoder would perform much better than (10). In this paper we do not discuss the connectivity encoding problem, since we are interested in the geometry encoding only. Yet, we assume at each level of detail the decoder knows the connectivity of that level mesh.

Suppose $(Mesh_{nL-1}, map_{nL-2,nL-1}, Mesh_{nL-2}, map_{nL-3,nL-2}, \ldots, map_{1,2}, Mesh_1, map_{0,1}, Mesh_0)$ is the sequence of meshes obtained by coarsening algorithm, where $Mesh_{nL-1}$ is the coarsest resolution mesh, $Mesh_0$ the finest resolution mesh, and $map_{l-1,l} : \{0, 1, \ldots, nV_{l-1} - 1\} \rightarrow \{0, 1, \ldots, nV_l - 1\}$ is the *collapsing map* that associates to the nV_{l-1} vertices of the finer resolution mesh the nV_l vertices of the coarser resolution mesh where they collapse. Each mesh $Mesh_l$ has two components $(Geom_l, Conn_l)$, the geometry and connectivity respectively, as explained earlier. We are concerned with the encoding of the sequence of geometries $(Geom_{nL-1}, Geom_{nL-2}, \ldots, Geom_1, Geom_0)$. Our basic encoding algorithm is the following:

The Basic Encoding Algorithm

Step 1. Encode $Geom_{nL-1}$ using an entropic or arithmetic encoder;

Step 2. For $l = nL - 1$ down to 1 repeat:

Step 2.1 Based on mesh $Mesh_l$ and connectivity $Conn_{l-1}$ find a set of parameters $Param_{l-1}$ and construct a predictor of the geometry $Geom_{l-1}$:

$$\hat{Geom}_{l-1} = Predictor(Mesh_l, Conn_{l-1}, map_{l-1,l}; Param_{l-1})$$

Step 2.2 Encode the parameters $Param_{l-1}$;

Step 2.3 Compute the approximation error $Diff_{l-1} = Geom_{l-1} - \hat{Geom}_{l-1}$ and encode the differences. $\diamond$

The decoder will reconstruct the geometry at each level by simply adding up the difference to his prediction:

$$Geom_{l-1} = Predictor(Mesh_l, Conn_{l-1}, map_{l-1,l}; Param_{l-1}) + Diff_{l-1}$$

It is clear that different predictors yield different performance results. In the next section we present several desirable properties of the encoding scheme.

The data packet structure is represented in Table 1.

$Mesh_{nL-1}$	$Map_{nL-2,nL-1}\&Conn_{nL-2}$	$Param_{nL-2}$	$Diff_{nL-2}$	$\cdots$
$\cdots$	$Map_{0,1}\&Conn_0$	$Param_0$	$Diff_0$	

Table 1: The data packet structure for the basic encoding algorithm

The *Predictor* consists of applying the sequence of operators *extension*, where the geometry of level l is extended to level $l-1$, and *update*, where the geometry is updated using a polynomial filter whose coefficients are called *parameters* of the predictor and whose matrix is the finer resolution incidence matrix K_{l-1}.

The extension step is straightforwardly realized using the collapsing maps:

$$r_i^{l-1;ext} = r^l_{map_{l-1,l}(i)} \tag{11}$$

Thus the first "prediction" of the new vertex i is on the same point where it collapses, i.e. the position of the vertex $map_{l-1,l}(i)$ in mesh l. Next, the updating step is performed by polynomial filtering as in (2). The filter coefficients are the predictor parameters and have to be found and encoded. On each coordinate we use a separate filter. In the next section we introduce different criteria to measure the prediction error associated to a specific property. In [TaGuHoLa98] Taubin filters (i.e. of the form $P(X) = (1-\lambda X)(1+\mu X))$ have been used as predictors, but no optimization of the parameters has been done. Here we use more general linear filters taking into account several performance criteria as well.

More specific, let us denote by $x_i^{l-1;ext}$ the nV_{l-1}-vector of x-coordinates obtained by extension (11), and by $x^{l-1;updt}$ the filtered vector with the polynomial $P_x(X) = \sum_{k=0}^d c_k X^k$ of degree d,

$$x^{l-1;updt} = P_x(K_{l-1}) x^{l-1;ext} \tag{12}$$

Let r^{l-1} denote the $nV_{l-1} \times 3$ matrix containing all the coordinates in the natural order, $r^{l-1} = [x^{l-1}|y^{l-1}|z^{l-1}]$. Similar for $r^{l-1;updt}$. The update is our prediction for the geometry $Geom_{l-1}$. Then the coefficients are chosen to minimize some l^p-norm of the prediction error:

$$\min_{Filters\ Coefficients} J^{l-1} = \|r^{l-1} - r^{l-1;updt}\|_{l^p} \tag{13}$$

Note the optimization problem decouples into 3 independent optimization problems, because we allow different filters on each coordinate. The polynomial $P_x(X)$ can be represented either in the power basis, i.e. $P_x(X) = \sum_{k=0}^d c_k X^k$, or in another basis. We tried the Chebyshev basis as well, in which case $P_x(X) = \sum_{k=0}^d c_k T_k(X)$ with T_k the k^{th} Chebyshev polynomial. On each coordinate, the criterion J^{l-1} decouples as follows:

$$J^{l-1} = \|(J_x^{l-1}, J_y^{l-1}, J_z^{l-1})\|_{l^p} \ , \ J_x^{l-1} = \|A_x c_x^{l-1} - x^{l-1}\|_{l^p} \ , \ J_y^{l-1} = \|A_y c_y^{l-1} - y^{l-1}\|_{l^p} \ , \ J_z^{l-1} = \|A_z c_z^{l-1} - z^{l-1}\|_{l^p} \ , \tag{14}$$

where the $nV \times d+1$ matrix A_x is either

$$A_x = [x^{l-1;updt} | K_{l-1} x^{l-1;updt} | \cdots | K_{l-1}^d x^{l-1;updt}] \tag{15}$$

in the power basis case, or

$$A_x = [x^{l-1;updt} | T_1(K_{l-1}) x^{l-1;updt} | \cdots | T_d(K_{l-1}) x^{l-1;updt}] \tag{16}$$

in the Chebyshev basis case. c_x^{l-1} is the $d+1$ - vector of the x-coordinate filter coefficients and x^{l-1} the nV_{l-1}-vector of the actual x-coordinates all computed at level $l-1$. Similar for A_y, A_z, c_y, c_z, and y^{l-1}, z^{l-1}.

The Basic Encoding Algorithm can be modified to a more general context. The user may select the levels for which the differences are sent. Then, for those levels the differences are not sent, the extension step to the next level has to use the predicted values instead of the actual values of the current level. In particular we may want to send the differences starting with level $nL-1$ and going down to some level $S+1$; then, from level S down to level 0 we do not send any difference but just the parameters, excepted for the level 0 when we send the differences as well. The algorithm just described is presented next:

The Variable Length Encoding Algorithm

Step 1. Encode $Mesh_{nL-1}$;

Step 2. For $l = nL-1$ down to S repeat:

Step 2.1 Estimate the parameters $Param_{l-1}$ by minimizing J^{l-1}, where the predictor uses the true geometry of level l, $Geom_l$:

$$\hat{Geom}_{l-1} = f(Conn_l, Conn_{l-1}, map_{l-1,l}, Geom_l; Param_{l-1})$$

Step 2.2 Encode the parameters $Param_{l-1}$;

Step 2.3 If $l \neq S$, encode the differences $Diff_{l-1} = Geom_{l-1} - \hat{Geom}_{l-1}$;

Step 3. For $l = S-1$ down to 1

Step 3.1 Estimate the parameters $Param_{l-1}$ by minimizing J^{l-1} where the predictor uses the estimated geometry of level l:

$$\hat{Geom}_{l-1} = f(Conn_l, Conn_{l-1}, map_{l-1,l}, \hat{Geom}_l; Param_{l-1})$$

Step 3.2 Encode the parameters $Param_{l-1}$;

Step 4. Using the last prediction of level 0, encode the differences, $Diff_0 = Geom_0 - \hat{Geom}_0$. $\diamond$

In this case the data packet structure is the one represented in Table 2. In particular, for $S = nL$ only the last set of differences is encoded. This represents an alternative to the single-resolution encoding scheme.

$Mesh_{nL-1}$	$Map_{nL-2,nL-1}\&Conn_{nL-2}$	$Param_{nL-2}$	$Diff_{nL-2}$	$\cdots$	$Map_{s+2,2+1}\&Conn_{s+1}$		
	$Param_{s+1}$	$Diff_{s+1}$	$Map_{s+1,s}\&Conn_s$	$Param_s$	$Map_{s,s-1}\&Conn_{s-1}$	$Param_{s-1}$	$\cdots$
$\cdots$	$Map_{2,1}\&Conn_1$	$Param_1$	$Map_{0,1}\&Conn_0$	$Param_0$	$Diff_0$		

Table 2: The data packet structure for the variable encoding algorithm

4 Desired Properties

In this section we discuss three properties we may want the encoding scheme to possess. The three peoperties, accuracy, robustness, compression ratio, yield different optimization problems all of the type mentioned before. The l^p-norm to be minimized is different in each case. For accuracy the predictor has to minimize the l^∞ norm, for robustness the l^2 norm should be used, whereas the compression ratio is optimized for $p \in [1, 2]$ in general. Thus a sensible criterion should be a trade-off between these various norms. Taking the computational complexity into account, we have chosen the l^2-norm as our criterion and in the following section of examples we show several results we have obtained.

4.1 Accuracy

Consider the following scenario: Suppose we choose $S = nL$, the number of levels, in the Variable Length Encoding Algorithm. Suppose also the data block containing the level zero differences is lost (note this is the only data block containing differences because $S = nL$) In this case we would like to predict the finest resolution mesh as accurately as possible based on the available information. Equivalently, we would like to minimize the distance between $Mesh_0$ and the prediction $\hat{Mesh}_0$, under the previous hypotheses. There are many ways of measuring mesh distances. One such measure is the Haussdorf distance. Although it describes very well the closeness of two meshes, the Haussdorf distance yields a computational expensive optimization problem. Instead of Haussdorf distance one can consider the maximum distance between vertices (i.e. the l^∞-norm, see [Al&all88]):

$$\varepsilon_a = \max_{0 \le i \le nV_0 - 1} \|r_i^0 - \hat{r}_i^0\| =: \|r^0 - \hat{r}^0\|_{l^\infty}$$

Note the l^∞-norm is an upper bound for the Haussdorf distance. Consequently ε_a controls the meshes closeness as well. As mentioned in the previous section, the optimization problem (13) decouples in three independent optimization problems. For $p = \infty$, these have the following form:

$$\inf_c \|Ac - b\|_{l^\infty} \tag{17}$$

where A was introduced by (15) and (16), depending on the basis choice, c is the nf-vector of unknown fiter coefficients, and b is one of the three vectors x, y or z. For $0 \le i \le nV - 1, 0 \le j \le fL - 1$, $A = [a_{ij}]$, $b = (b_i)$ and writing $c_j = f_j - g_j$ with $f_j \ge 0$, the positive part, and $g_j \ge 0$, the negative part of c_j (thus at least one of them is always zero), the optimization problem (17) turns into the following linear programming problem:

$$\begin{array}{ll} \max\limits_{w, f_j, g_j, u_i, v_i} & [-w - \varepsilon \sum_{j=0}^{d} (f_j + g_j)] \\ \textit{subject to}: & w, f_j, g_j, u_i, v_i \ge 0 \\ & b_i = u_i + \sum_{j=0}^{d} a_{ij}(f_j - g_j) - w \\ & b_i = -v_i + \sum_{j=0}^{d} a_{ij}(f_j - g_j) + w \end{array} \tag{18}$$

with ε a small number to enforce at least one of f_j or g_j to be zero (for instance $\varepsilon = 10^{-6}$). With the standard simplex algorithm, this problem requires the storage of a $(2nV + 2) \times (2nV + 2d + 2)$-matrix (the so called *tableaux*) which is prohibitive for large number of vertices (nV of order 10^5, for instance). In any case, the moral of this subsection is to point out that the more accurate predictor is the one that achieves a lower l^∞-norm error.

4.2 Robustness

Consider now the following scenario: the differences associated to the prediction algorithm are not set to zero but perturbed by some random quantities. This may be due to several causes. We can either imagine irretrievable transmission errors or even a resampling process at the transmitter to reduce the code length of the entire object. In any case we assume the true difference d_i is perturbed by some stochastic process ν_i. Thus the reconstructed geometry has the form $x_i^{l-1;reconstr} = x_i^{l-1;updt} + diff_i + \nu_i$. We assume the perturbations are about of the same size as the prediction differences. Next suppose we want to minimize *in average* the effect of these perturbations. Then one such criterion is the noise variance $E[\nu_i^2]$. Assuming the stochastic process is ergodic, it follows the noise variance can be estimated by the average of all the coordinate perturbations: $E[\nu_i^2] = \frac{1}{N}\sum_{i=0}^{N-1} \nu_i^2$. Next, since the perturbation is of the same order as the prediction error, the later term can be replaced by the average of the differences. Hence we want to minimize:

$$E[\nu_i^2] \simeq \frac{1}{N}\sum_{i=0}^{N-1} d_i^2$$

This shows that a criterion of robustness is the total energy of the differences. In this case our goal to increase the robustness of the algorithm is achieved by decreasing the l^2-norm of the prediction errors. Thus the filters are the solvers of the optimization problem (13) for $p = 2$. The solution in terms of filter coefficients is very easily obtained by using the pseudoinverse matrix. Thus the solution of:

$$\inf_c ||Ac - b||_{l^2}$$

is given by:

$$c = (A^T A)^{-1} A^T b \tag{19}$$

4.3 Compression Ratio

The third property we discuss now is the compression ratio the algorithm achieves. In fact, if no error or further quantization is assumed, the compression ratio is perhaps the most important criterion in judging and selecting an algorithm. In general estimating compression ratios is a tough problem due to several reasons. First of all one should assume a stochatic model of the data to be encoded. In our case we encode the vectors of prediction errors, which in turn depend on the mesh geometry and the way we choose the filters coefficients. Next one should have an exact characterization of the encoder's compression ratio. The best compression ratio, assuming a purely stochastic data, is given by Shannon's entropic formula and consequently by the entropic encoder which strives to achieve this bound (Shannon-Fano and Huffman codings - see [ZiTr90] or [DaGr76]). However the entropic encoder requires some a priori information about the data to be sent, as well as overhead information that may affect the global compression ratio. Alternatively one can use adaptive encoders like the adaptive arithmetic encoder as in the JPEG/MPEG standards (see [PeMi93]). This encoder may perform better in practice than *blind* entropic or arithmetic encoders, however it has the important shortcoming that its compression ratio is not characterized by a closed formula. In any case, for purely stochastic data the best compression ratio is bounded by Shannon's formula which we discuss next. We thus assume our bit sequence encoding scheme achieves this optimal bound. Suppose the quantized differences x_i, $0 \leq i \leq N-1$, are independently distributed and have a known probability distribution, say $p(n)$, $-2^{B-1} \leq n \leq 2^B$. Thus $p(n)$ is the probability that a difference is n. In this case the average (i.e. expected value) of the number of bits needed to encode one such difference is not less than:

$$R_{Shannon} = -\sum_{n=-2^{B-1}}^{2^{B-1}} p(n) log_2 p(n)$$

where 2^B is the number of quantization levels (see [ZiTr90]). Assuming now the ergodic hypothesis holds true, $p(n)$ can be replaced by the repetition frequency $p(n) = \frac{f(n)}{N}$, where $f(n)$ is the repeatition number of the value n and N is the total number of values (presumably $N = 3nV$). Thus, if we replace the first $p(n)$ in the above formula by this frequency, the sum turns into

$$R = -\frac{1}{N}\sum_{i=0}^{N-1} log_2 p(n = x_i)$$

Note the summation index has changed. At this point we have to asssume a stochastic model for the prediction errors. We consider the power-type distribution that generalizes both the Gaussian and Laplace distributions, that are frequently used in computer graphics models (see [PaRo99], for instance):

$$p(x) = \frac{a^{\frac{1}{\alpha}}\alpha}{2\Gamma(\frac{1}{\alpha})} exp(-a|x|^{\alpha}) \tag{20}$$

where $\Gamma(x)$ is the Euler's Gamma function (to normalize the expression) and a is a parameter. For $\alpha = 1$ it becomes the Laplace distribution, whereas for $\alpha = 2$ it turns into the Gauss distribution. Then, the previous rate formula turns into:

$$R = R_0 + \frac{a log_2 e}{N}\sum_{i=0}^{N-1} |x_i|^{\alpha} \quad , \quad R_0 = 1 + log_2\Gamma(\frac{1}{\alpha}) - log_2\alpha - \frac{1}{\alpha}\log_2 a$$

Now we replace the parameter a by an estimate of it. An easy computation shows the expected value of $|x|^{\alpha}$ for the α-power p.d.f. (20) is $E[|x|^{\alpha}] = \frac{1}{\alpha a}$. Thus we get the following estimator for the parameter a:

$$\hat{a} = \frac{1}{\alpha}\frac{N}{\sum_{i=0}^{N-1}|x_i|^{\alpha}}$$

and the above formula of the rate becomes:

$$R = \rho_0(\alpha) + \frac{1}{\alpha}\log_2[\sum_{i=0}^{N-1}|x_i|^{\alpha}] \quad , \quad \rho_0(\alpha) = 1 + \log_2\frac{\Gamma(\frac{1}{\alpha})}{\alpha} + \frac{1}{\alpha}\log_2\frac{e\alpha}{N} \tag{21}$$

Consider now two linear predictors associated to two different linear filters. Each of them will have different prediction errors. If we assume the prediction errors are independent in each case and distributed by the same power law with exponent α but maybe different parameters a_1, respectively a_2, then the prediction scheme that yields the sequence of differences with smaller l^α-norm has a better (entropic) compression bound and therefore is more likely to achieve a better compression ratio. Equivalently, the p.d.f. that has a larger parameter a, or is narrower, would be encoded using fewer bits.

The argument we presented here suggests that a better compression ratio is achieved by the prediction scheme that minimizes the l^α-norm of the prediction error, where α is the p.d.f.'s characteristic exponent (when it is a power-type law), usually between 1 (the Laplace case) and 2 (the Gaussian case). For $p = 2$ the optimizing filter is found by using the pseudoinverse of A as in (19). For $p = 1$, the optimizer solves the linear programming problem:

$$\begin{array}{ll} \max_{f_j, g_j, u_i, v_i} & \sum_{i=0}^{N-1}[-u_i - v_i - \varepsilon \sum_{j=0}^{d}(f_j + g_j)] \\ \textit{subject to}: & f_j, g_j, u_i, v_i \geq 0 \\ & b_i = u_i - v_i + \sum_{j=0}^{d} a_{ij}(f_j - g_j) \end{array} \tag{22}$$

with ε as in (18), which involves (in the simplex algorithm) a $(N+2) \times (2N + 2d + 1)$ matrix and the same computational problems as (18).

5 Examples

In this section we present a number of examples of our filtering algorithm. For several meshes we study the accuracy the fine resolution mesh is approximated, and also the compression ratio obtained for different filter lengths.

First we analyze the Basic Encoding Algorithm presented in section 3. The filters coefficients are obtained by solving the optimal problem (13) for $p = 2$, i.e. we use the least squares solution.

The car mesh represented in Figure 1 (left) having $nV_0 = 12784$ vertices and 24863 faces is decomposed into a sequence of 8 levels of details. The coarsest resolution mesh of $nV_7 = 219$ vertices is rendered in Figure 1 (right). We used several filter lengths to compress the meshes. In particular we study four types of filters, namely the Zeroth Order Filter, the Gaussian Smoother and filters of order $d = 1$ and $d = 3$ (decomposed in power basis). The last two filters will be termed as "higher order filters", although their order is relatively low. To check the accuracy of the approximation we used the prediction algorithm assuming the differences are zero at all levels. The four meshes corresponding to the four filters are represented in Figure 2.

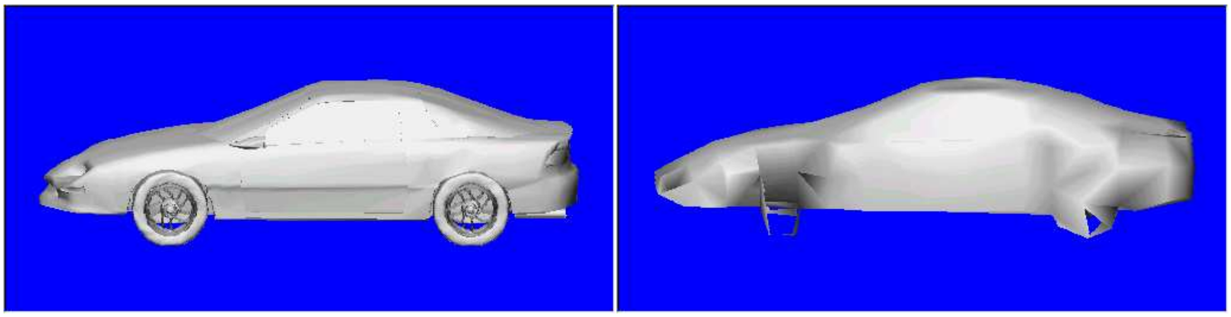

Figure 1: The car mesh at: the finest resolution (left) and the coarsest resolution (right) after 7 levels of reduction.

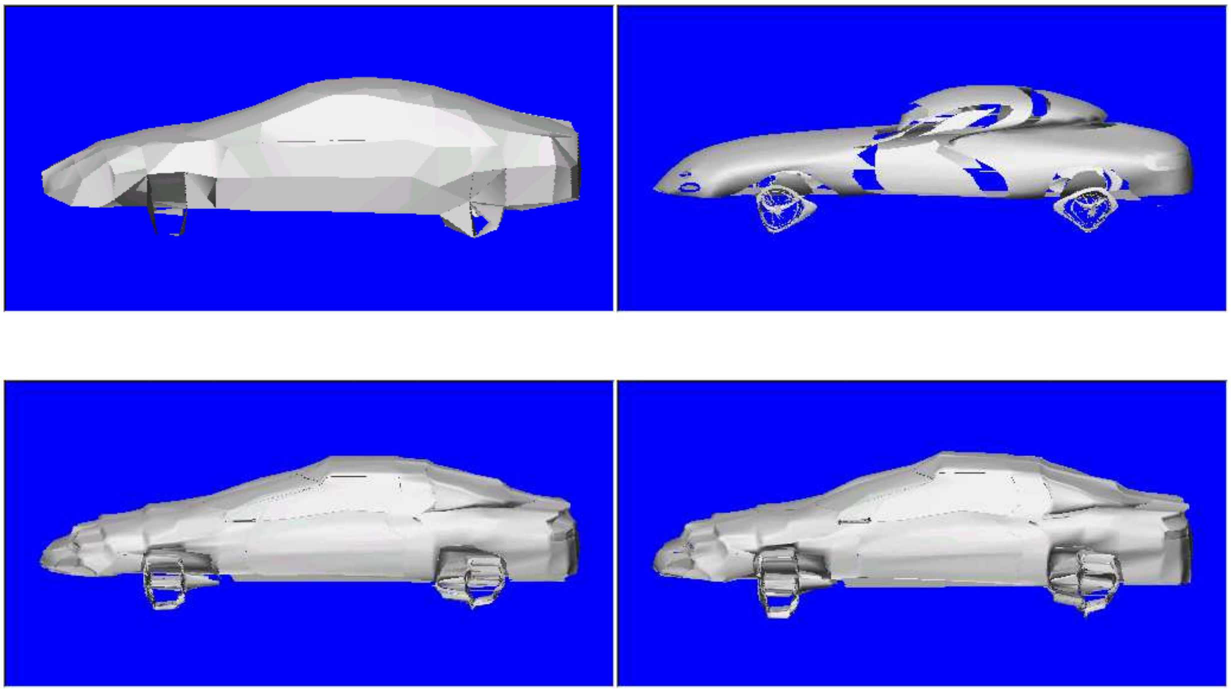

Figure 2: The car mesh at the finest resolution level when no difference is used and the filtering is performed by: the Zeroth Order Filter (top left), the Gaussian Smoother (top right), the least squares filter of order 1 (bottom left) and the least squares filter of order 3 (bottom right).

Note the Zeroth Order Filter does not change the geometry at all (because of its pure extension nature). It gives the worst approximation of the mesh, yet it has the best compression ratio (see below). The Gaussian filter smoothes out all the edges, a natural consequence since it really corresponds to a discrete diffusion process. The higher order filters (i.e. first order and third order) trade-off between smoothing and compression.

In terms of the compression ratio, the four filters have performed as shown in Table 3. Varying the filter length we found the compression ratios indicated in Table 4. All the results apply to the geometry component only. The connectivity is not presented here.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	795	795	795	795
Coefficients	0	0	168	336
Differences	21840	64611	22520	22574
Total (bytes)	22654	65425	23503	23725
Rate (bits/vertex)	14.17	40.94	14.71	14.82

Table 3: Compression ratio results for several filtering schemes applied to the car mesh rendered in Figure 1.

Filter's Degree	1	2	3	4	5	6	7
bits/vertex	14.71	14.82	14.85	14.89	14.96	15.04	15.11

Table 4: Compression ratios for different filter lengths in power basis.

Next we study the Variable Length Encoding Algorithm for the four particular filters mentioned before with the parameter $S = nL$ (i.e. in the Single Resolution case). Thus the mesh geometry is obtained by successively filtering the extensions of the coarser mesh and, at the last level, the true differences are encoded. In terms of accuracy we obtained very similar meshes. More significantly are the compression ratios, shown in Table 5. To analyse the compression ratios of these four filters, we have also plotted the histogram of the errors on a semilogarithmic scale in Figure 3. Note the power-type p.d.f. hypothesis is well satisfied by the Zeroth, LS 1^{st} and LS 3^{rd} order filters, and less by the Gaussian smoother. Also as smaller the l^2-norm error gets, as narrower the p.d.f. and as smaller the rate becomes, in accordance with the conclusions of Section 4.3.

Equally important is how these errors are distributed on the mesh. In Figure 4 we convert the actual differences into a scale of colors and set this color as an attribute for each vertex. Darker colors (blue,green) represent a smaller error, whereas lighter colors (yellow, red) represent a larger prediction error. The darker the color the better the prediction and also the accuracy. All the errors are normalized with respect to the average l^2-norm error per vertex for that particular filter. The average l^2-norm error is given on the last row in Table 5.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	795	795	795	795
Coefficients	0	0	168	336
Differences	27489	27339	25867	25532
Total (bytes)	28306	28156	26859	26690
Rate (bits/vertex)	17.71	17.62	16.81	16.68
l^2 error/vertex($\cdot 10^{-4}$)	11.96	18.64	9.33	8.44

Table 5: Compression ratios in the Single Resolution implementation of the Variable Length Encoding Algorithm applied to the car mesh rendered in Figure 1.

Note in the Single Resolution case there is no much difference among the filtering schemes considered. In particular the higher order filters perform better than the Zeroth Order Filter, and the Gaussian filter behaves similarly to the other filters. This is different to the Multi Resolution case in Table 3. There, the Gaussian filter behaves very poorly, and the Zeroth Order Filter gives the best compression ratio. In fact it is better to the Single Resolution case. On the other hand, with respect to the accuracy, the higher order filters give a more accurate approximation than the Zeroth Order Filter.

About the same conclusions hold for three other meshes we used: the round table, the skateboard and the piping construction.

The round table rendered in Figures 5, left, has $nV_0 = 11868$ vertices and 20594 faces. The coarsest resolution mesh (at level 8, pictured on the right side) has $nV_7 = 112$ vertices. The predicted mesh after 8 levels of decomposition when no difference is used, is rendered in Figure 6.

The skateboard mesh at the finest resolution (left, in Figure 9) has $nV_0 = 12947$ vertices and 16290 faces. At the coarsest resolution (right, in the same figure) it has $nV_7 = 125$ vertices.

The piping construction has $nV_0 = 18138$ vertices, and after 7 levels of details it is reduced to $nV_0 = 147$ vertices. The first simplification step achieves almost the theoretical bound: from 18138 vertices, the mesh is simplified to 2520 vertices. The original mesh and its approximations are rendered in Figures 13-14.

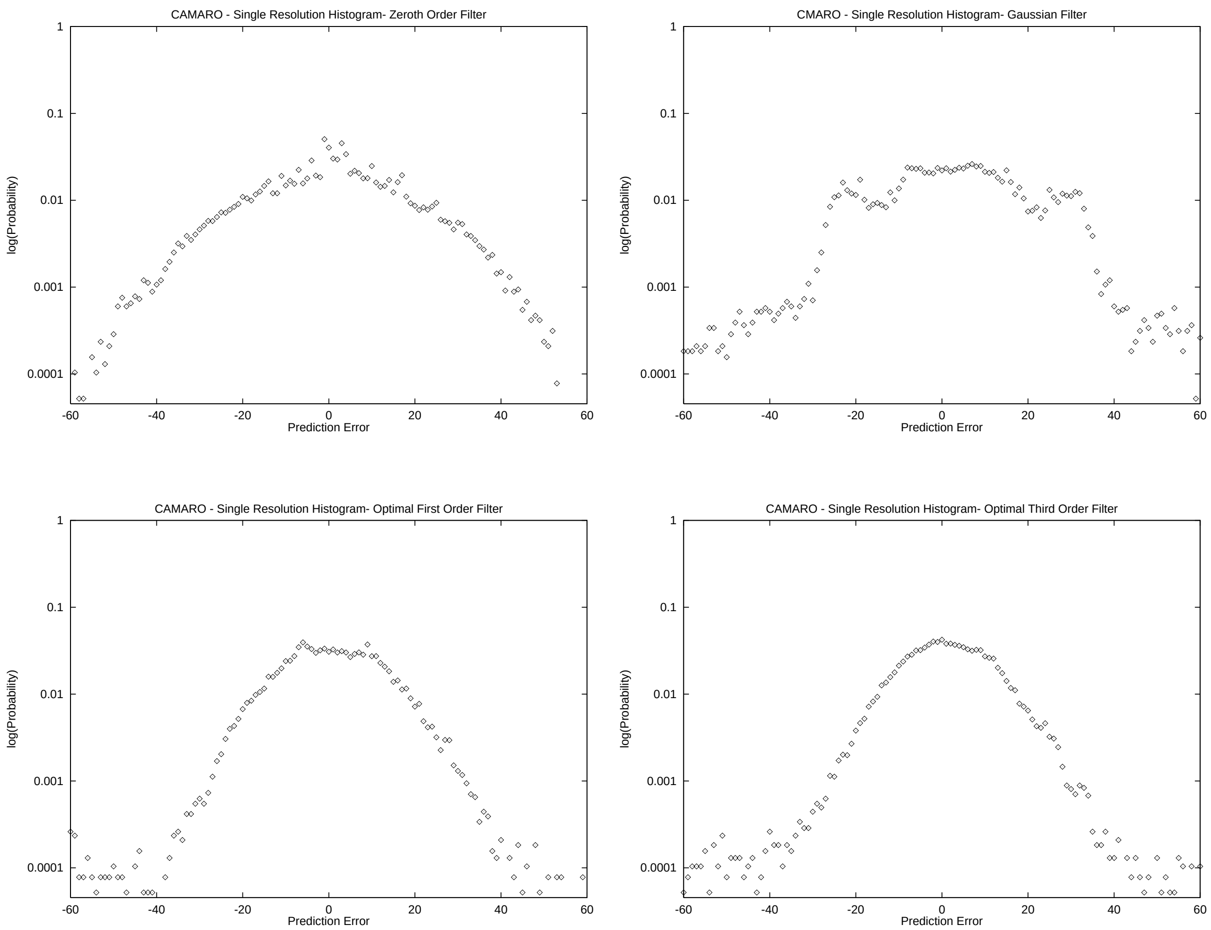

Figure 3: Semilog histograms of the prediction errors associated to the four filters for the Single Resolution scheme applied to the car mesh rendered in Figure 1.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	438	438	438	438
Coefficients	0	0	168	336
Differences	22739	51575	23645	23497
Total (bytes)	23196	52032	24274	24295
Rate (bits/vertex)	15.63	35.07	16.36	16.37

Table 6: Compression ratio results for several filtering schemes applied to the round table mesh rendered in Figure 5.

Filter's Degree	1	2	3	4	5	6	7
bits/vertex	16.36	16.34	16.37	16.50	16.62	16.73	16.88

Table 7: Compression ratios for different filter lengths, in power basis, for the round table.

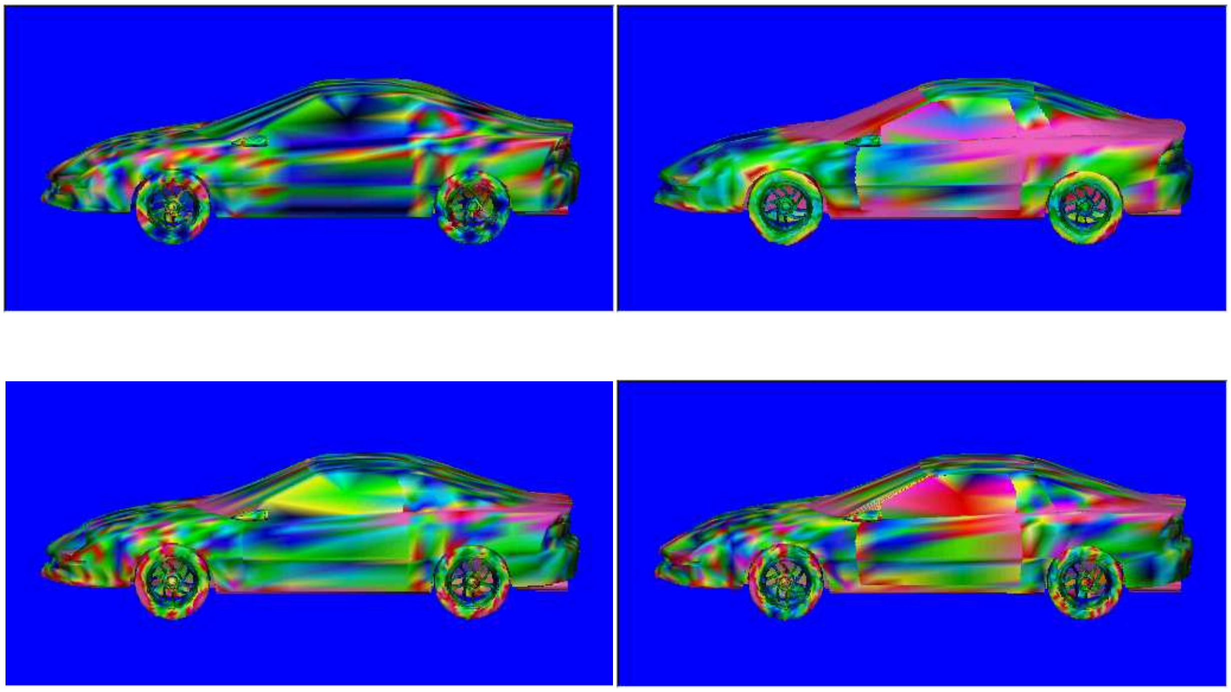

Figure 4: The color plot of the single resolution approximation errors for the four different filters: Zeroth Order Filter (upper left), Gaussian Smoothing (upper right), LS first order (lower left) and LS third order (lower right).

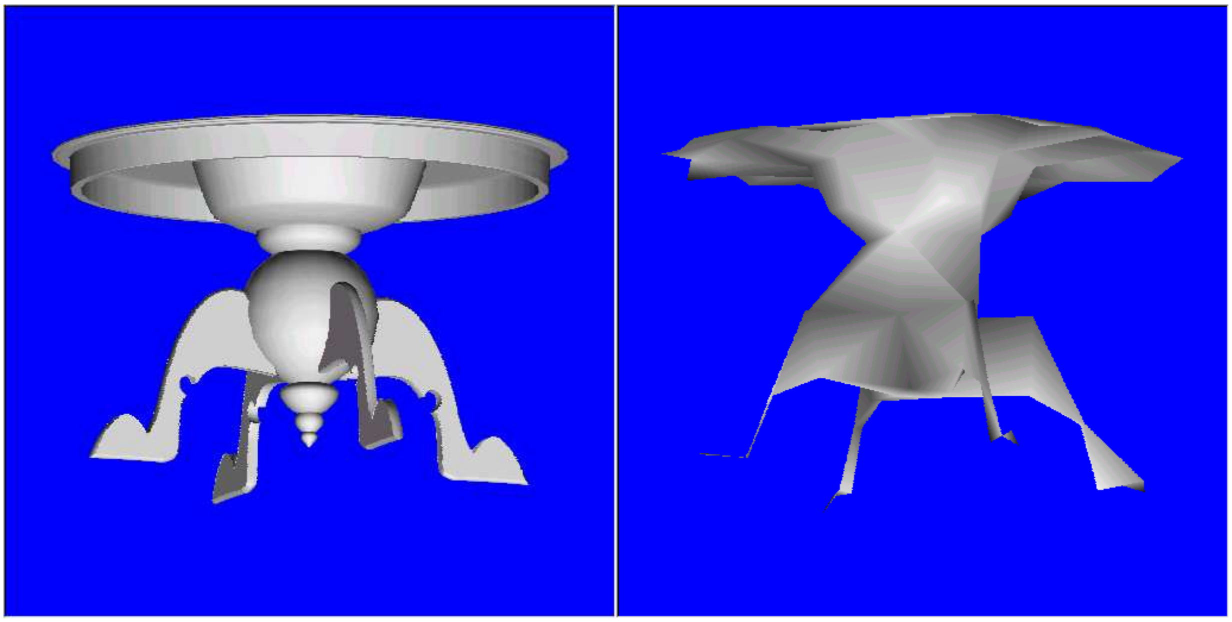

Figure 5: The round table mesh at: the finest resolution (left) and the coarsest resolution (right) after 7 levels of reduction.

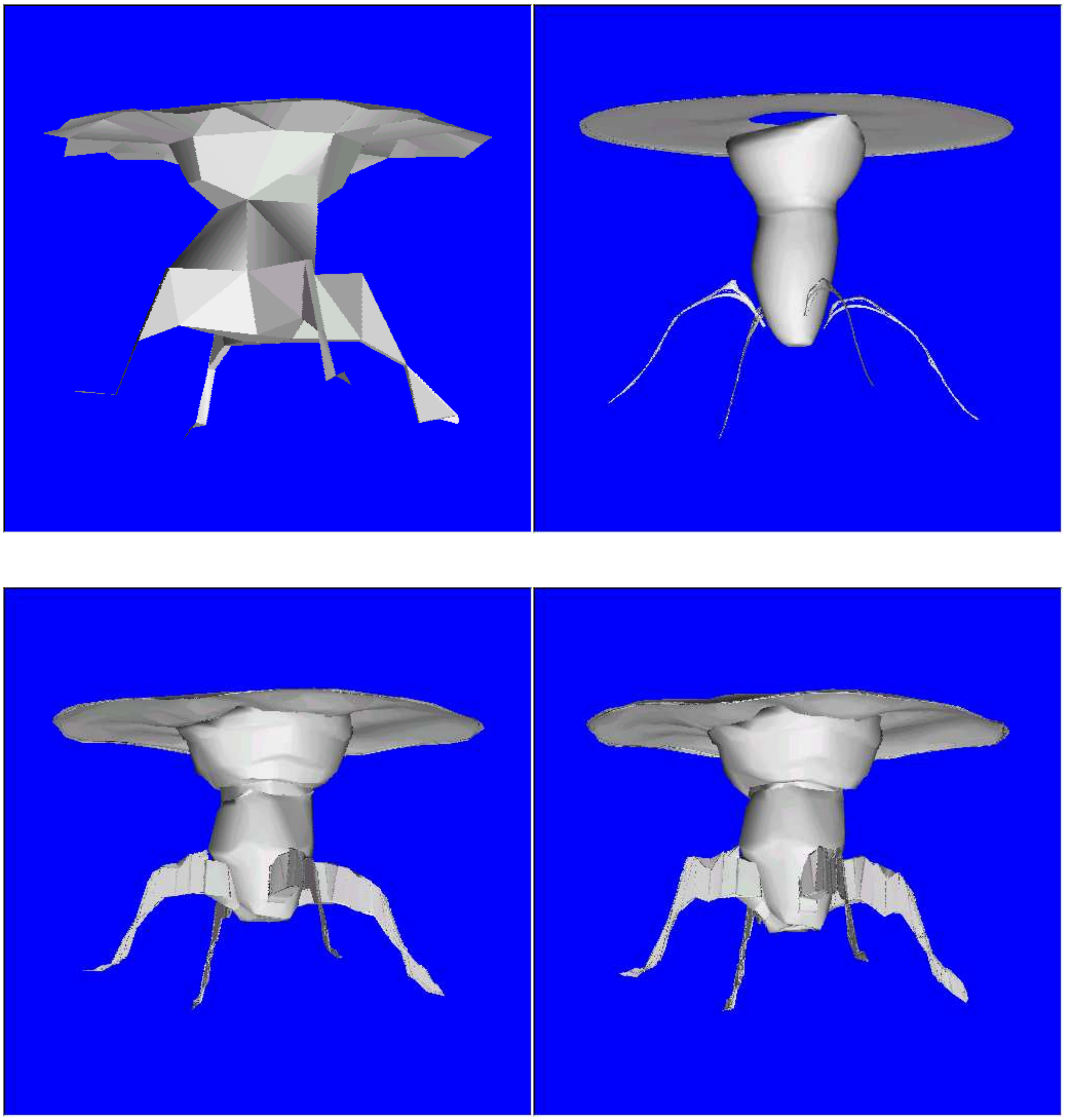

Figure 6: The round table mesh at the finest resolution level when no difference is used and the filtering is performed by: the Zeroth Order Filter (top left), the Gaussian Smoother (top right), the least squares filter of order 1 (bottom left) and the least squares filter of order 3 (bottom right).

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	438	438	438	438
Coefficients	0	0	168	336
Differences	30386	29138	27972	27377
Total (bytes)	30847	29599	28609	28146
Rate (bits/vertex)	20.79	19.95	19.28	18.97
l^2 error/vertex($\cdot 10^{-2}$)	15.85	17.38	11.17	9.97

Table 8: Compression ratios in the Single Resolution implementation of the Variable Length Encoding Algorithm applied to the round table mesh rendered in Figure 5.

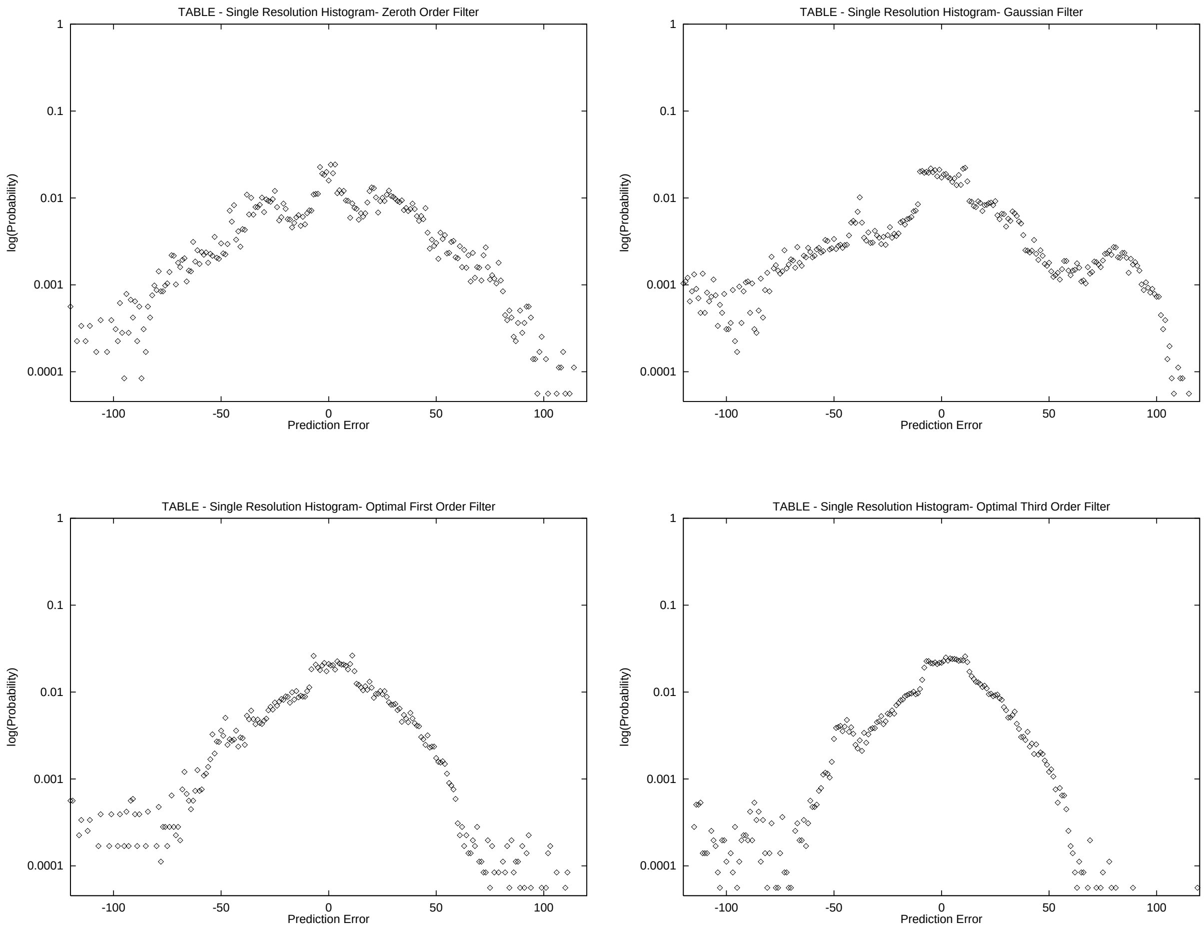

Figure 7: Semilog histograms of the prediction errors associated to the four filters for the Single Resolution scheme applied to the round table rendered in Figure 5.

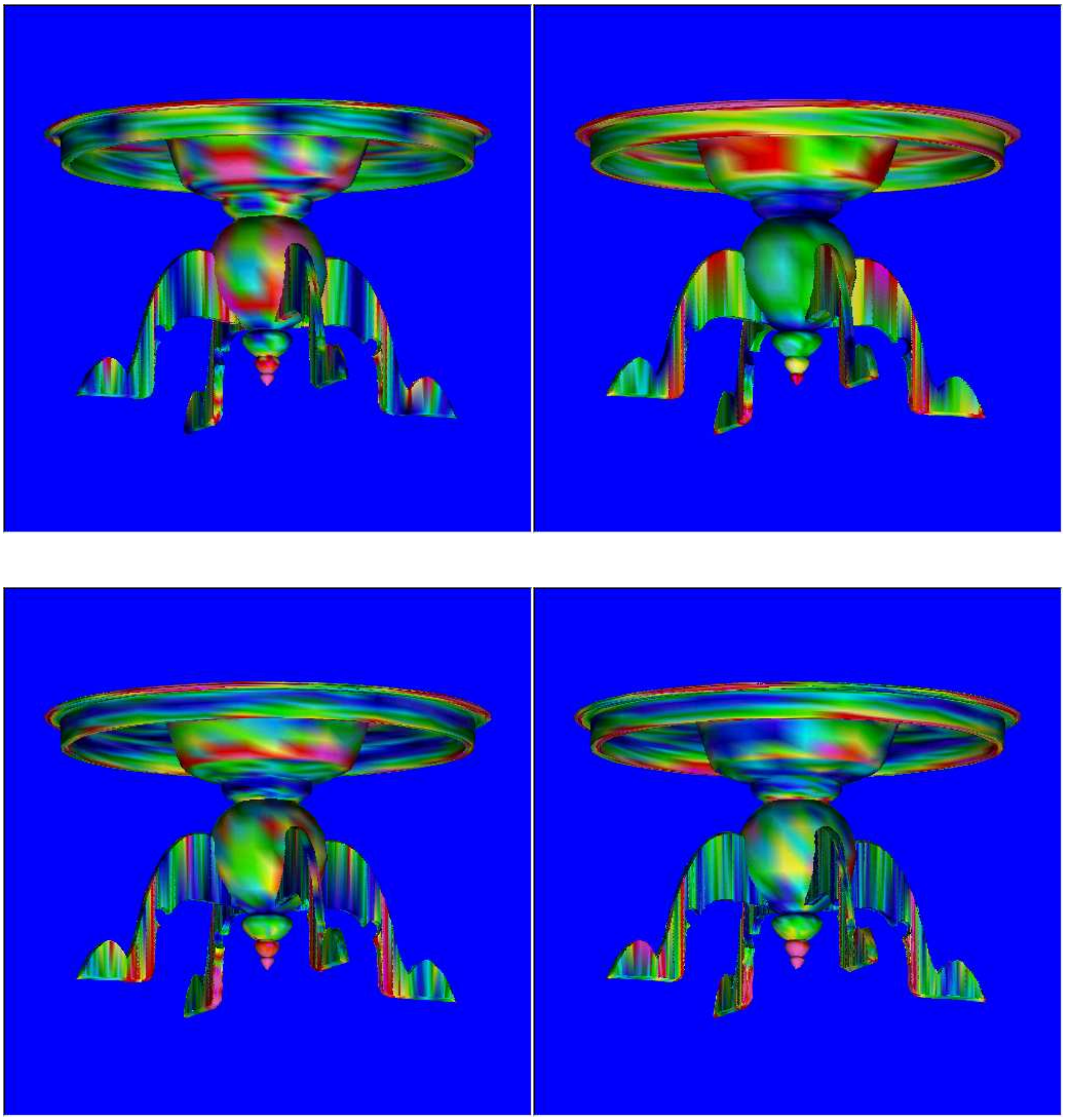

Figure 8: The color plot of the single resolution approximation errors for the four different filters: Zeroth Order Filter (upper left), Gaussian Smoothing (upper right), LS first order (lower left) and LS third order (lower right).

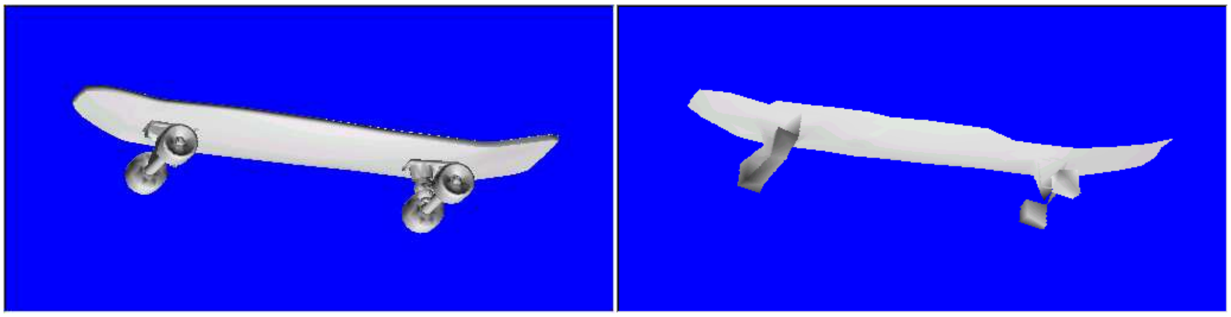

Figure 9: The skateboard mesh at: the finest resolution (left) and the coarsest resolution (right) after 8 levels of reduction.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	444	444	444	444
Coefficients	0	0	168	336
Differences	22735	46444	22627	22443
Total (bytes)	23199	46908	23259	23247
Rate (bits/vertex)	14.33	28.98	14.37	14.36

Table 9: Compression ratio results for several filtering schemes applied to the skateboard rendered in Figure 9.

Filter's Degree	1	2	3	4	5	6	7
bits/vertex	14.37	14.32	14.36	14.45	14.48	14.54	16.01

Table 10: Compression ratios for different filter lengths, in power basis, for the round table.

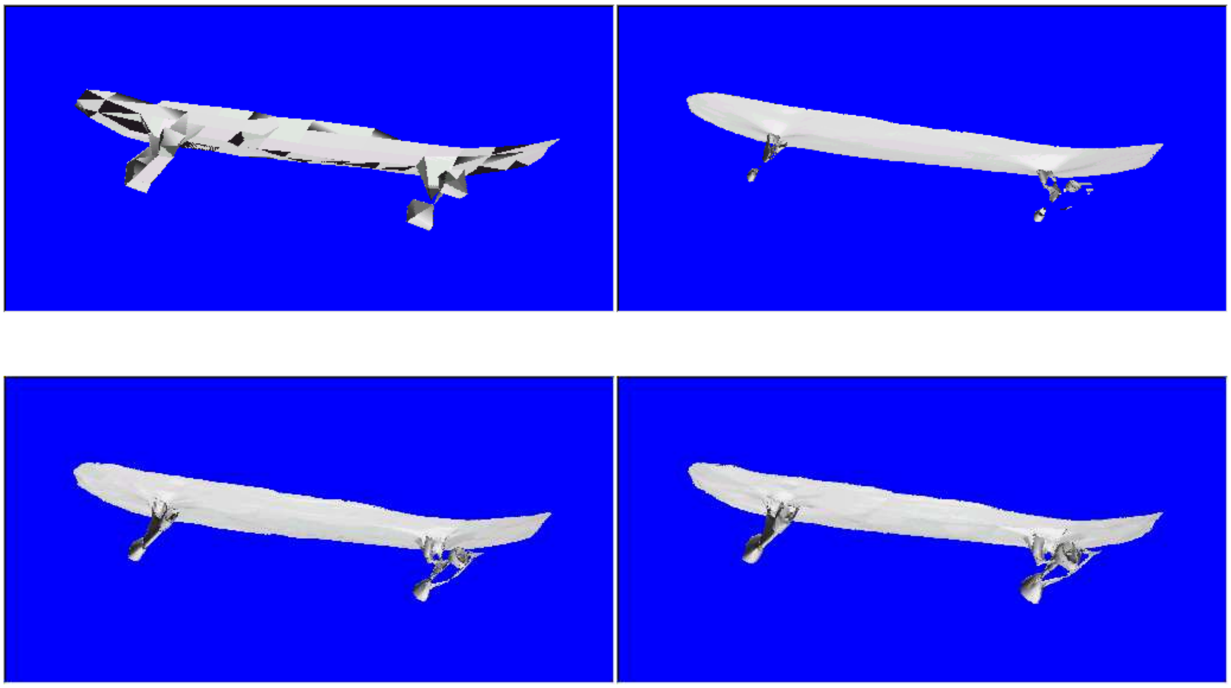

Figure 10: The skateboard mesh at the finest resolution level when no difference is used and the filtering is performed by: the Zeroth Order Filter (top left), the Gaussian Smoother (top right), the least squares filter of order 1 (bottom left) and the least squares filter of order 3 (bottom right).

	Zeroth Order Filter	Gaussian Smoothing	LS Filter of Degree 1	LS Filter of Degree 3
Coarsest mesh	444	444	444	444
Coefficients	0	0	168	336
Differences	28931	27082	26542	26436
Total (bytes)	29397	27549	27184	27250
Rate (bits/vertex)	18.16	17.02	16.80	16.84
l^2 error/vertex($\cdot 10^{-4}$)	43.87	51.83	38.82	36.12

Table 11: Compression ratios in the Single Resolution implementation of the Variable Length Encoding Algorithm applied to the skateboard rendered in Figure 9.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	522	522	522	522
Coefficients	0	0	144	288
Differences	2605	31595	2625	2673
Total (bytes)	3146	32136	3311	3503
Rate (bits/vertex)	1.38	14.17	1.46	1.54

Table 12: Compression ratio results for several filtering schemes applied to the piping construction mesh rendered in Figure 13.

Filter's Degree	1	2	3	4	5	6	7
bits/vertex	1.46	1.50	1.54	1.58	1.61	1.65	1.69

Table 13: Compression ratios for different filter lengths, in power basis, for the piping construction.

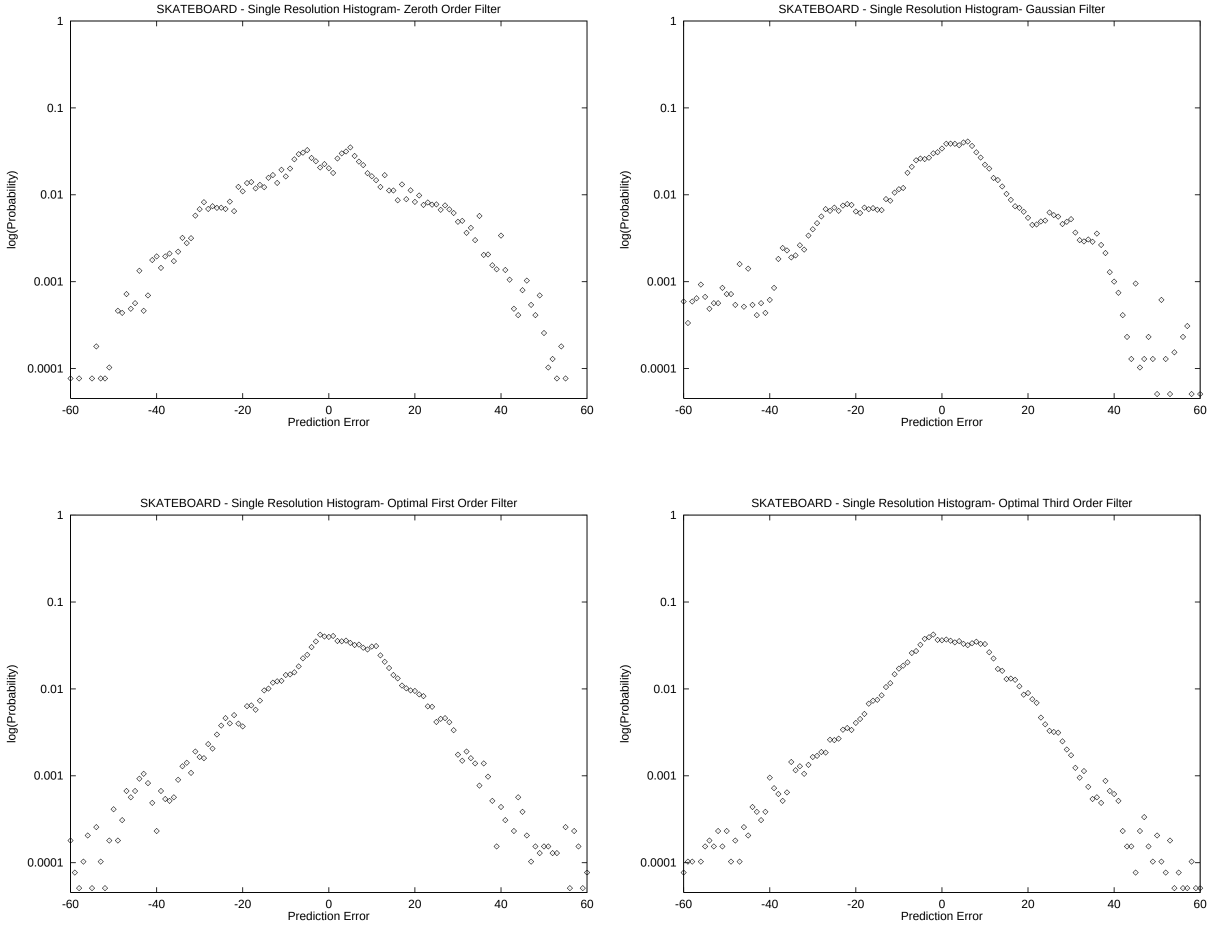

Figure 11: Semilog histograms of the prediction errors associated to the four filters for the Single Resolution scheme applied to the mesh rendered in Figure 9.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	522	522	522	522
Coefficients	0	0	168	336
Differences	19160	41278	21009	21573
Total (bytes)	19704	41823	21701	22458
Rate (bits/vertex)	8.69	18.44	9.57	9.90
l^2 error/vertex$(\cdot 10^{-2})$	1.21	22.96	1.20	1.20

Table 14: Compression ratios in the Single Resolution implementation of the Variable Length Encoding Algorithm applied to the piping construction rendered in Figure 13.

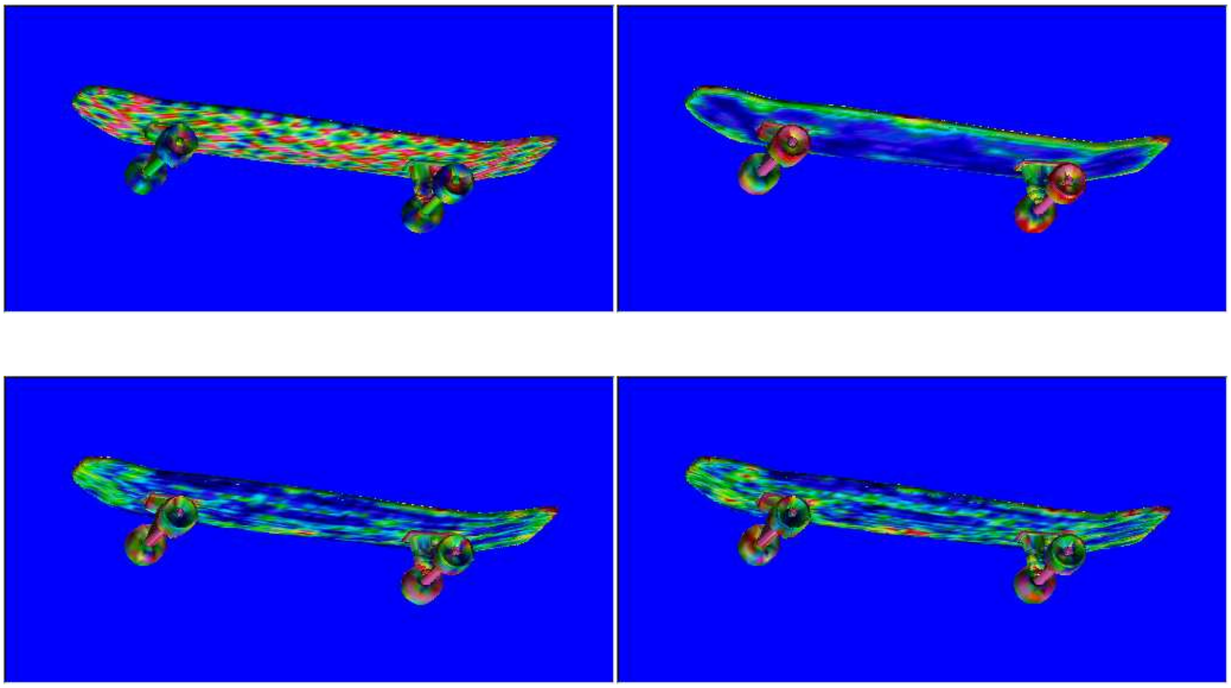

Figure 12: The color plot of the single resolution approximation errors for the four different filters: Zeroth Order Filter (upper left), Gaussian Smoothing (upper right), LS first order (lower left) and LS third order (lower right).

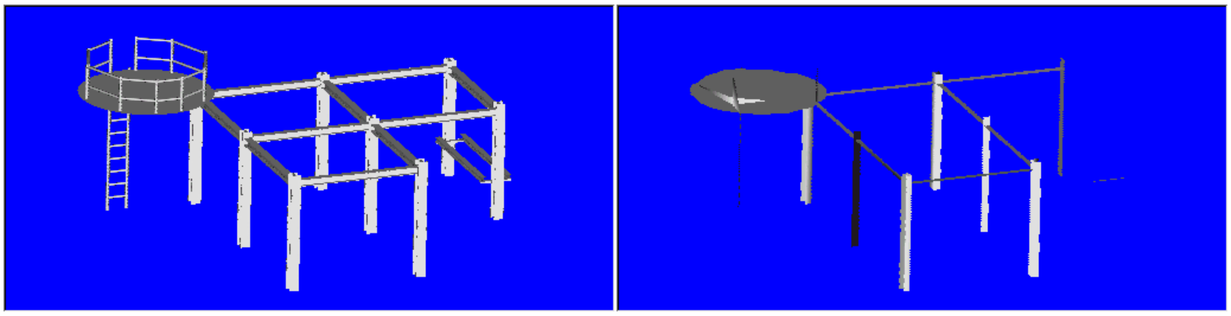

Figure 13: The piping mesh at: the finest resolution (left) and the coarsest resolution (right) after 7 levels of reduction.

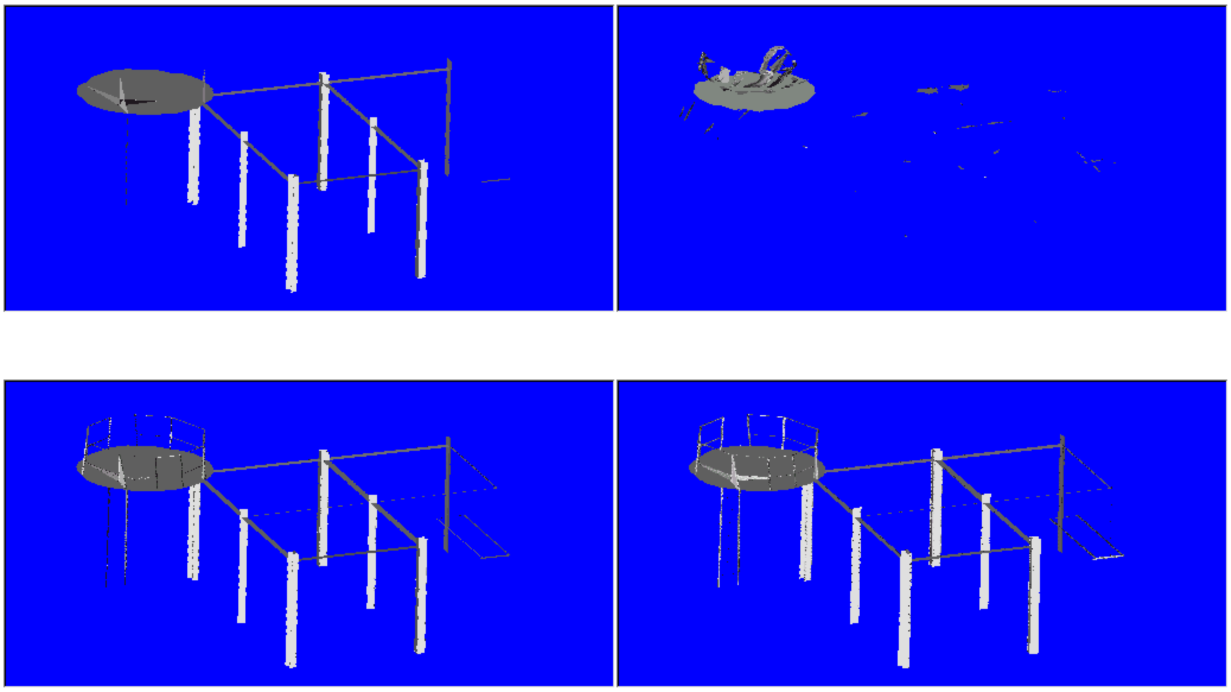

Figure 14: The piping mesh at the finest resolution level when no difference is used and the filtering is performed by: the Zeroth Order Filter (top left), the Gaussian Smoother (top right), the least squares filter of order 1 (bottom left) and the least squares filter of order 3 (bottom right).

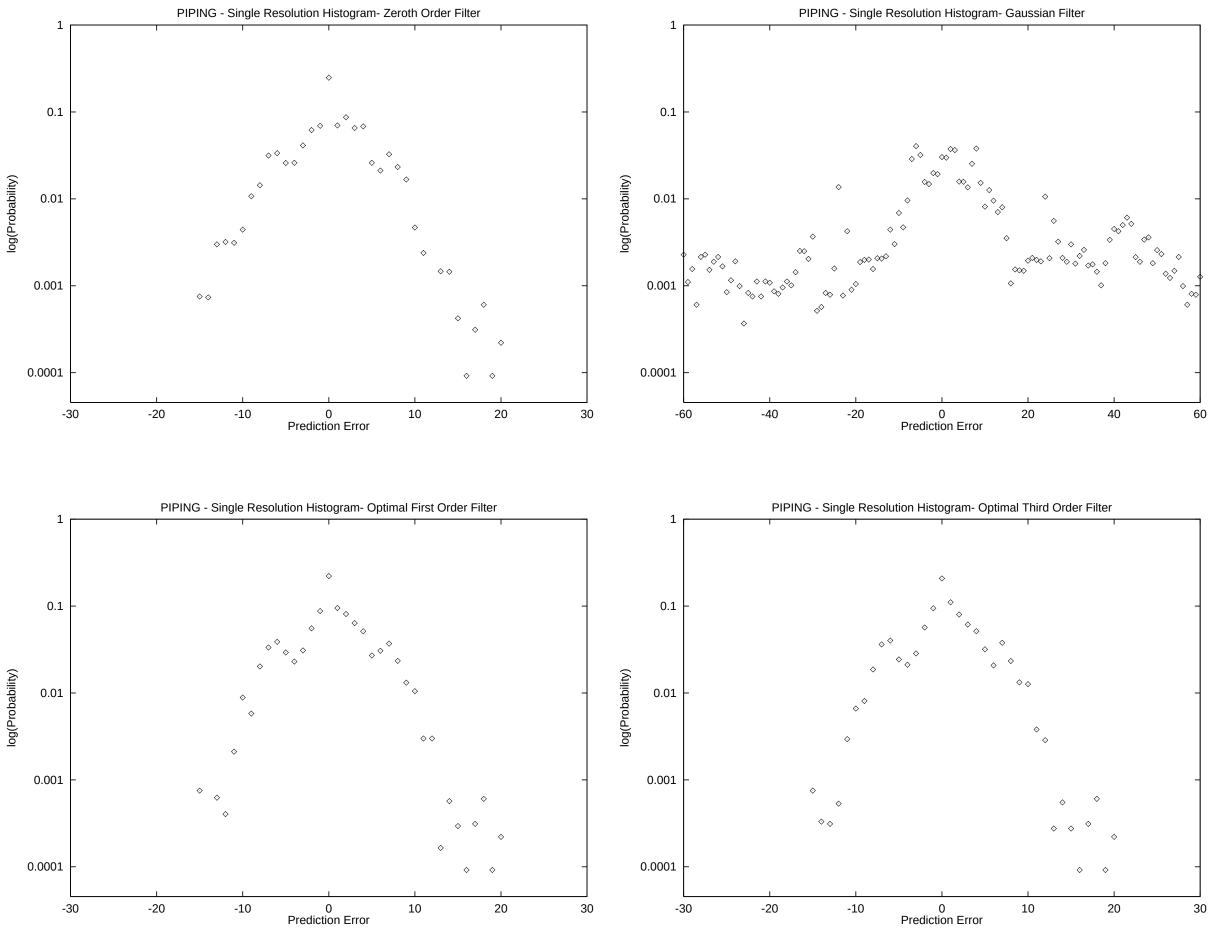

Figure 15: Semilog histograms of the prediction errors associated to the four filters for the Single Resolution scheme applied to the mesh rendered in Figure 13.

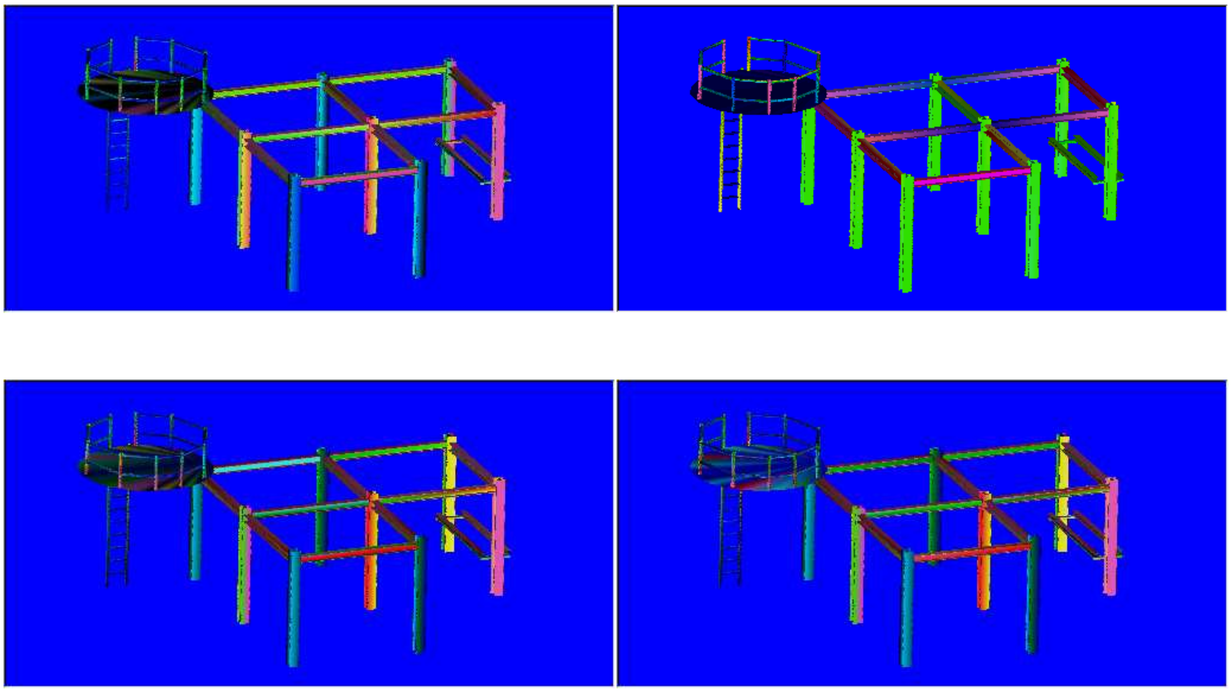

Figure 16: The color plot of the single resolution approximation errors for the four different filters: Zeroth Order Filter (upper left), Gaussian Smoothing (upper right), LS first order (lower left) and LS third order (lower right).

The last mesh we discuss is somewhat different to the other. It is a sphere of $nV_0 = 10242$ vertices and 20480 faces that reduces after 4 levels to $nV_3 = 224$ vertices. The striking difference is the compression ratio of the Zeroth Order filter: it is the worst of all the filters we checked. Even the Gaussian filter fares better than this filter. Snapshots of the approximated meshes are pictured in Figures 17-18. The mesh used is non-manifold but this is not a problem for the geometry encoder. The histograms shown in Figure 19 are in accordance with the rate results presented in Table 17: the narrower the distribution the better the rate. Note also how well a power-type low fits the 3^{rd} order filtered distribution.

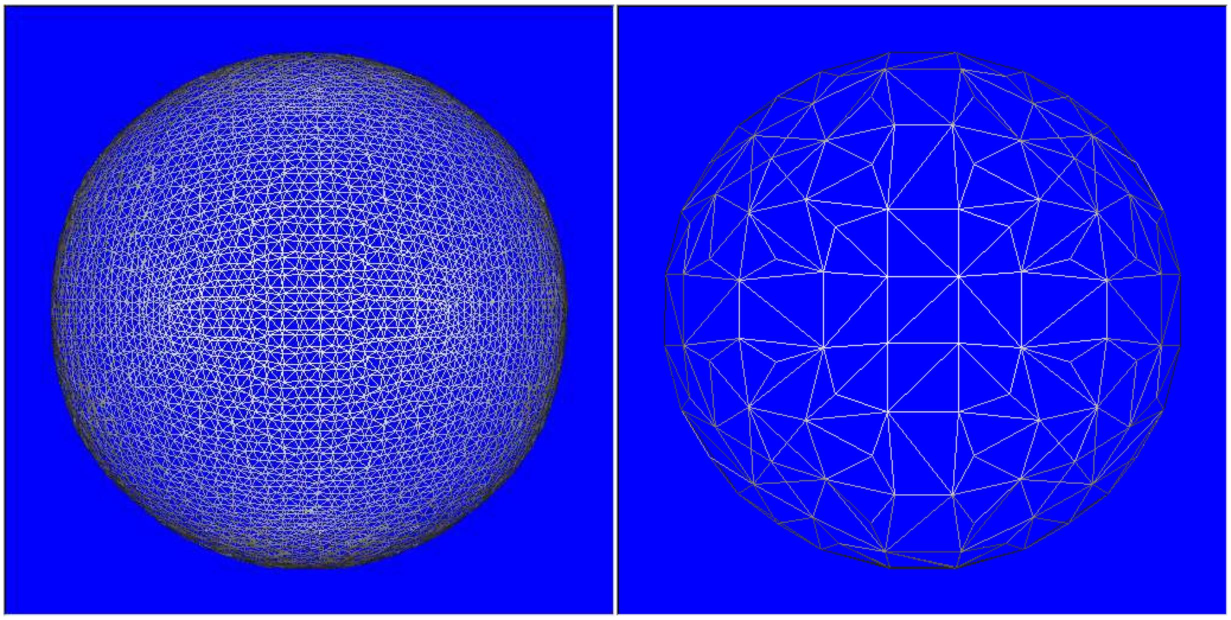

Figure 17: The sphere at: the finest resolution (left) and the coarsest resolution (right) after 4 levels of reduction.

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	881	881	881	881
Coefficients	0	0	72	144
Differences	29770	22614	19762	13395
Total (bytes)	30673	23518	20738	14440
Rate (bits/vertex)	23.96	18.37	16.20	11.28

Table 15: Compression ratio results for several filtering schemes applied to the sphere rendered in Figure 17.

Filter's Degree	1	2	3	4	5	6	7
bits/vertex	16.20	12.90	11.28	10.59	10.36	10.42	10.69

Table 16: Compression ratios for different filter lengths, in power basis, for sphere.

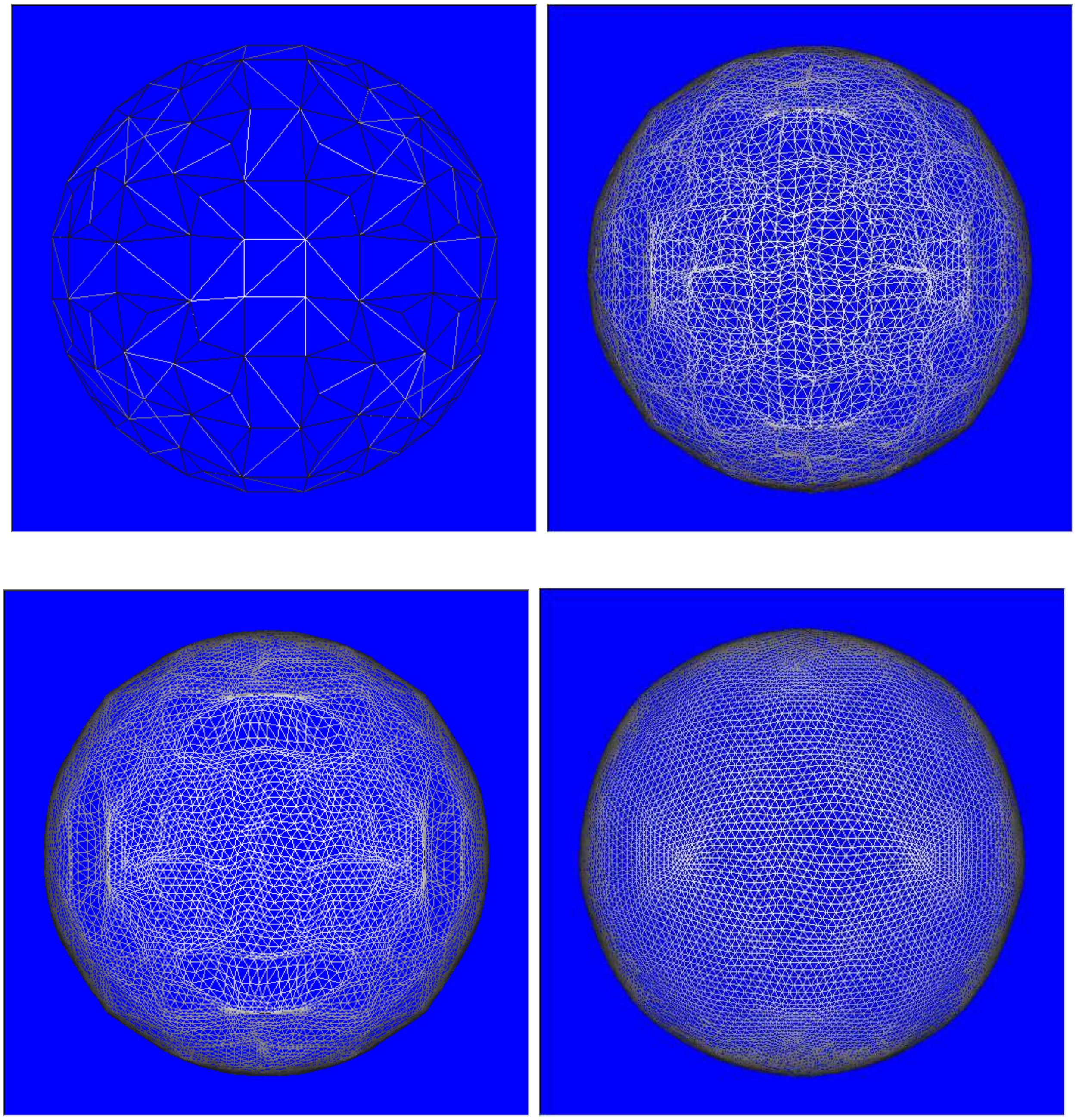

Figure 18: The sphere at the finest resolution level when no difference is used and the filtering is performed by: the Zeroth Order Filter (top left), the Gaussian Smoother (top right), the least squares filter of order 1 (bottom left) and the least squares filter of order 3 (bottom right).

Filter	Zeroth Order	Gaussian Smoothing	LS of Degree 1	LS of Degree 3
Coarsest mesh	881	881	881	881
Coefficients	0	0	168	336
Differences	28904	22152	20904	17920
Total (bytes)	29806	23055	21885	18971
Rate (bits/vertex)	23.28	18.00	17.09	14.82
l^2 error/vertex$(\cdot 10^{-4})$	6.85	2.36	1.81	1.11

Table 17: Compression ratios in the Single Resolution implementation of the Variable Length Encoding Algorithm applied to the sphere rendered in Figure 17.

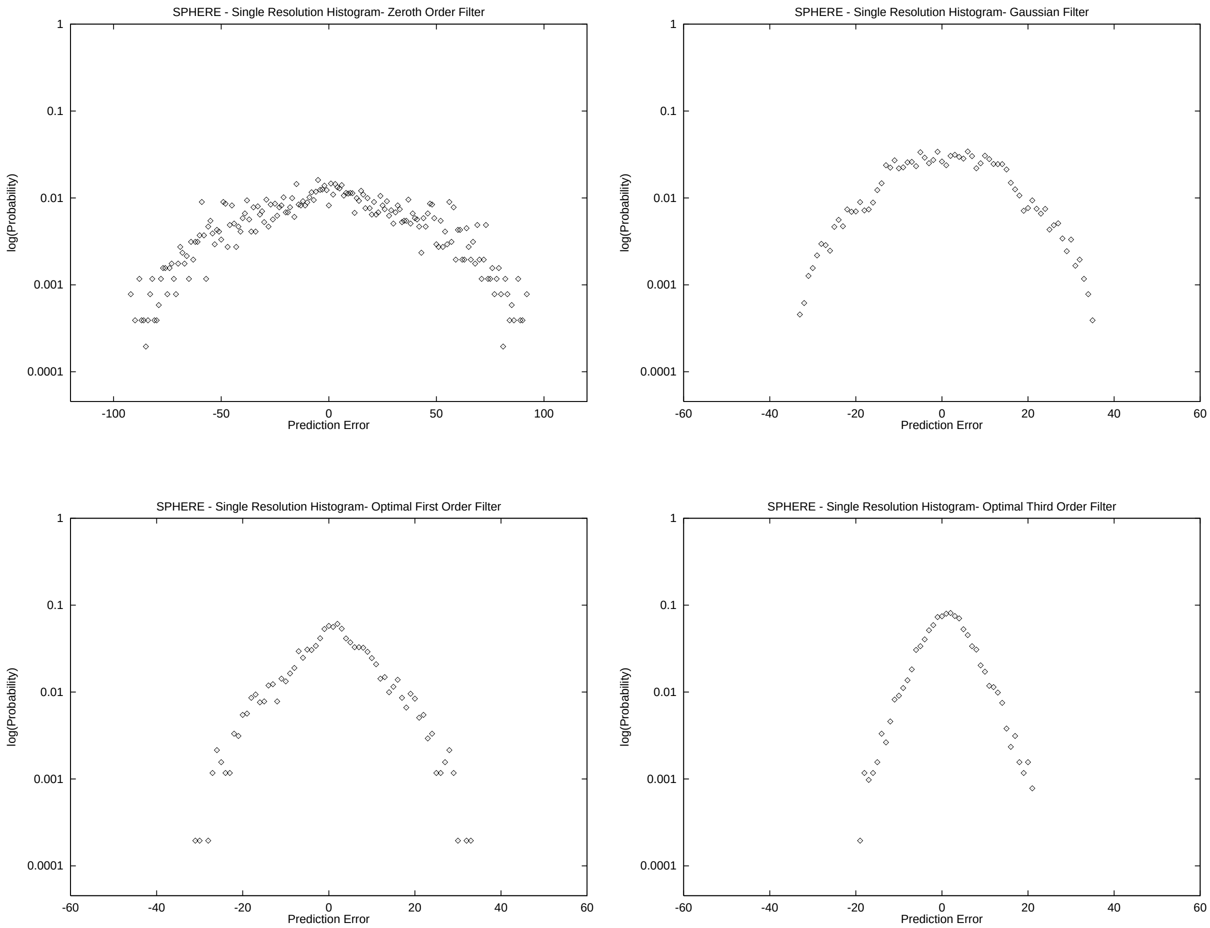

Figure 19: Semilog histograms of the prediction errors associated to the four filters for the Single Resolution scheme applied to the sphere mesh rendered in Figure 17.

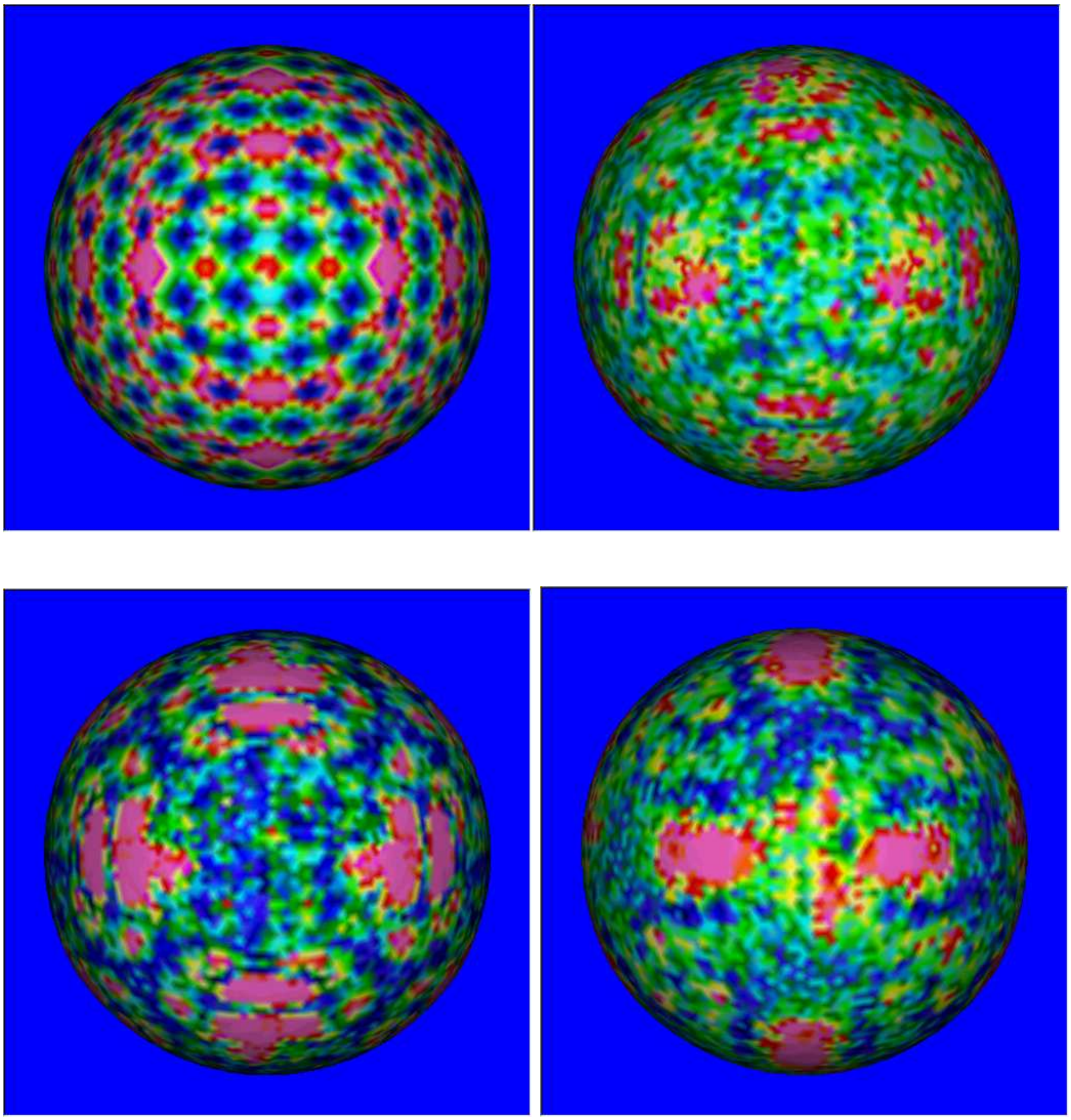

Figure 20: The color plot of the single resolution approximation errors for the four different filters: Zeroth Order Filter (upper left), Gaussian Smoothing (upper right), LS first order (lower left) and LS third order (lower right).

These examples show that in terms of compression ratio, the Zeroth Order Filter compresses best the irregular and less smooth meshes, whereas higher order filter are better for smoother and more regular meshes. However, in terms of accuracy and robustness, the higher order filters perform much better than its main "competitor", the Zeroth Order Filter. Note, except for highly regular meshes (like sphere, for instance), relatively low order filters are optimal. The range [1..5] seems enough for most of the encoding schemes.

6 Conclusions

In this paper we study the 3D geometry filtering using the discrete Laplace operator. We next apply the filtering technique to Multi Resolution Analysis where the original mesh is converted into a sequence of successive refinements. Based on the coarser resolution mesh, the finer resolution mesh is predicted using an extension map followed by filtering. At each level, the coordinate vectors are filtered separately using different filters. These filters are optimizers of some prediction error norm. Thus the geometry of a sequence of successively refined meshes is encoded in the following format: first the coarsest resolution mesh geometry and next for each successive level, the filters coefficients and prediction errors. The connectivity information is supposed known at each level separately.

Next we study several desirable properties of any encoding scheme, finding for each one the appropriate criterion to be optimized. Thus for a better accuracy of the predicted mesh when no difference is available, the filter coefficients should minimize the l^∞-norm of the prediction errors. For robustness, as understood in signal processing theory, the filters should minimize the l^2-norm of the differences. The third property, the compression rate, is maximized when the l^∞-norm is replaced by a l^α-norm with α usually between 1 and 2, depending on the prediction error's p.d.f. Thus, if the differences are Laplace distributed, the l^1-norm should be minimized, whereas if they are Gaussian, then the l^2-norm should be used. In any case, each of the three extreme cases (l^∞, l^2 or l^1) can be solved exactly. The l^2-norm case is the simplest and reltively computational inexpensive, and is solved by a linear system. The other two cases turn into linear programming problems which are very computational expensive to solve.

These theoretical results are next applied to concrete examples. In general for large, non-smooth and irregular meshes the Zeroth Order Filtering scheme yields the best compression ratio, but the poorest accuracy or, for the same reason, robustness. Instead, by paying a small price in the compression ratio, a least square filter give a better rendering accuracy and superior robustness. At the other end of the scale, for very smooth and regular meshes, the Gaussian filter (which in general behaves very poorly) gives a better compression ratio than the Zeroth Order filter.

The Basic Encoding Algorithm can be modified to allow a variable structure. The user can choose for what levels the differences are encoded and, by choosing a limit case, only the highest resolution level errors are encoded. Thus the MRA scheme becomes a Single Resolution encoding scheme. Examples in terms of accuracy and compression ratio are shown in the Examples section.

The novelty of this study consists in using linear filter in Multi Resolution encoding schemes and finding appropriate optimization criteria for specific compression or rendering properties. We hope this compression scheme will prove effective in Progressive Transmission protocols as MPEG4.

References

[Al&all88] H.Alt, K.Mehlhorn, H.Wagener, E.Welzl, *Congruence, similarity and symmetries of geometric objects*, Discrete Comp.Geom. **3**, 237–256 (1988)

[DaGr76] *Data Compression*, Ed. by L.D.Davisson and R.M.Gray, Dowden, Hutchinson & Ross Inc. (1976)

[DeMeScBa99] M.Desbrun, M.Meyer, P.Schröder, A.H.Barr, *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow*, SIGGRAPH 1999 (to appear)

[Eck&all95] M.Eck, T.DeRose, T.Duchamp, H.Hope, M.Lounsbery, W.Stuetzle, *Multiresolution Analysis of Arbitrary Meshes*, SIGGRAPH Conf.Proc. 1995, 173–182

[FrInSp79] S.H.Friedberg, A.J.Insel, L.E.Spencer, *Linear Algebra*, Prentice-Hall 1979

[GuSt98] S.Gumhold, W.Strasser, *Real Time Compression of Triangle Mesh Connectivity*, SIGGRAPH 1998, 133–140

[GuSwSc99] I.Guskov, W.Sweldens, P.Schröder, *Multiresolution Signal Processing for Meshes*, SIGGRAPH 1999,

[Hart98] E.Hartmann, *A marching method for the triangulation of surfaces*, Visual Computer **14**, 95–108 (1998)

[Hoppe96] H.Hoppe, *Progressive Meshes*, SIGGRAPH 1996, 99–108

[LiKuo98] J.Li, C.-C.J.Kuo, *Progressive Coding of 3-D Graphics Models*, Proceedings of IEEE (Special Issue on Multimedia Signal Processing) **86**, no.6 June 1998, 1052–1063

[Lee&all98] A.W.F.Lee, W.Sweldens, P.Schröder, L.Cowsar, D.Dobkin, *MAPS: Multiresolution Adaptive Parametrization of Surfaces*, SIGGRAPH Proceedings, 1998, 95–104

[MaYaVe93] J.Maillot, H.Yahia, A.Verroust, *Interactive Texture Mapping*, SIGGRAPH Proceedings, 1993, 27–34

[PaRo99] P.Pajarola, J.Rossignac, *Compressed Progressive Meshes*, Technical Report GIT-GVU-99-05, GVU Center, Georgia Institute of Technology, 1999

[PeMi93] B.P.Pennebaker, J.L.Mitchell, *JPEG, Still Image Compression Standard*, Von Nostrand Reinhold, 1993

[PoHo97] J.Popocić, H.Hoppe, *Progressive Simplicial Complexes*, SIGGRAPH 1997, 217–224

[Ross99] J.Rossignac, *Edgebreaker: Connectivity Compression for Triangle Meshes*, IEEE Trans.on Vis.Comp.Graph. **5**, no.1 (1999), 47–61

[Taubin95] G.Taubin, *A Signal Processing Approach to Fair Surface Design*, Computer Graphics Proc., Annual Conference Series 1995, 351–358

[TaZhGo96] G.Taubin, T.Zhang, G.Golub, *Optimal Surface Smoothing as Filter Design*, IBM Research report RC-20404 1996

[TaGuHoLa98] G.Taubin, A.Guézier, W.Horn, F.Lazarus, *Progressive Forest Split Compression*, SIGGRAPH Proceedings , 1998,

[TaHoBo99] G.Taubin, W.Horn, P.Borrel, *Compression and Transmission of Multi-Resolution Clustered Meshes*, IBM Research report RC-21398, February 1999

[TaRo98] G.Taubin, J.Rossignac, *Geometry Compression through Topological Surgery*, ACM Transaction on Graphics **17**, no.2 April 1998, 84–115

[ZiTr90] R.E.Ziemer, W.H.Tranter, *Principles of Communications - System, Modulation, and Noise*, Houghton Mifflin Comp. 1990