Reducing and Filtering Point Clouds With Enhanced Vector Quantization

Stefano Ferrari, Giancarlo Ferrigno, Vincenzo Piuri, *Fellow, IEEE*, and N. Alberto Borghese, *Member, IEEE*

***Abstract*—Modern scanners are able to deliver huge quantities of three-dimensional (3-D) data points sampled on an object's surface, in a short time. These data have to be filtered and their cardinality reduced to come up with a mesh manageable at interactive rates. We introduce here a novel procedure to accomplish these two tasks, which is based on an optimized version of soft vector quantization (VQ). The resulting technique has been termed enhanced vector quantization (EVQ) since it introduces several improvements with respect to the classical soft VQ approaches. These are based on computationally expensive iterative optimization; local computation is introduced here, by means of an adequate partitioning of the data space called hyperbox (HB), to reduce the computational time so as to be linear in the number of data points N, saving more than 80% of time in real applications. Moreover, the algorithm can be fully parallelized, thus leading to an implementation that is sublinear in N. The voxel side and the other parameters are automatically determined from data distribution on the basis of the Zador's criterion. This makes the algorithm completely automatic. Because the only parameter to be specified is the compression rate, the procedure is suitable even for nontrained users. Results obtained in reconstructing faces of both humans and puppets as well as artifacts from point clouds publicly available on the web are reported and discussed, in comparison with other methods available in the literature. EVQ has been conceived as a general procedure, suited for VQ applications with large data sets whose data space has relatively low dimensionality.**

***Index Terms*—Clustering, filtering, point-clouds reduction, reconstruction error, space partitioning, three-dimensional (3-D) scanner.**

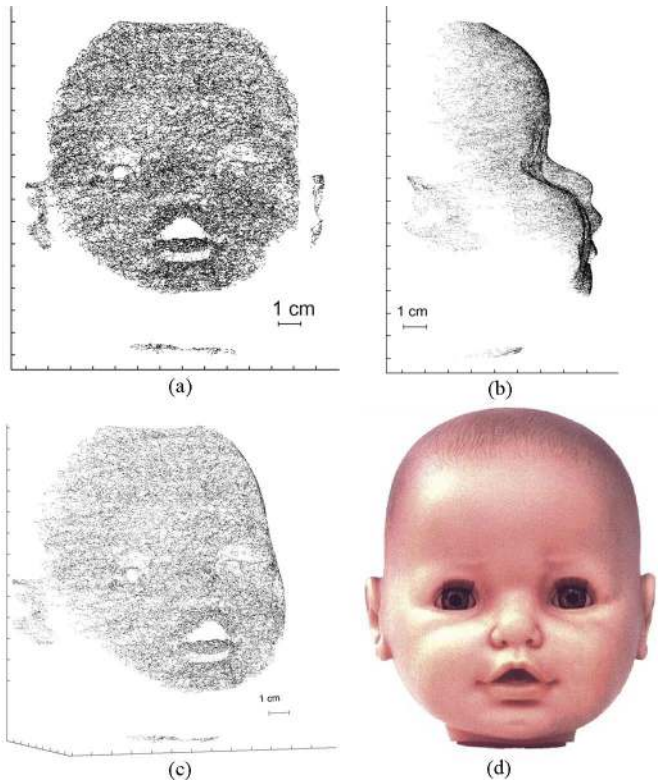

Fig. 1. Set made up of $N = 100\,000$ points sampled over the doll face in panel (d) is plotted in panels (a), (b), and (c). The points were sampled with a home-made scanner inspired by [2] and are not uniformly distributed over the surface.

I. Introduction

THE use of digital models of real-life artifacts is becoming a fundamental tool in many fields ranging from virtual architecture to image processing, from medicine to reverse engineering, three-dimensional (3-D) fax and videoconferencing. A common approach is to create these models from a set of 3-D points sampled over the artifacts' surface through 3-D digitizers [1], [2] (cf. Fig. 1). These points are then transformed into a continuous surface in the form of a triangular mesh (cf. Fig. 2), which is the *de facto* standard for handling 3-D surfaces [3].

Manuscript received July 13, 2005; revised June 19, 2006. This work was supported in part by the Italian CNR-MIUR under Grant 449/97: "Robocare."

S. Ferrari and V. Piuri are with the Department of Information Technologies, University of Milano, Crema (CR) 26013, Italy (e-mail: ferrari@dti.unimi.it; piuri@dti.unimi.it).

G. Ferrigno is with the Bioengineering Department, Politecnico di Milano, Milano 20133, Italy (e-mail: ferrigno@biomed.polimi.it).

N. A. Borghese is with the AIS-Lab, Department of Computer Science, University of Milano, Milano 20135, Italy (e-mail: borghese@dsi.unimi.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2006.886854

The solutions adopted by real scanning systems are based on pipeline processing [1], [4]–[6] (cf. Fig. 3). Because the whole shape of complex real objects cannot be captured in a single acquisition session, different sets of 3-D point clouds are acquired in several acquisition sessions, each taken with a different sensor location and at a different orientation. Each cloud can be made up of several hundred thousand data points and represents part of the scanned object. These clouds are first registered (rototranslated) so that they are represented in the same Cartesian reference system, obtaining a single very large cloud of data points, from which the final model can be constructed (e.g., [4] and [6]–[9]).

The transformation of the cloud of 3-D points into a 3-D mesh is a critical task. The simple connection of the points would produce a rather wavy surface, because measurement noise at the data points is transferred into the 3-D mesh. A model like the one in Fig. 2(b) would be useless. Moreover, this approach produces a very large number of triangles, many more than required by most applications. Therefore, a stage

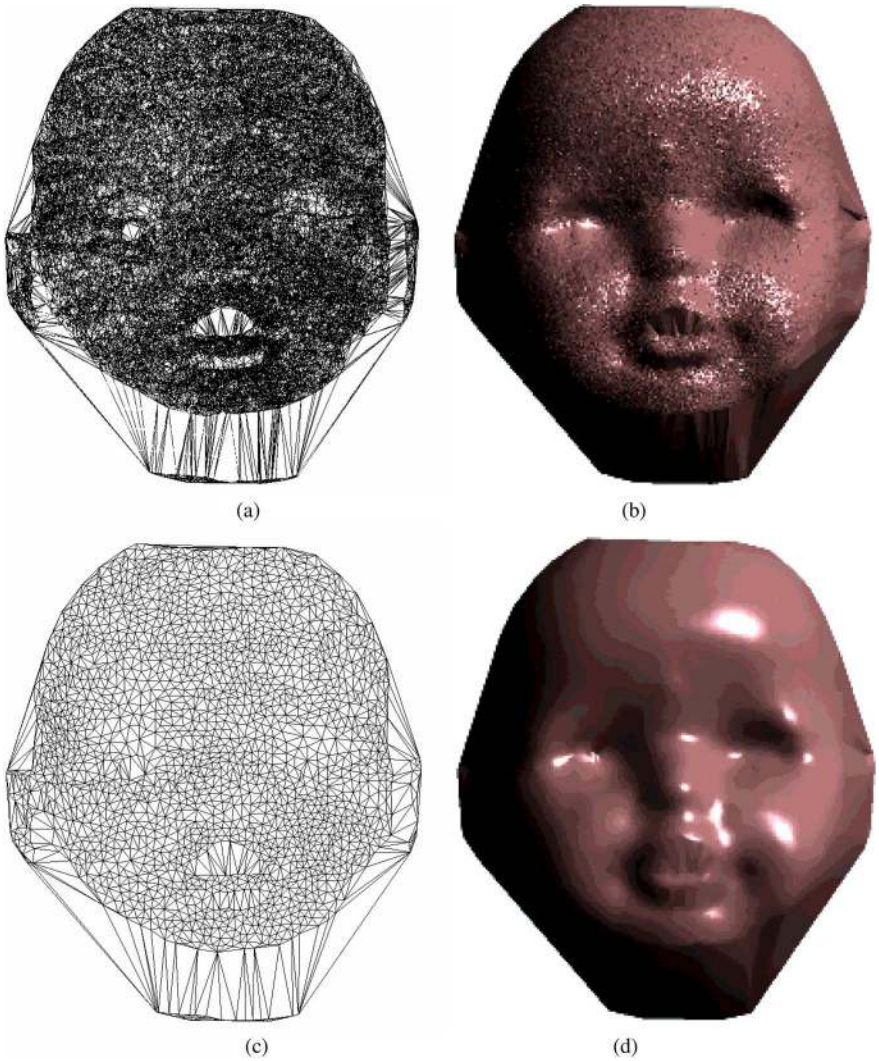

Fig. 2. (a) Mesh constructed with a Delaunay triangulation [3] from the set of N sampled points of Fig. 1 ($\approx$200 000 triangles), is plotted in wireframe in panel (a) and Gouraud shaded in panel (b). High-frequency noise makes the model useless and should be removed. Moreover, the number of triangles is far greater than needed to represent this model, for many applications. This is made evident in panels (c) and (d) were the same surface is represented with the set of M =2000 RVs ($\approx$4000 triangles) obtained by processing the data set in Fig. 1 with EVQ. Noise was removed and the size of the data set was reduced to 2%. The parameters used were: $\tau = 0.02$, $T_{\max} = 5N = 500\,000$, and $L = 12.20$ mm, which produce $11 \times 13 \times 8$ boxes, $\varepsilon_i = 0.344$, $\eta = 0.2$, and $M_q = 9.9$.

devoted to filtering and data reduction is required; this can be viewed as a nonlinear optimization problem [4], [6], [8], [10]; cf. also [9], [11], and [12].

Two main approaches have been proposed in the literature. In the first approach [Fig. 3(a)], a partial mesh is constructed for each set of data points. These partial meshes are then zippered together to obtain a single large mesh. A few algorithms, which incorporate filtering, have been proposed for constructing the partial 3-D meshes. Most of these are based on "warping" [8], a two-dimensional (2-D) manifold, like a lattice, e.g., a Kohonen map, [10], [13], [14] or a general mesh [9], [15]. The quality of the result depends heavily on the initial configuration of the mesh and on the degree of similarity between the mesh and the surface topology. Moreover, these approaches are extremely time-consuming (e.g., [10] requires more than 40 min to create a 52×52 mesh from $\approx$30 000 range data points on a Pentium II 350-MHz machine). A different strategy is based on fitting a set of small piecewise linear [7] or nonlinear [16] patches, with continuity constraints (cf. also [17]). Construction time is several hundred minutes on a HP 735, 105-MHz machine. Here, the size of the patch is critical in achieving a reliable reconstruction, because it must be adapted to the data's density and local spatial frequency (cf. also [18]). Once a 3-D mesh has been constructed, a mesh simplification stage may follow, so as to reduce the number of vertices and faces according to geometric/topological criteria. The computational time of these algorithms varies greatly, the fastest one reducing a mesh of $\approx$75 000 vertices in $\approx$120 s with a compression rate of 2% [19, Table IV], on an SGI Indigo2, R4400, 250-MHz machine. Overall, this two-stage pipeline [Fig. 3(a)] requires heavy memory use because both the data points and the mesh connectivity should be stored for each scan and because out-of-core techniques, which require that subsets of data be resident in main memory, have to be employed [20], [21].

The second approach uses volumetric techniques [22] [Fig. 3(b)]. It is based on the analysis of the entire set of all the

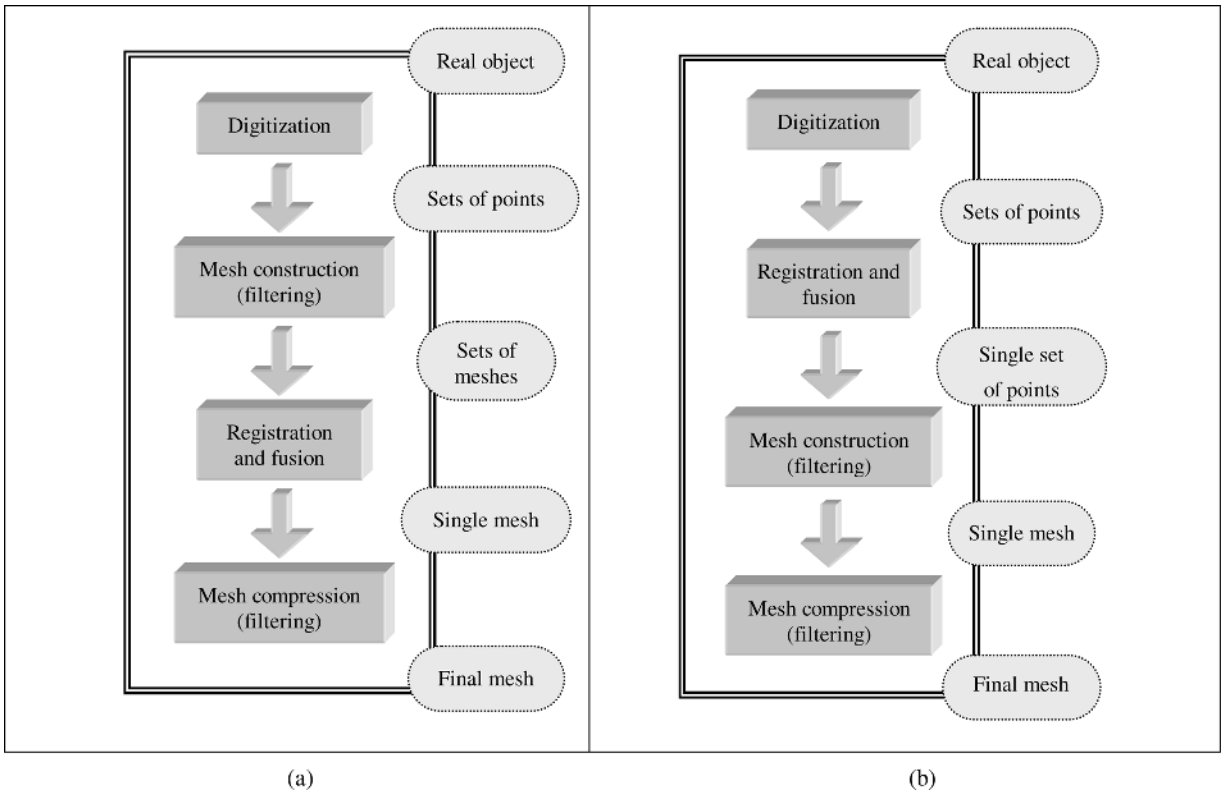

Fig. 3. Two pipelines commonly used to convert clouds of points into a 3-D mesh. (a) Partial meshes are first created, and then zippered together to produce a single large mesh. (b) Data points obtained by different partial scans are first fused together and then filtered; only afterwards is a unique mesh created. This pipeline is the one adopted here.

data points sampled [6], [22]–[24]: 3-D points from different scans are pulled together and analyzed. First, the number of 3-D points is reduced, so as to eliminate measurement noise, producing a reduced set of points whose position is error free. In the second stage, a host of standard, fast-interpolating algorithms, like marching cubes, can be reliably used to convert the reduced points cloud into a mesh [19], [25]. Because the approaches belonging to this second pipeline do not require building and storing intermediate meshes, their additional memory-allocation needs are greatly diminished. The decrease in main-memory loading and unloading may save processing time [21].

The simplest algorithms in this class subdivide the volume occupied by the object into microvoxels: All the points contained in the same voxel are replaced by a single point obtained as a weighted average of these points [6], [22], [23]. The appeal of this approach lies in the local nature of the computation, which makes it especially fast. However, it suffers from a few drawbacks. The reduced set of data points tends to be uniformly distributed in the object's space. This is often not optimal, because greater point density would be required in regions with greater detail than in shallow regions. Moreover, this method performance is critically affected by the voxel sidelength, which has to be accurately defined *a priori*.

An improved version of this technique has been proposed by Low and Tan [26]. In their approach, the points are put in an ordered list according to their perceptual importance. A box, centered on the first point on the list, is generated, and a single point is substituted for all the points inside the box. Since for assessing the perceptual importance of each data point a mesh has to be built, this approach has not been further developed for reducing and filtering point clouds.

Adaptive solutions have recently been introduced to adapt the voxel size locally to the data. In [6], a region-growing approach is reported: A point is selected at random and a cluster is grown by iteratively grouping its nearest neighboring points until a certain criterion is met (cluster size or variation). A second data point is then chosen and the cluster is grown by considering only the remaining data points. The algorithm terminates when all the data points have been considered (cf. also [27]). The initial choice of the first data point examined is critical to this approach. Different choices may produce very different results. The opposite approach is hierarchical clustering [22], [28], [29], where clusters are generated by recursively splitting clusters that do not meet predefined criteria (e.g., cluster size and estimated curvature). The whole data set is considered the initial starting cluster.

Both these approaches suffer from the drawbacks posed by disjunctive subdivision of the data set (each point belongs to only one cluster), which can produce a suboptimal solution and may lead to unsatisfactory quality overall. Spurious brisk variations in surface orientation or thickening effects do occur, since neighboring points on the surface may be assigned to two different voxels [6], [22]. These are well-known problems associated with hard clustering [30] and call for substantial post-processing. Moreover, because clusters are created sequentially,

these algorithms cannot be parallelized and the appeal of the microvoxel approach disappears.

We introduce here soft vector quantization (VQ) techniques to overcome these problems. VQ techniques have long been applied to lossy compression in multidimensional signal transmission; they are used here to *lose* the digitization noise as well as to reduce the cardinality of the input data set. In a VQ framework, a set of M points, called *reference vectors* (RVs), is used to represent a set of $N > M$ data points, such that a certain cost function (e.g., reconstruction error) is minimized. Since VQ is an NP-hard problem, suboptimal solutions, which can be obtained through iterative adaptation of the RVs position [30]–[32], are generally accepted. Algorithms of this class have been developed mainly in the connectionist domain. They are based on combining soft-max strategies to move the RVs with a deterministic annealing schedule for the parameters. However, they share two main drawbacks: The annealing-based optimization procedure takes a long time to converge even to a suboptimal result, and the parameter setting is critical.

Another main contribution of this paper is the introduction of a regular volume subdivision into disjointed regions (macrovoxels) named hyperboxes (HBs), in association with soft-clustering. This is fully exploited to achieve a great speed increase during the optimization phase. Moreover, it allows deriving a fully parallel implementation of the algorithm.

The last main contribution is the derivation, from theoretical issues, of an automated procedure for reliably setting all the parameters by analyzing data distribution, which, as far as we know, is one of the very few examples of this kind. Because the only parameter specified by the user is the compression rate, the procedure can be used by not-trained users.

The overall procedure has been called enhanced vector quantization (EVQ). Results on reducing and filtering 3-D point clouds are reported and discussed.

The paper is organized as follows. After introducing soft VQ in Section II, we introduce HBs processing in Section III and the computation of the parameters in Section IV. Results are reported in Section V and discussed in Section VI.

II. Soft VQ

VQ techniques [33] use a set of M RVs, $W = \{w_j \in R^D\}$, to approximate an input data set $V = \{v_k \in R^D\}$ of cardinality N where $N > M$. The RVs are positioned such that they minimize the following reconstruction error, $E(V, W)$:

$$E(V,W) = \frac{\sum_{k=1}^{N} d(v_k, w_j(v_k))^2}{N} = \frac{\sum_{k=1}^{N} ||v_k - w_j(v_k)||^2}{N} \tag{1}$$

where $w(v_k) \in W$ is the RV closest to v_k or the "winning" RV.

To determine the optimal position of the RVs, different iterative techniques, based on soft-max adaptation, have been proposed [30]. In this approach, at each iteration t, a single data point $\tilde{v}(t)$, is randomly extracted from the input data set, and the position of all the RVs is updated according to an adequate weighting function. Among soft VQ techniques, "neural gas" (NG) [34], [35] has been chosen here as the "computational engine." NG has been shown to be superior to other VQ algorithms (Kohonen maps, k-means, or maximum-entropy clustering), at least for dense problems of low dimensionality [34]. Other choices [30] may be successful as well. On the other hand, we have experimentally verified that EVQ based on NG is effective for point clouds reduction and filtering.

In NG, the RVs $\{w_j\}$ are updated according to the following soft-max rule:

$$\Delta w_j(t) = \varepsilon(t) \cdot h_{\lambda(t)}(k_j(\tilde{v}(t), w_j))(\tilde{v}(t) - w_j) \tag{5}$$

where $k_j \in \{0, 1, 2, \ldots M-1\}$ is the ranking of w_j with respect to the actual data vector $\tilde{v}(t)$, assessed through the Euclidean distance. $h_{\lambda(t)}(\cdot)$ controls the amplitude of the region of influence of each $\tilde{v}(t)$ and is implemented with the following soft-max weighting function:

$$h_{\lambda(t)}(\tilde{v}(t), w_j) = \exp\left(-\frac{k_j(\tilde{v}(t), w_j)}{\lambda(t)}\right). \tag{6}$$

The function $\lambda(t)$ controls the number of RVs that are meaningfully updated by the data point $\tilde{v}(t)$. $\lambda(t)$ and $\varepsilon(t)$ decrease as optimization progresses according to

$$\lambda(t) = \lambda_i \left(\frac{\lambda_f}{\lambda_i}\right)^{t/T_{\max}} \tag{7a}$$

$$\varepsilon(t) = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/T_{\max}} \tag{7b}$$

where $T_{\max}$ is the number of iterations. The role of $\lambda(t)$ and $\varepsilon(t)$ is to reduce, as optimization progresses, the number of RVs effectively displaced and the extent of the displacement associated with $\tilde{v}(t)$, respectively. If, at the beginning, each point sampled induces an appreciable displacement of most RVs, in the end, only the winning RV is significantly displaced. This behavior has been called "first-search-then-converge" [36] [cf. Fig. 4(a) and (b)]. Initial optimization iterations are devoted to homogenizing the RVs inside the input space; the refinement towards optimal distribution takes place afterwards.

III. EVQ

For large data sets, soft VQ optimization can be extremely time-consuming. To meet high throughput specifications, we have developed several improvements. These are described in this section. They are aimed to reduce the initial search phase and the computational cost of each iteration. To this purpose, data locality and data partitioning into a regular structure were fully exploited (cf. also [37] and [38]).

A. HB and RV Initialization

In the asymptotic condition, the statistical distribution of the RVs is proportional to that of the data points according to the Zador's criterion [34], [39]

$$\rho_w(u) \propto p_v(u)^{\gamma}, \quad \text{with} \quad \gamma = \frac{D}{D+2} \tag{8}$$

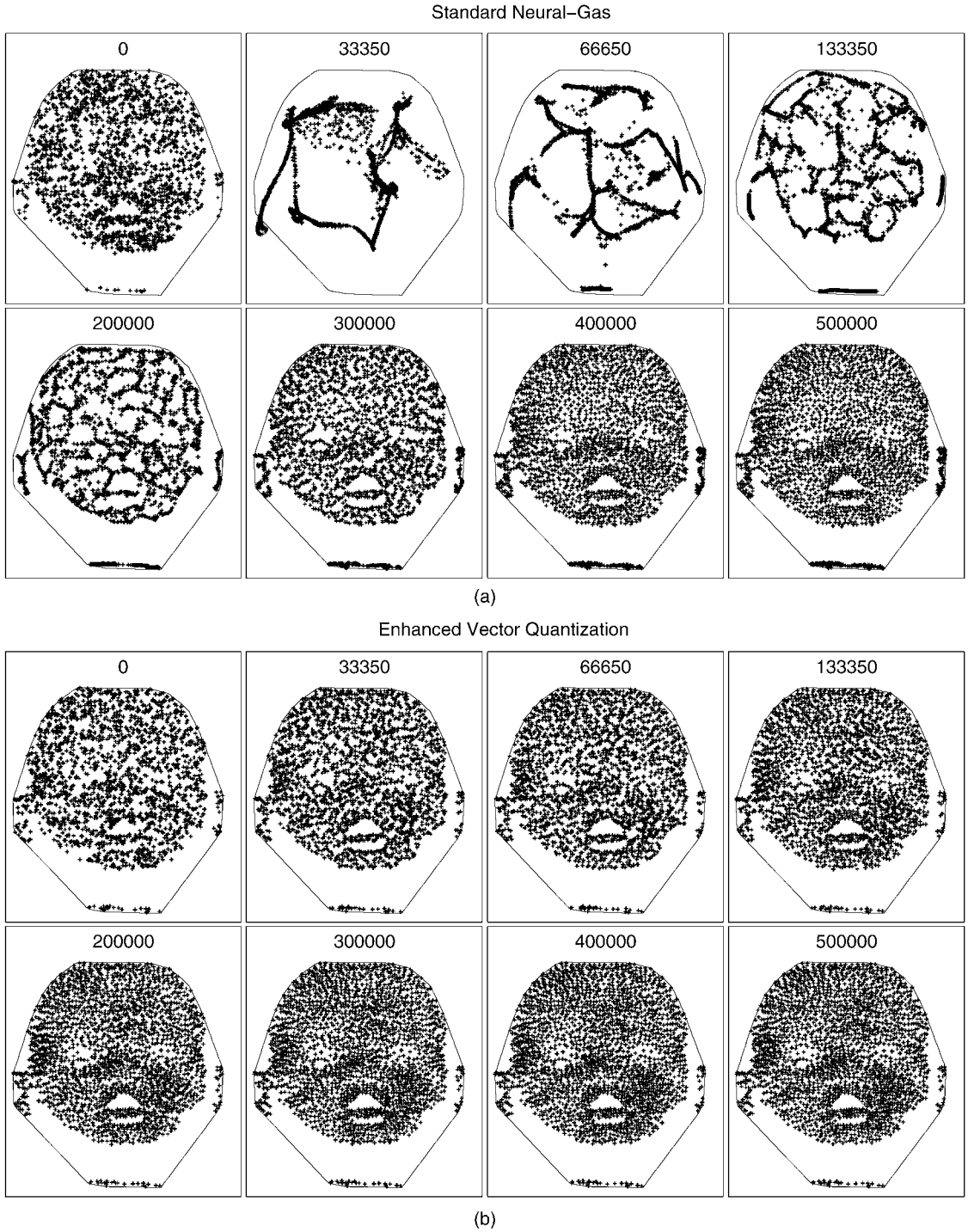

Fig. 4. Evolution of the position of the 2000 RVs used to construct the model in Figs 2(c) and (d) is plotted at different iteration steps: $t = 0$, $t = 33\,350$, $t = 66\,650$, $t = 133\,350$, $t = 200\,000$, $t = 300\,000$, $t = 400\,000$, and $t = 500\,000$, when standard (a) NG or (b) EVQ was used. In standard NG, at the outset, each sampled point causes an appreciable displacement of most RVs, while, at the end, only the winning RV is significantly displaced. This behavior has been called "first-search-then-converge" [36]: The initial optimization iterations are dedicated to homogenizing the RVs inside the input space; refinement towards optimal distribution takes place afterwards. In EVQ, the search stage can be skipped as the RVs are initialized near to their optimal position.

where $\rho_w(u)$ and $p_v(u)$ are, respectively, the density of the RVs and the probability density of the data points, whereas D is the dimensionality of the data space ($D = 3$ and $\gamma = 0.6$ for clouds of 3-D points). Following the central limit theorem, we will approximate $p_v(u)$ with the local density of the data points $\rho_w(u)$. From (8), an efficient initial distribution of the RVs can be implemented.

Let us call τ the desired compression rate

$$\tau = \frac{M}{N} \quad \Rightarrow \quad M = \tau N \tag{9}$$

and B the region of R^D that contains all the data points. Partitioning B into N_H disjointed regions $\{B_k\}$, it holds that

$$\bigcup_{k=1}^{N_H} B_k = B, \text{ with } B_k \cap B_j = \phi \qquad \forall k \neq j \tag{10a}$$

$$V_B = \sum_{k=1}^{N_H} V_k \tag{10b}$$

where V_B and V_k are the volumes of B and B_k, respectively. As a result

$$N = \sum_{k=1}^{N_H} N_k \quad \text{and} \quad M = \sum_{k=1}^{N_H} M_k \tag{11}$$

where N_k and M_k are the number of data points and of RVs inside B_k, respectively. Applying (8) to the data points inside B_k, the RVs' mean density can be determined as

$$\bar{\rho}_w(u)|_k = \beta \bar{\rho}_w(u)|_k^{\gamma} \quad \Rightarrow \frac{M_k}{V_k} = \beta \left(\frac{N_k}{V_k}\right)^{\gamma} \tag{12}$$

where $\bar{\rho}_w(u)|_k$ and $\bar{\rho}_v(u)|_k$ are the mean density of the data points and the RVs inside B_k. From (10) and (11), it follows:

$$\beta = \frac{M}{\sum_{k=1}^{N_H} N_k^{\gamma} V_k^{1-\gamma}}. \tag{13}$$

The number of RVs to be inserted inside each region B_k, M_k can, therefore, be computed through (9), (12), and (13) as

$$M_k = M \frac{N_k^{\gamma} V_k^{1-\gamma}}{\sum_{k=1}^{N_H} N_k^{\gamma} V_k^{1-\gamma}} = \tau N \frac{N_k^{\gamma} V_k^{1-\gamma}}{\sum_{k=1}^{N_H} N_k^{\gamma} V_k^{1-\gamma}}. \tag{14}$$

The partitioning schema can be applied, in principle, to disjointed regions of any shape. If the regions are polyhedrons with their faces parallel to the axes, they are called HBs. In 3-D space, the HBs are parallelepipeds, which are usually termed voxels [3].

Whenever B is an HB and all the subregions $\{B_k\}$ have the same shape and volume, (14) can be simplified as

$$M_k = M \frac{N_k^{\gamma}}{\sum_{k=1}^{N_H} N_k^{\gamma}} = \tau N \frac{N_k^{\gamma}}{\sum_{k=1}^{N_H} N_k^{\gamma}} \tag{15}$$

which does not depend on any volume measurement. Equation (15) will be called *partitioning function*.

If (15) gives noninteger values for any M_k, one possibility is to round these M_k to the superior integer. In this case, the total number of RVs, M, will be slightly larger than that prescribed by the compression rate (9). This is sufficient in all cases where the compression rate is an estimate. When the exact number of RVs prescribed by (9) has to be met, the following two-step procedure can be adopted. First, all the M_k's are rounded down to the nearest integer $\lfloor M_k \rfloor$ as a result; for each B_k, a fraction $(M_k - \lfloor M_k \rfloor < 1)$ of RVs is left out. The sum of these fractions $M_{\text{rem}} = M - \sum_{k=1}^{N_H} \lfloor M_k \rfloor$ represents the total number of RVs, which still have to be distributed inside the B_k's. To distribute them, the relative partitioning error e_k

$$e_k = \frac{M_k - \lfloor M_k \rfloor}{M_k} \quad \Rightarrow \quad 0 \leq e_k < \frac{1}{M_k} \tag{16}$$

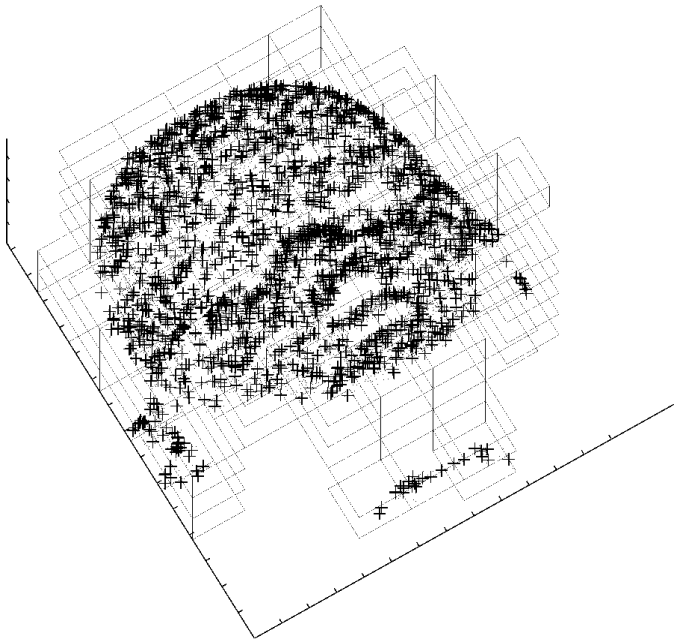

Fig. 5. Macrovoxels in which data space is partitioned through HB are shown. The data points are plotted as small dots and the initial position of the reference vectors, determined as in Section III-A, is represented by "+." For the sake of clarity, only 10 000 points of the original 100 000 points of the data set, are plotted. For each iteration, a data point is randomly extracted. With HB partitioning, all computation is restricted to the RVs inside the macrovoxel to which the data point belongs or inside the adjacent ones.

is computed for each B_k. The B_k's are then sorted by decreasing e_k, creating a priority list. The M_{rem} RVs are assigned to the B_k's, one per box, to the first M_{rem} boxes. The choice of relative error (16) favors those B_k's that contain fewer data points.

Once the number of RVs for each B_k has been decided, their position inside B_k must be defined. If inserted randomly, they might be far from the object's surface and it might take too long to be attracted to it. A better (and simpler) solution is to make the RVs coincident with one of the points sampled that belongs to the region. An example of HB partitioning is reported in Fig. 5.

The HBs are stored into a D-dimensional array. The box associated with each data point (or RV) can be directly addressed through the set of D indices $(i_1, i_2, \ldots, i_n)$, computed as

$$i_j = \lfloor ((x_j - x_{j_{\min}})/L_j) \rfloor \tag{17}$$

where $x_{j_{\min}}$ is the minimum coordinate along the jth axis of the input space and L_j is the length of side of the box along that direction.

B. Speedup Through HB Data Partitioning

In NG, the most costly operation in the optimization phase (5) is ordering the RVs by their distance from $\tilde{v}(t)$, so as to compute their ranking. This operation, common to most soft VQ techniques, is very expensive, being equal to $M \cdot \log M$. Taking full advantage of HB, a drastic reduction in processing time can be achieved by considering only the RVs closest to $\tilde{v}(t)$ in (5).

In fact, at the beginning of the optimization procedure, $\lambda(t)$ [see (6) and (7a)] must be large enough to allow all RVs maximum freedom of motion, since RVs may have to move through the whole data space to reach the region of their final destination

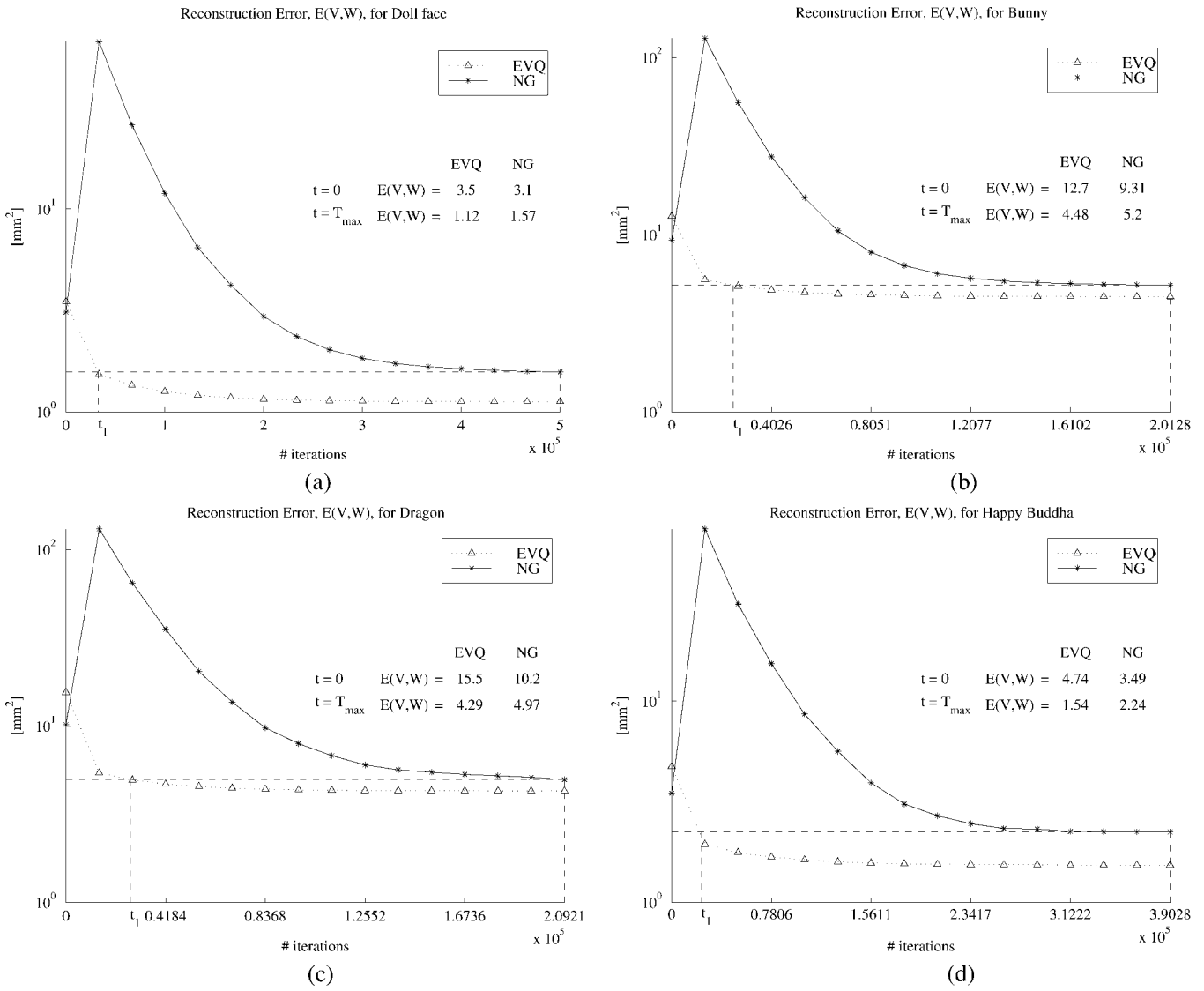

Fig. 6. Reconstruction error $E(V,W)$ for four different models, (a) doll face, (b) bunny, (c) dragon and (d) happy Buddha, is reported as a function of the number of iterations when standard NG (continuous line with asterisks) and EVQ (dotted line with triangles) were used. The error achieved by NG after $5N = 500\,000$ iterations is achieved by EVQ already after t_1 iterations. t_1 ranges in $6\% \div 13\%$ of the total number of iterations (see Table I for numerical details on the figures of merit). $E(V,W)$ has been averaged over 40 trials. The parameters used are the ones suggested by experimental results: $M_g = 12$ and $\eta = 0.2$.

[36] (cf. Fig. 4). Therefore, initial optimization steps are spent distributing the RVs inside the data space. Only in a second phase are the RVs directed towards their final destination [cf. Fig. 4(a)]. With HB processing, the first optimization phase can be skipped: Because RV distribution (15) is already close to optimal [cf. Figs. 4(b) and 6], there is no need to apply function (5) to all the RVs, but only to those RVs near the data point analyzed $\tilde{v}(t)$, which have to be displaced.

To this end, for each point sampled $\tilde{v}(t)$, an *influence region* $R_\alpha(\tilde{v}(t))$ is defined. This is the portion of the D-dimensional space that contains all the RVs that are reasonably close to $\tilde{v}(t)$. These are the ones that have to be moved toward $\tilde{v}(t)$; hence, (5) needs to be computed only for those RVs that belong to $R_\alpha(\tilde{v}(t))$. A natural choice for $R_\alpha(\tilde{v}(t))$ would be the box to which $\tilde{v}(t)$ belongs, $B_k(\tilde{v}(t))$. However, this would force all the RVs in $B_k(\tilde{v}(t))$ into the convex hull of the data points in that box, leading to patchy reconstruction. To avoid this, $R_\alpha(\tilde{v}(t))$ was defined as the region made up of the 2^D boxes, (eight in the 3-D space) nearest $\tilde{v}(t)$. This reduces the computational cost of ranking the data points to a small fixed quantity, equal approximately to $2^L \bar{M} \log(2^L \bar{M})$, where $\bar{M}$ is the mean number of RVs inside each box. As stated previously, when M increases, the number of boxes also increases, leaving unaffected the expected number of RVs to be sorted in (5) (i.e., those inside $R_\alpha(\tilde{v}(t))$).

IV. Automatic Setting of the Parameters

Another major contribution of this paper is an original procedure that enables us to derive a reliable value for the parameters ε_i, λ_i and the B_k's side, L_k. For simplicity, all HBs are assumed to have all sides of the same length $L = L_k$, but the procedure can easily be extended to sides of different length. The initialization procedure is described below.

A. Setting the HB Side

The HB sidelength L is a critical parameter (cf. [1], [6], and [22]). If L were chosen too large, little advantage would be gained from HB processing; if it were too small, the statistical significance of (15) would be questionable due to excessive fragmentation of the data points. The strategy followed here guarantees that, on average, a certain number of RVs, M_g, are placed inside each B_k. An iterative procedure has been developed for this purpose.

The initial value of L is set to

$$L = \sqrt{[D]}\frac{V}{M}M_g. \tag{18}$$

This value guarantees M_g RVs inside each box, when the data points are uniformly distributed. The actual mean number of RVs per box is computed, according to (15), as

$$\bar{M}(L) = \frac{\sum_{k=1}^{N_H} M_k(L)}{N_H(L)}.$$

If $\bar{M}(L) < M_g, L$ is increased, vice versa if $\bar{M}(L) > M_g$. This procedure is iterated until $\bar{M} \approx M_g$, which requires few iterations.

B. Saving Computational Time and Setting the Value of λ

The parameter λ_i (7a) sets the number of RVs that are appreciably updated at the first iteration step of the optimization procedure. A reasonable value for λ_i can be derived using the influence region $R_\alpha(\tilde{v}(t))$. Let us consider the first point extracted $\tilde{v}(t_o)$, and assume that, at the first iteration, only a certain subset η of the RVs inside $R_\alpha(\tilde{v}(t))$ receives a consistent update. If, by somehow arbitrary choice, a consistent update is considered when $h_\lambda(\cdot) \geq e^{-1}$ [see (5)], it follows:

$$\lambda_i \geq \eta\frac{\sum_{k:B_k \in R_\alpha(\cdot)} M_k}{2^D}. \tag{19}$$

λ_i is, therefore, a function only of the number of RVs contained inside $R_\alpha(\tilde{v}(t))$; and, it can be different for data points that belong to different boxes.

In deterministic annealing, λ starts from λ_i and goes to zero asymptotically (for $t \to \infty$). Given the limited number of iterations available, λ_f as to be specified. A natural choice is to set it as a very small value at the end of the optimization phase [for $t = T_{\max}$, in (7)] [30], [31], [34]. For sake of simplicity, this value is assumed to be a small fraction θ_λ of $\lambda_i (\lambda_f = \theta_\lambda \lambda_i)$ with $\theta_\lambda = 0.001$. With this choice, the displacement of the second closest RV, in the last optimization step, is weighted $\exp(-(1000/\lambda_i))$, which can be assumed to be reasonably close to zero.

C. Setting the Value of ε

The parameter ε controls the plasticity of the RVs, such that in the optimization phase, each RV w_j can explore the whole influence region to which it belongs. This idea is translated into mathematical terms by setting ε_i (5) so that the expected length of the path followed by each w_j in the optimization phase $\Delta w_{j_{\mathrm{TOT}}}$ can be at least as large as the diagonal of the influence region

$$\Delta w_{j_{\mathrm{TOT}}} = \sum_{t=0}^{T_{\max}} |\Delta w_j(t)| \geq 2L_k\sqrt{D}. \tag{20}$$

Using a probabilistic approach and taking into account (5), (6), and (7), $\Delta w_{j_{\mathrm{TOT}}}$ can be modeled as

$$\Delta w_{j_{\mathrm{TOT}}} = \sum_{t=0}^{T_{\max}} P(w_j \in R_\alpha(\tilde{v}(t)))\varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/T_{\max}} \cdot h_{\lambda(t)}(\tilde{v}(t), w_j)|\tilde{v}(t) - w_j| \tag{21}$$

which depends on $\varepsilon_i, \varepsilon_f$ and on three functions: $P(\cdot)$ is the probability that w_j belongs to $R_\alpha(\tilde{v}(t))$, $h_{\lambda(t)}(\cdot)$ is the soft-max weighting function (6), and $|\tilde{v}(t) - w_j|$ is the distance between w_j and the actual data point $\tilde{v}(t)$. To get a reasonable simple relationship between ε and the other terms in (21), a few simplifications are required.

First, $P(\cdot)$ can be approximated with the fraction of RVs inside: $R_\alpha(\tilde{v}(t))$

$$P(w_j \in R_\alpha(\tilde{v}(t))) \cong \frac{2^D\bar{M}}{M} \tag{22}$$

where the mean number of RVs inside $R_\alpha(v)$ is assumed to be equal to $2^D\bar{M}$. Under this hypothesis, the probability that w_j ranks p with respect to $\tilde{v}(t)$ is $1/2^D\bar{M}$ and (21) can be rewritten as

$$\Delta w_{j_{\mathrm{TOT}}} = \frac{1}{M}\sum_{t=0}^{T_{\max}}\sum_{p=0}^{(2^D\bar{M})-1} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{\frac{t}{T_{\max}}} \times \exp\left(-\frac{p}{\lambda_i\left(\frac{\lambda_f}{\lambda_i}\right)^{\frac{t}{T_{\max}}}}\right) \times E\left[|\tilde{v}(t) - w_j|\,; k_j\left(w_j, \tilde{v}(t)\right) = p\right] \tag{23}$$

where $E[|\tilde{v}(t) - w_j|; k_j(w_j, \tilde{v}(t)) = p]$ is the estimated mean distance between $\tilde{v}(t)$ and the jth RV, w_j, when w_j is the pth closest to $\tilde{v}(t)$. Considering that the displacement of an RV, aside from the winning one $(p = 0)$, decays rapidly toward zero with the number of iterations, (23) becomes

$$\Delta w_{j_{\mathrm{TOT}}} > \frac{1}{M}\sum_{t=0}^{T_{\max}} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{\frac{t}{T_{\max}}} \times E[|\tilde{v}(t) - w_j|; k_j(w_j, \tilde{v}(t)) = 0] \tag{24}$$

where only the contribution of the winning RV is taken into account. If we hypothesize that the mean distance between an RV and the closest data point is approximately constant in the optimization phase, the cumulative value of $E(\cdot), \bar{W}_0$ can be derived as reported in Appendix.

$\bar{W}_0$ turns out to be a nonlinear function of the number of RVs and of number of dimensions the data space has and a linear

function of L : $\bar{W}_0 = f_0(\bar{M}, D)L$ (37). From all the previous considerations, (24) becomes

$$\Delta w_{j_{\mathrm{TOT}}} > \frac{\bar{W}_0}{M} \sum_{t=0}^{T_{\max}} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i} \right)^{t/T_{\max}}. \tag{25}$$

Substituting (25) for (20) and expressing ε_f as a fraction of $\varepsilon_i (\varepsilon_f = \theta_\varepsilon \varepsilon_i), \varepsilon_i$ can be estimated as

$$\begin{aligned} \varepsilon_i &< \frac{M}{\bar{W}_0} \frac{1-\theta_\varepsilon}{1-(\theta_\varepsilon)^{1+\frac{1}{T_{\max}}}} 2L\sqrt{D} \\ &= \frac{2M}{f_0\left(\bar{M}, D\right)} \frac{1-\theta_\varepsilon}{1-(\theta_\varepsilon)^{1+\frac{1}{T_{\max}}}} \sqrt{D} \end{aligned} \tag{26}$$

which gives an upper bound for ε_i. As expected, ε_i does not depend on the size of the boxes.

When a tighter bound is required for ε_i, (23) can be modified to take into account the contributions of the other RVs, obtaining

$$\begin{aligned} \Delta w_{j_{\mathrm{TOT}}} = \frac{1}{M} \sum_{t=0}^{T_{\max}} \sum_{p=0}^{(2^D \bar{M})-1} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i} \right)^{\frac{t}{T_{\max}}} \\ \times \exp\left(- \frac{p}{\lambda_i \left(\frac{\lambda_f}{\lambda_i} \right)^{\frac{t}{T_{\max}}}} \right) f_p(\bar{M}, D) L. \end{aligned} \tag{27}$$

In this case, an iterative procedure to determine ε_i must be adopted. First, ε_i is initialized with the value of $\Delta w_{j_{\mathrm{TOT}}}$ computed in (23). This value is then iteratively increased or decreased until $\Delta w_{j_{\mathrm{TOT}}}$ becomes close enough to $\sqrt{DL}$. This procedure converges in a few iterations. Alternatively, when ε_f and λ_f are expressed as a fraction of ε_i and λ_i, respectively, (27) is simplified as

$$\begin{aligned} \Delta w_{j_{\mathrm{TOT}}} = \varepsilon_i \frac{1}{M} \sum_{t=0}^{T_{\max}} \sum_{p=0}^{(2^D \bar{M})-1} \theta_\varepsilon^{\frac{t}{T_{\max}}} \\ \times \exp\left(- \frac{p}{\lambda_i \theta_\lambda^{\frac{t}{T_{\max}}}} \right) f_p(\bar{M}, D) L \end{aligned} \tag{28}$$

from which ε_i can be directly computed.

V. Experimental Evaluation

We have used EVQ extensively to reduce the cardinality of 3-D data sets acquired by 3-D scanners, eliminating the noise on them. Some results on our own models and on models taken from the web are reported here.

A. Reconstruction Error Measure

If the surface could be measured accurately, a natural error measurement would have been the difference between true surface height and reconstructed surface height (cf. [19]). However, this measurement is not available in our case and two alternative indicators must be used. Although the Hausdorff distance and the cross distance between different clouds of points have been proposed [18], [19], there is not complete agreement on these measurements, especially when the points are affected by noise, which biases these measurements.

Therefore, $E(V, W)$ in (1), which is a typical error measurement used in VQ—and used implicitly by some simplification techniques based on clustering [6], [22]—was adopted here. The rationale according to which (1) can be assumed to be as a good indicator of surface-reconstruction error is discussed later.

The correct position of an RV, w_j, is on the object's surface S

$$\|w_j - S\| = 0. \tag{29}$$

However, only noisy surface samples $\{v_k\}$ are available. The relationship between their true position and their measured one can be broken down as

$$\sum_k \|v_k - v_{k_{\mathrm{true}}}\|^2 \leq \sum_k \|v_k - v_{k\perp}\|^2 + \sum_k \|v_{k\perp} - v_{k_{\mathrm{true}}}\|^2 \tag{30}$$

where $v_{k_\perp}$ is the projection of v_k normal to S. Notice that the first term contains a surface-measurement error, while the second term is due to a displacement of v_k on the manifold and in this respect does not contain any error in surface measurement.

Because we have dense distributions for both V and W, the manifold can be approximated locally with a plane. In this case, (30) can be rewritten with an equal sign because $v_k - v_{k\perp}$ is perpendicular to $v_{k\perp} - v_{k_{\mathrm{true}}}$. If we call Π_j the plane tangential to S in the jth RV, the error measurement (1) can be split into

$$\sum_k \|v_{kj} - w_j\|^2 = \sum_k \left\|v_{kj} - v_{kj_\perp}\right\|^2 + \sum_k \left\|v_{kj_\perp} - w_j\right\|^2 \tag{31}$$

where the sum is limited only to those v_k's which lie in the neighborhood of w_j, v_{kj}, i.e., those for which w_j is the winner: $v_{kj} - w_j < |v_{kj} - w_i|, \forall i \neq j$. The terms $\|v_{kj_\perp} - w_j\|$ lie on Π_j.

If we hypothesize that error measurement is additive, Gaussian and has zero mean (which is very often the case), the following observations can be made. The first term in (31) is minimized when Π_j's coincident to S (locally), i.e., when it is the plane that best fits in the least squares sense the $\{v_{kj}\}$. This term constraints w_j onto Π_j and, therefore, onto S. The second term increases with the density of v_{kj} and their distance from w_j: It guides the distribution of the RVs, placing more RVs where more data have been sampled. Therefore, (1) evaluates both the surface-reconstruction error and the quality of the distribution of the RVs.

We explicitly note that the error function (1) could be modified by weighting the different data points, or even their coordinates, differently, whenever *a priori* local information on measurement error is available.

B. Results

This method has been applied extensively to data sets acquired through 3-D scanners. A typical data ensemble sampled on a face is reported in Fig. 1(a)–(c), where a total of $N = 100\,000$ 3-D points have been sampled over the face of the doll reported in Fig. 1(d). In Fig. 2(b), the 3-D mesh obtained

TABLE I
QUANTITATIVE RESULTS OBTAINED WITH THE DIFFERENT ALGORITHMS ARE REPORTED FOR FOUR DIFFERENT MODELS. THE FIGURES ARE REPORTED AS MEAN (STANDARD DEVIATION) VALUE AVERAGED OVER 40 TRIALS. FOR EVQ THE VALUE OF t_1 IS ALSO REPORTED. THIS IS THE NUMBER OF ITERATIONS NEEDED TO MATCH THE RECONSTRUCTION ERROR ACHIEVED BY NG AFTER $T_{\max} = 5N$ ITERATIONS IS REPORTED AS NUMBER OF ITERATIONS. THE PERCENTAGE WITH RESPECT TO $T_{\max}$ IS INDICATED IN PARENTHESES

algorithm	figures of merit	Dataset: doll face (100000 points)	Bunny (40256 points)	Dragon (41841 points)	Happy Buddha (78056 points)
NG	Err(0)	3.1 (0.196)	9.31 (0.386)	10.2 (0.687)	3.49 (0.402)
	Err($T_{\max}$)	1.57 (0.413)	5.2 (0.893)	4.97 (1.8)	2.24 (0.95)
	$N_{du}(0)$	20.2 (4.29)	8.13 (2.66)	8.13 (3.16)	14.8 (3.94)
	$N_{du}(T_{\max})$	4.15 (4.32)	0.225 (0.62)	3.45 (3.66)	9.78 (9.66)
	Time	1516.0 s	225.2 s	243.7 s	905.9 s
EVQ with random parameters	Err(0)	4.57 (3.24)	14.1 (8.49)	14.9 (5.92)	4.39 (0.829)
	Err($T_{\max}$)	2.04 (2.42)	5.79 (2.53)	5.57 (2.36)	1.66 (0.105)
	$N_{du}(0)$	0 (0)	0 (0)	0 (0)	0 (0)
	$N_{du}(T_{\max})$	0 (0)	0.9 (2.92)	0.875 (2.02)	1.55 (2.64)
EVQ	Err(0)	3.5 (0.114)	12.7 (0.548)	15.5 (0.864)	4.74 (0.211)
	Err($T_{\max}$)	1.12 (0.00395)	4.48 (0.0184)	4.29 (0.0204)	1.54 (0.00483)
	$N_{du}(0)$	0 (0)	0 (0)	0 (0)	0 (0)
	$N_{du}(T_{\max})$	0 (0)	0.025 (0.158)	0.25 (0.439)	0.05 (0.221)
	t_1	32,600 (6.52% $T_{\max}$)	24,800 (12.3% $T_{\max}$)	27,000 (12.9% $T_{\max}$)	23,300 (5.96% $T_{\max}$)
	Optimization time	78.63 s	26.41 s	29.40 s	52.44 s
	Init time	30.53 s	12.16 s	12.39 s	22.15 s

by interpolating all the points sampled emphasizes the need for filtering. Moreover, the huge number of triangles obtained (≈200 000) calls for data reduction Fig. 2(a). The method presented here is successful in accomplishing both tasks, as can be seen in Fig. 2(c)–(d), where only 2000 points (≈4000 triangles) are used to represent the same face.

Besides this qualitative evaluation of the reconstruction, a quantitative assessment has been carried out with the classical indexes used in soft VQ techniques: the reconstruction error, $E(V, W)$ as defined in (1) and the number of "*dead units*," N_{du}. A dead unit is an RV that, at the end of the optimization phase, is not the one nearest to any data point (it never wins) [31]. An RV becomes a dead unit when, during the optimization phase, 1) it becomes trapped between two or more data points, or 2) when its initial position is so distant from that of the data points that it is never significantly attracted to them. Dead units clearly produce a suboptimal solution, and, especially in the second case, they produce degeneration in mesh quality. For a safe reconstruction, dead units have to be checked and discarded. Furthermore, they waste computational resources. In order to compare the effectiveness and the efficiency of EVQ with respect to the standard NG, each of the two algorithms has been challenged with a 40-run session, each run made up of $T_{\max} = 5N$ iteration steps and characterized by a different initialization of the RVs' position, with a different presentation order of the data points during the optimization.

The value of $E(V, W)$ averaged over the 40 runs for the doll face is plotted in Fig. 6(a) as a function of the number of iterations for NG and EVQ. It is apparent that EVQ enables much faster convergence since the same value of $E(V, W)$ is already obtained after $t = t_1 = 6.52\% T_{\max}$ iterations, saving 93.48% of the iterations. Moreover, guiding the initial distribution of the RVs with HB, the dead units disappear. Overall, EVQ reduces computational time for the doll face data set from the 1516 s, measured on a Pentium IV, 2.0 GHz, with 1 GB of main memory, required by NG to compute the $T_{\max} = 500\,000$ optimization steps, to 109 s required by EVQ to obtain even better accuracy ($E(V, W)$ of 1.12 mm^2 versus 1.57 mm^2 and zero dead units versus 4.15 on the average). Of the 109 s total processing time, 30 s are used for accurate initialization, (see Section IV). If we accept the same value of $E(V, W)$ of NG, the optimization procedure can be stopped after only 5.1 s. Therefore, EVQ allows a great increase in speed and/or accuracy. Moreover, computational time does not depend on compression rate, i.e., on the number of RVs.

Similar results have been consistently obtained with data made available by other groups, such as those from the Stanford Computer Graphics Laboratory repository, Stanford University, Stanford, CA [40]. In particular, the $E(V, W)$ for one scan of the models: bunny (*file bun000.ply*), dragon (*dragonStandRight_0.ply*), and happy Buddha (*happyStandRight_0.ply*) is reported in Fig. 6(b)–(d). As can be seen, EVQ yields similar speed increases: Overall, the $E(V, W)$ obtained after $T_{\max} = 5N$ iterations with NG is already obtained after $t = t_1 = 6\% \div 13\%\ T_{\max}$iterations with EVQ. The average number of dead units which ranges from 0.225 to 14.9 with NG, almost disappears (0÷0.25) with EVQ (cf. Table I).

Saving in computational time increases with the data size cardinality, as it could be expected. Results on the complete dragon model (*dragon_vrip.ply*), made up of 437 645 data points, are reported in detail in Fig. 7. NG required more than 9 h of computations to process such model, while EVQ completed the task in ≈80 s. As expected, the detail increases with the number of RVs (with different compression rates); however, visual quality is already close to the original at a compression rate of 10%. Computational time does not depend on the number of RVs (on the compression rate).

The free parameters of the algorithm are M_g (17a), which determines the length of each box, and η (19), which determines the number of RVs whose position is significantly updated by

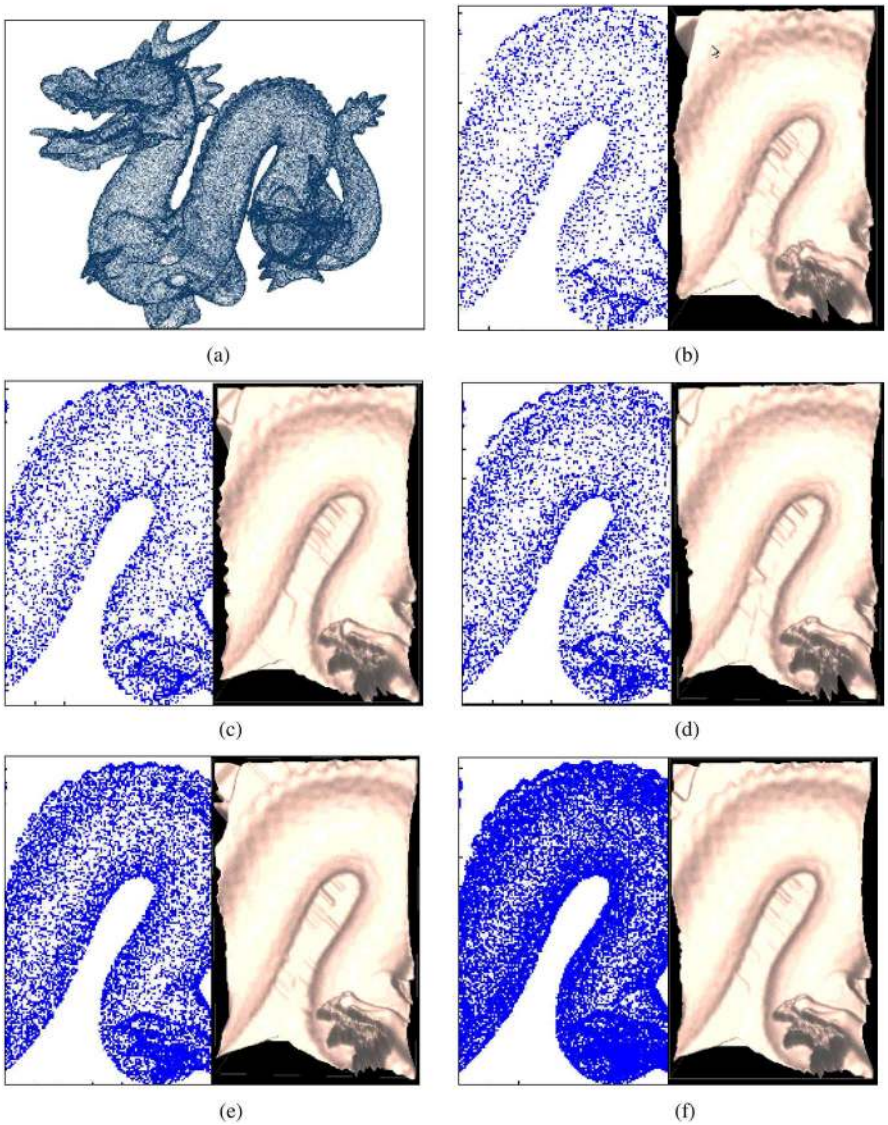

Fig. 7. (a) Original dragon data set [40], made up of 437 645 points. (b)–(f) Closeup of the mesh constructed after processing the original data with EVQ. (b) Reconstruction with 21 882 RVs (5%), computing time 80.2 s, number of voxels $23 \times 16 \times 10$, and voxel side 10.8 mm. (c) Reconstruction with 35 812 RVs (8%), computing time 79.7 s, number of voxels $28 \times 20 \times 13$, and voxel side 8.6 mm. (d) Reconstruction with 44 202 RVs (10%), computing time 80.3 s, number of voxels $32 \times 23 \times 15$, and voxel side 7.8 mm. (e) Reconstruction with 87 529 RVs (20%), computing time 80.4 s, number of voxels $44 \times 31 \times 20$, and voxel side 5.7 mm. (f) Reconstruction with 218 823 RVs (50%), computing time 80.1 s, number of voxels $69 \times 49 \times 31$, and voxel side 3.4 mm.

each data point. The value of these parameters was set experimentally: $M_g = 12$ and $\eta = 0.05$ have proven adequate for a large variety of data sets and compression rates. As can be seen in Fig. 8, where $E(V, W)$ is plotted as a function of these parameters for the models plotted, it exhibits a parabolic shape, which is consistent among the models. In a few cases, a lower $E(V, W)$ can be obtained by tuning the parameters to that particular data set. However, the minimal advantage in $E(V, W)$ is not justified by the time required for the tuning.

To test the adequacy of the most critical parameters ε_i, λ_i, and L, which are automatically determined inside the algorithm, $E(V, W)$ and N_{du} were computed when EVQ was run using randomly generated parameters in the range of $0.1 \div 10$ times the default values. Fig. 9 and Table I show that the error obtained with automatically determined parameters is equal to or smaller than that obtained with a random setting. Similar results were also obtained for a wide spectrum of compression rates and models.

VI. Discussion

The approach presented here can be regarded as a hybrid batch/online approach. Initialization is actually carried out on the whole data set which is a characteristic of batch procedures. The learning stage, instead, is carried out considering one data point at a time as in online algorithms. This approach opens new possibilities for all those domains where the prior probability of data points is available.

Pure batch approaches, like batch self-organizing maps (SOMs), require several iterations on the whole data sets to collect reliable statistics [13], while less than two iterations are usually sufficient to EVQ to achieve good results. This suggests not to consider batch approaches for reducing dense data sets in low-dimensionality spaces, like the data sets considered here. Moreover, batch approaches are very sensitive to initialization (they usually tend to cluster the RVs in the central region) [13].

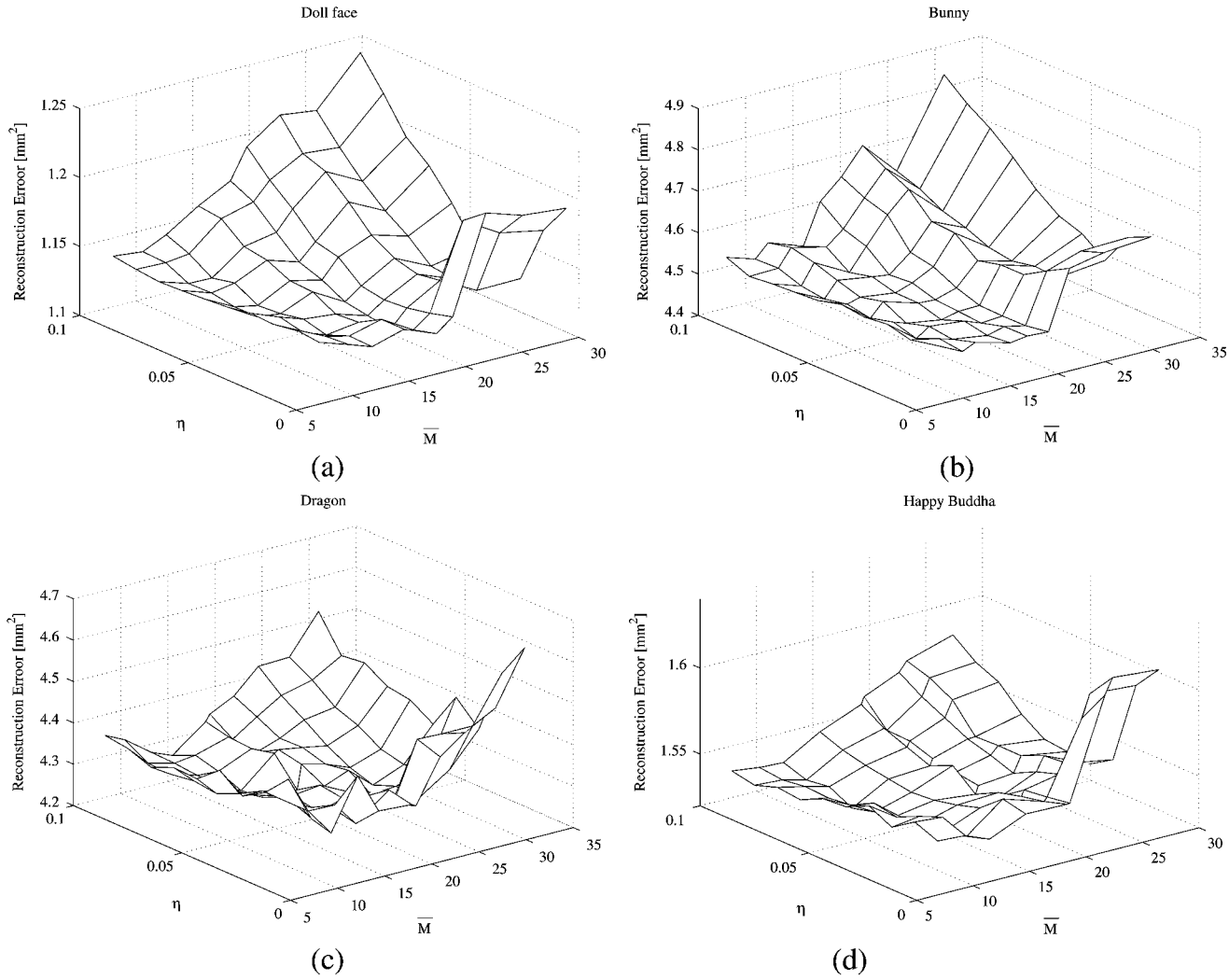

Fig. 8. Determining the parameters empirically. The reconstruction error is plotted as a function of $\bar{M}$ and η for four different models: (a) doll face, (b) bunny, (c) dragon, and (d) happy Buddha.

A. *Comparison With Other Approaches*

Local computation was the key to cutting down computational time. This has been achieved through HBs, which partition the data space into disjointed macrovoxels and is the core of EVQ. This mechanism is more powerful than that proposed in [31] where, for each data point, a fixed number RVs is displaced. Indeed, in [31], sorting all the RVs, which is the most computationally demanding operation, was still required at each iteration step.

A technique similar to HB partitioning is adopted by volumetric approaches, pioneered by [41], and further refined in [6], [18], [23], [28], [29], and [42]. However, the purpose of the partition is profoundly different in the technique presented here. In volumetric approaches, microvoxels are used: each voxel contains few data points, which are collapsed into a single representative point. As a result voxels are very small; for instance, in [6], a voxel size equal to the spacing in range images (0.5 mm) is suggested and, in [23], a voxel size of 0.35 mm was used with a partitioning into $712 \times 501 \times 332$ voxels. This choice often produces surface thickening [6], [23] and spurious discontinuities, which are due to assigning two nearby points each to a different (adjacent) voxel [22]. This may easily produce spurious peaks and gradients in the reconstructed mesh. This problem is similar to that encountered when hard clustering is adopted [30]. Moreover, due to the regular voxel structure, a regular spacing of the points is obtained, which does not reflect the local differential properties of the manifold.

Rather, in the approach presented here, macrovoxels are used: Each voxel contains a certain amount of data points and produces more than one RV. These are positioned so as to represent the data locally, inside their influence region ($R_\alpha(\tilde{v}(t))$ in Section III-A). Because adjacent influence regions partially overlap, thickening and surface discontinuities do not occur. This is well represented by the absence of dead units at the end of the optimization phase. Moreover, as stated previously, the error cost function (1) forces more RVs into those regions where the surface is more variable, producing a denser set of RVs in these regions [cf. Fig. 1(d)].

We explicitly note that the use of parallelepiped HBs introduces an approximation into the computation of a data point's influence region, which is in fact a hypersphere. However this

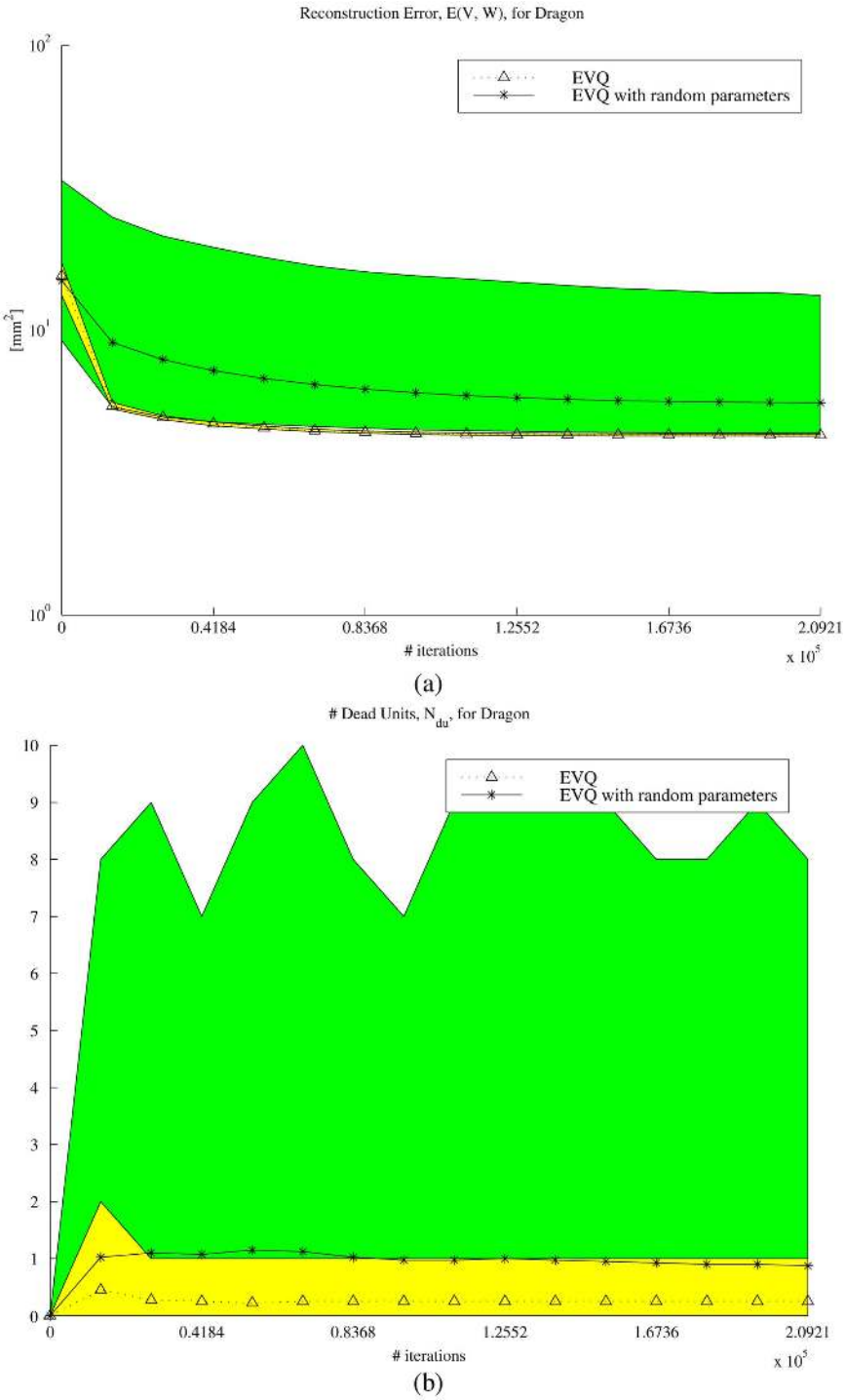

Fig. 9. Evaluation of the parameters set according to the procedure described in Section IV for the dragon data set. (a) Reconstruction error $E(V, W)$ and (b) number of dead units N_{du} are plotted as a dashed line with triangles for the parameters ε_i and λ_i, set according to Section IV. Different values of ε_i and λ_i, chosen randomly, between $0.1 \div 10$ times the value automatically set, produce a different curve for $E(V, W)$ and N_{du}. The ensemble of the curves obtained lies inside the dark shaded area. It is evident that setting the parameters automatically provides a good value for these parameters. Dashed lines with triangles in panels (a) and (b) are the same as in Fig. 5(a) and (b).

would not allow an efficient partitioning schema. Parallelepiped boxes can be accepted when using macrovoxels, since many RVs lie inside each influence region: Only those close to $\tilde{v}(t)$ are meaningfully displaced, while those that lie close to the border of $R_\alpha(\tilde{v}(t))$ are usually far from $\tilde{v}(t)$ and receive a low ranking and almost no updating through (5) (cf. also [22]). The approximation of the sphere with a parallelepiped can be questioned in those approaches based on microvoxels and it may be one of the main sources of thickening, spurious peaks, and surface gradients generated by these approaches. In any case, the approximation of a sphere with a cubic box can be accepted only for spaces of low dimensionality. Indeed, the probability of an RV inside $R_\alpha(\tilde{v}(t))$ being farther from the center of the box than is an RV outside $R_\alpha(\tilde{v}(t))$ is rather low in space that has low dimension-

alities. However, it increases exponentially as the number of the dimensions grows.

A potential risk of EVQ, shared by all the procedures based on volumetric techniques, is clustering points that are geometrically close but topologically distant, for instance points that lie on the two slopes of the same valley. In this case, spurious RVs would be positioned half-way between the two slopes. However, as 3-D digitizers usually produce very dense data sets, RVs that lie on the opposite slopes with respect to a data point will be ranked low in (6) and receive negligible updating in (5), while they will be attracted to data points on the same valley. This is particularly evident in analyzing the lips region in Figs. 1 and 2. No range data have been acquired in the mouth region for this model but only on the surrounding lips (Fig. 1). The RVs (Fig. 2) follow this topology and they distribute only along the lips; no RV is attracted into the empty space of the mouth. Moreover, the HB structure contrasts RVs traveling to distant boxes. In this respect, the approach presented here lowers the risk. Moreover, it should be remarked that clustering points that are topologically distant but geometrically close might not be a problem when rendering objects far from the observer (low resolution, low visual angle models); in such cases, overall appearance is perceptually more important than topology [12].

B. Computational Time

The cost of NG is $O(NM \log M)$. This can be broken down into the cost of each iteration, dominated by sorting the RVs, which is $O(M \log M)$, and the number of optimization steps, which increases linearly with the number of points sampled [34]. Experimental values of $1N \div 2N$ can be considered more than adequate with EVQ (cf. Fig. 6). HB processing allows reducing the cost of each iteration to $O(\bar{M} \log \bar{M})$, a fixed quantity, independent of the total number of RVs. Therefore, when a different compression rate is used, the computational time per iteration does not change: As the number of RVs increases (larger values of M), L decreases and the number of boxes increases (and vice versa for smaller values of M) such as to keep $\bar{M} \approx M_g$ constant (Section IV-A).

A speed increase of almost two orders of magnitude with respect to standard NG has been consistently obtained in experiments on data sets of different size. Moreover, EVQ already presents a computational advantage already in the initialization phase. As is made clear in Fig. 4, the RVs can already be used in building a reliable 3-D mesh after few iteration steps. A further saving in computational time can be gained by fully parallelizing the algorithm. Simultaneously updating the position of those RVs whose influence regions do not overlap can reduce computational time down to sublinear in N, opening the door to real-time implementation.

In EVQ implementation in standard sequential machines, the computational time is about 2.5 times that of volumetric techniques based on hard clustering [22] (which reduces to less than 2 times when initialization is not required). This overhead can be accepted when considering the superior quality of the data obtained.

The drawback of HB processing, and the other voxel-based volumetric techniques, is additional memory occupancy. This has been acknowledged as a major problem for those approaches based on microvoxels [6], [22], [23]. As a matter of fact, efficient space partitioning requires a table, which allows immediate access to the data (17) at the price of one pointer per box. Its memory occupancy increases linearly with the number of boxes per dimension and exponentially with the space dimension D. For example, a total of 1440 boxes were used for the data in Fig. 1, with memory occupancy of 5760 B and 102 kB for the data in Fig. 7(f) (at a compression rate of 50%), with memory occupancy of 420 kB. A tradeoff between memory occupancy and algorithm speed was recently proposed by [22], [24], and [42]. In their approach, to avoid allocating memory to boxes that do not contain sampled points, a list has been substituted for the table: Only 202/1440 boxes would have been allocated for the data in Fig. 1. However, determining which box is associated with a given sampled point means searching a hash table, generating a computational cost that increases linearly with the number of boxes. This is not justified when the overhead for the HB pointers is much smaller than the size of the data as in our case; for instance, for dragon data, against 420 kB for the HB structure, more than 5 MB are required for the data (at 12 B per point). Therefore, the simpler table structure, which requires only one operation to search the data, whatever the number of boxes, has been adopted here.

C. Setting Parameters

The parameters are set automatically by analyzing data-point distribution. This procedure is quite time-consuming requiring about 1/5 of the total computational time (cf. Section V-B). Its cost can be reduced when only a rough initialization is accepted [no iterations are required to refine the estimate of L and ε_i through (18) and (23) or (24)], or a good value for the parameters is known in advance.

The voxel size is automatically adapted to the distribution of data points and to the compression rate (Section IV); for instance, for the dragon data in Fig. 7, the side of the voxel varies from 10.8 mm for a compression rate of 5% to 3.6 mm for a compression rate of 50%. Moreover, in EVQ, the number of RVs is not the same for all the boxes but is based on the local data-point density (15). This feature is particularly useful when used in conjunction with adaptive 3-D digitizers (like applied research's hand-held scanner) that allow the laser beam to be directed manually. In this case, longer scanning time is spent in those artifact areas where spatial details are concentrated, collecting more points inside such regions. A similar increase in data-point density is obtained by acquiring additional partial scans of artifacts's more detailed regions.

The parameters L, λ_i, and ε_i are derived from theoretical considerations (cf. Section IV). They are not claimed to be optimal, but they do guarantee a good solution in a reasonable time under a wide variety of compression rates and sampled point distributions (cf. Fig. 8 and Table I). L is the result of a tradeoff. A small value of L does not allow a reliable estimate of the local density of the data points and the RVs (12); on the other hand, too large a value of L does not allow a meaningful reduction in computational time. An attempt to find the optimal L (so that the optimization time is minimized) was recently proposed [43]. However, this approach does not yield an analytical solution, as proposed here. λ_i regulates the amount of "soft-max"

adaptation, or the amplitude of a data point's influence region, through the parameter η (18), which was determined experimentally for all models along with M_g. The larger η is, the greater the number of RVs meaningfully updated through (5). This decreases exponentially (7), as it does in most soft-clustering algorithms [30]. Although different decay functions may be employed [44], we chose the standard solution, which guarantees good results in this domain and allows a thorough comparison with other methods in the same class.

EVQ solves the problem of dead units through careful initial distribution of the RVs. This obviates the need for *ad hoc* techniques to avoid dead units, based on computing an adequate value of ε for each RV at each iteration step [31]. It is based on forcing all RVs to move during the optimization phase. The overall extent of the adaptation, $\varepsilon(t)$, is weighted for each RV using a measure of the displacement accumulated in the preceding optimization steps. Because EVQ solves the problem of dead units, no *ad hoc* techniques are required.

VII. Conclusion

EVQ is presented here as a natural tool for reducing and filtering very large data sets (see also [45]). It can be used in all applications whose data space has low dimensionality and large cardinality, which benefit from the large speed increase offered by EVQ; in particular, results have shown that EVQ is particularly suitable for reducing and filtering the huge data sets obtained by 3-D scanners. The only parameter that has to be set by the user is the compression rate, which makes EVQ a friendly procedure for untrained users.

Appendix

The procedure to compute the mean distance, $\bar{W}$ [see (22) in Section IV-C] between a sampled point and the nearest RV, given a certain mean number of RVs per box, is outlined here. The formulation leads to nonintegrable equations for dimensions greater than two ($D \geq 3$), while for smaller dimensions an analytical solution is offered.

Let v and w be two points randomly sampled from the unitary hypercube in R^D, I^D. Let $P(r)$ be the probability distribution function of the distance between v and w

$$P(r) = P[||w - v|| \leq r]. \tag{32}$$

The distance can be seen as a random variable Δ, and the actual form of its probability depends on D. Let us randomly sample n vectors in $I^D, W = \{w_1, w_2, \ldots, w_n\}$ and compute their distance from another point v, randomly sampled. Sorting the n distances in ascending order yields the set of n ordered distances $[\Delta_1, \Delta_2, \ldots, \Delta_n]$. It holds that $\Delta_q \leq \Delta_{q+1}$, where Δ_q is the distance between v and w_j if and only if there are exactly $q-1$ points in W closer to v than w_j. It can be demonstrated that the probability distribution of Δ_q is [16]

$$\begin{aligned} P[\Delta_q \leq r] &= P_q(r) \\ &= \sum_{j=q}^{n} \binom{n}{j} P^j(r) \left(1 - P(r)\right)^{n-j}. \end{aligned} \tag{33}$$

The mean value of Δ_q can be calculated as

$$E[\Delta_q] = \int_R r p_q(r) \mathrm{d}r \tag{34}$$

where $p_q(r)$ is the probability density of Δ_q

$$p_q(r) = \frac{\partial P_q(r)}{\partial r}. \tag{35}$$

Since the analytical formulation of $P(\cdot)$ is not available for $D \geq 3$, (33), given that $0 \leq r \leq \sqrt{D}$, (34) can be calculated as

$$\begin{aligned} E[\Delta_q] &= \int_0^{\sqrt{D}} r p_q(r) \mathrm{d}r \\ &= [r P_q(r)]_0^{\sqrt{D}} - \int_0^{\sqrt{D}} P_q(r) \mathrm{d}r \\ &= \sqrt{D} - \int_0^{\sqrt{D}} P_q(r) \mathrm{d}r. \end{aligned} \tag{36}$$

The value of $E(\Delta_q)$ we are interested in is where $n = 2^D \bar{M}$, which will be termed $f_q(\bar{M}, D)$. $f_0(\bar{M}, D)$ is the value of $E(\cdot)$ when only the winning RV is taken into account. $P_q(\cdot)$ can be computed through (33) and integrated into (34) to obtain $f_q(\bar{M}, D)$ through (36). This is the mean distance of the qth closest RV to a sampled point v, where both the RV and the points belong to a unitary cube. The mean distance, when the cube side is L is

$$\bar{W}_q = f_q(\bar{M}, D) L. \tag{37}$$

Acknowledgment

The authors would like to thank the Computer Graphics Laboratory, Stanford University, CA, for making the "dragon," "bunny," and "happy Buddha" data sets available, and P. Grew for reviewing the English of this paper.

References

[1] M. Petrov, A. Talapov, T. Robertson, A. Lebedev, A. Zhilayaev, and L. Polonskiy, "Optical 3D digitizers: Bringing life to the virtual world," *IEEE Comput. Graph. Appl.*, vol. 18, no. 3, pp. 28–37, May/Jun. 1998.

[2] N. A. Borghese, G. Ferrigno, G. Baroni, S. Ferrari, R. Savare, and A. Pedotti, "AUTOSCAN: A flexible and portable scanner of 3D surfaces," *IEEE Comput. Graph. Appl.*, vol. 18, no. 3, pp. 38–41, May/Jun. 1998.

[3] D. Foley, R. VanDam, P. Feiner, and P. Hughes, *Computer Graphics Principles and Practice*. Reading, MA: Addison-Wesley, 1995.

[4] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Trans. Vis. Comput. Graph.*, vol. 5, no. 4, pp. 349–359, Oct./Dec. 1999.

[5] M. Levoy, S. Rusinkiewicz, M. Ginzton, J. Ginsberg, K. Pulli, D. Koller, S. Anderson, J. Shade, B. Curless, L. Pereira, J. Davis, and D. Fulk, "The digital Michelangelo project: 3D scanning of large statues," in *Proc. Siggraph'99*, 1999, pp. 121–132.

[6] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy, "Real-time 3D model acquisition," in *Proc. Siggraph'02*, 2002, pp. 438–446.

[7] H. Hoppe, "Surface reconstruction from unorganized points," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, Jun. 1994.

[8] R. Mencl and H. Müller, "Interpolation and approximation of surfaces from three-dimensional scattered data points," in *Proc. EUROGRAPHICS'98*, 1998, pp. 51–67.
[9] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust," in *Proc. 6th ACM Symp. Solid Modeling Appl.*, 2001, pp. 249–260.
[10] J. Bahrak and A. Fisher, "Parameterization and reconstruction from 3D scattered points based on neural networks and PDE techniques," *IEEE Trans. Vis. Comput. Graph.*, vol. 7, no. 1, pp. 1–16, Jan./Mar. 2001.
[11] A. Wilson and D. Manocha, "Simplifying complex environments using incremental textured depth meshes," in *Proc. Siggraph'03*, 2003, pp. 711–718.
[12] T. R. Jones, F. Durand, and M. Desbrun, "Non-iterative, feature-preserving mesh smoothing," in *Proc. Siggraph'03*, 2003, pp. 943–949.
[13] T. Kohonen, *Self Organizing Maps*. New York: Springer-Verlag, 1995.
[14] A. Baader and G. Hirzinger, "A self-organizing algorithm for multisensory surface reconstruction," in *Proc. Int. Conf. Robot. Intell. Syst. (IROS)*, Sep. 1994, pp. 87–92.
[15] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Comp. Vision*, vol. 1, no. 4, pp. 321–331, 1988.
[16] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proc. Siggraph'94*, 1994, pp. 295–302.
[17] T. Hastie and W. Stuetzle, "Principal curves," *J. Amer. Statist. Assoc.*, vol. 84, pp. 502–516, 1989.
[18] M. Pauly and M. Gross, "Spectral processing of point-sampled geometry," in *Proc. Siggraph'01*, 2001, pp. 211–219.
[19] P. Cignoni, C. Montani, and R. Scopigno, "A comparison of mesh simplification algorithms," *Comput. Graph.*, vol. 22, no. 1, pp. 37–54, 1998.
[20] L. Ibaria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," *Comput. Graph. Forum*, vol. 22, no. 3, pp. 347–352, 2003.
[21] M. Isenburg and S. Gumhold, "Out-of-core compression for gigantic polygon meshes," in *Proc. Siggraph'03*, 2003, pp. 935–942.
[22] M. Pauly, H. M. Gross, and L. Kobblet, "Efficient simplification of point-sampled surfaces," in *Proc. Vis.*, 2002, pp. 163–170.
[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. Siggraph'96*, 1996, pp. 303–312.
[24] G. Roth and E. Wibowoo, "An efficient volumentric method for building closed triangular meshes from 3D image and point data," in *Proc. Graph. Interfaces*, 1997, pp. 173–180.
[25] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface reconstruction algorithm," *Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.
[26] K. L. Low and T. S. Tan, "Model simplification using vertex clustering," in *Proc. ACM Symp. Interactive 3D Graph.*, 1997, pp. 75–81.
[27] B. Fritzke, "Growing cell structures—A self-organizing network for unsupervised and supervised learning," *Neural Netw.*, vol. 7, no. 9, pp. 1441–1460, 1994.
[28] D. Brodsky and B. Watson, "Model simplification through refinement," in *Proc. Graph. Interface 2000*, 2000, pp. 221–228.
[29] E. Shaffer and M. Garland, "Efficient adaptive simplification of massive meshes," in *Proc. Vis.*, 2001, pp. 127–134.
[30] A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition—Part I," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 29, no. 6, pp. 778–785, Dec. 1999.
[31] T. Hofman and J. M. Buhman, "Competitive learning algorithms for robust vector quantization," *IEEE Trans. Signal Process.*, vol. 46, no. 6, pp. 1665–1675, Jun. 1998.
[32] R. Xu; and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.
[33] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1995.
[34] T. M. Martinetz, G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to times-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 558–569, Jul. 1993.
[35] M. Milano, P. Koumoutsakos, and J. Schmidhuber, "Self-organizing nets for optimization," *IEEE Trans. Neural Netw.*, vol. 15, no. 3, pp. 758–765, May 2004.
[36] C. Darken and J. Moody, "Note on learning rate schedules for stochastic optimization," *Adv. Neural Inform. Process. Syst.*, vol. 3, pp. 832–838, 1991.
[37] N. A. Borghese, M. Maggioni, and S. Ferrari, "Multi-scale approximation with hierarchical radial basis functions networks," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 178–188, Jan. 2004.
[38] S. Ferrari, I. Frosio, V. Piuri, and N. A. Borghese, "Automatic multiscale meshing through HRBF networks," *IEEE Trans. Instrum. Meas.*, vol. 54, no. 4, pp. 1463–1470, Aug. 2005.
[39] P. I. Zador, "Asymptotic quantization error of continuous signals and the quantization dimension," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 139–149, Mar. 1982.
[40] Stanford Computer Graphics Lab., Stanford Univ., "The Stanford 3D Scanning Repository" Stanford, CA [Online]. Available: http://www-graphics.stanford.edu/data/3Dscanrep/
[41] C. I. Connolly, "Cumulative generation of octree models from range data," in *Proc. Int. Conf. Robot.*, 1984, pp. 25–32.
[42] A. Djouadi and E. Bouktache, "A fast algorithm for the nearest-neighbor classifier," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 3, pp. 277–282, Mar. 1997.
[43] J. Rossignac and P. Borrel, "Multi-resolution 3D approximation for rendering complex scenes," in *Modelling in Computer Graphics*, B. Falcidieno and T. L. Kunii, Eds. New York: Springer-Verlag, 1993, pp. 455–465.
[44] F. Mulier and V. Cherkassky, "Learning rate schedules for self-organizing maps," in *Proc. 12th IAPR Conf.*, 1994, vol. 2, pp. 224–228.
[45] G. Dong and M. Xie, "Color clustering and learning for image segmentation based on neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 925–936, Jul. 2005.

Stefano Ferrari received the Ph.D. degree in computer engineering from Politecnico di Milano, Milano, Italy, in 2001.

Since 2000, he has been an Assistant Professor at the University of Milano, Crema (CR), Italy. His research interests are in neural networks and soft-computing paradigms and the application to the computer graphics.

Giancarlo Ferrigno received the M.Sc. and Ph.D. degrees in bioengineering from Politecnico di Milano, Milano, Italy, in 1983 and 1990, respectively.

Since 1990, he has been a Researcher, Associate Professor and, since 2001, Full Professor of Bioengineering at Politecnico di Milano. He is an author of more than 60 full papers on international journals and more than ten patents. After several academic positions, he is currently Head of the Doctoral School of Politecnico di Milano, running 31 Ph.D. programs in architecture, engineering, and industrial design. His research interests range from information technology applied to rehabilitation and to motor control to new technologies for *in vitro* cell cultures analysis.

Vincenzo Piuri (S'84–M'86–SM'96–F'01) received the Ph.D. degree in computer engineering from Politecnico di Milano, Milano, Italy, in 1989.

From 1992 to September 2000, he was an Associate Professor in Operating Systems at Politecnico di Milano. Since October 2000, he has been a Full Professor in Computer Engineering at the University of Milano, Milano, Italy. He was a Visiting Professor at the University of Texas at Austin during the summers from 1993 to 1999. His research interests include distributed and parallel computing systems, computer arithmetic, application-specific processing architectures, digital signal processing architectures, fault tolerance, theory and industrial applications of neural networks, intelligent measurement systems, and biometrics. Original results have been published in more than 200 papers in book chapters, international journals, and proceedings of international conferences.

Dr. Piuri is a member of ACM, INNS, and AEI. He was an Associate Editor of the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT and the IEEE TRANSACTIONS ON NEURAL NETWORKS. He was a Vice President for Publications of the IEEE Instrumentation and Measurement Society, Vice President for Members Activities of the IEEE Neural Networks Society, and Member of the Administrative Committee both of the IEEE Instrumentation and Measurement Society and the IEEE Computational Intelligence Society. He is the President of the IEEE Computational Intelligence Society (2006–2007). In 2002, he received the IEEE Instrumentation and Measurement Society Technical Award for his contributions to the advancement of computational intelligence theory and practice in measurement systems and industrial applications.

N. Alberto Borghese (M'97) received the laurea in electrical engineering with full marks and honors from Politecnico of Milano, Milano, Italy, in 1984–1985.

Currently, he is an Associate Professor at the Department of Computer Science of the University of Milano, Milano, Italy, where he teaches the courses of intelligent systems and robotics and digital animation, and is the Director of the Laboratory of Applied Intelligent Systems. He was a Visiting Scholar at Center for Neural Engineering of University of Southern California, Los Angeles, in 1991, at the Department of Electrical Engineering, California Institute of Technology (Caltech), Pasadena, in 1992, and at the Department of Motion Capture, Electronic Arts, Vancouver, Canada, in 2000. He coauthored more than 40 peer-reviewed journal papers and five international patents. His research interests include quantitative human motion analysis and modeling, learning from data, applications to vision graphics, and medical imaging.