Mesh Construction with Fast Soft Vector Quantisation

N. Alberto Borghese[a] and Stefano Ferrari[a,b]

a) Laboratory of Human Motion Analysis and Virtual Reality - MAVR
Istituto Neuroscienze Bioimmagini - C.N.R. via f.lli Cervi 93 - 20090 Segrate (Milano), Italy
borghese@inb.mi.cnr.it – Http://www.inb.mi.cnr.it/Borghese.html

b) Department of Electronics and Information, Politecnico of Milano,
P.zza Leonardo da Vinci 32, 20133 Milano, Italy. sferrari@elet.polimi.it

Abstract

In this paper a method to accelerate Soft Vector Quantisation (VQ), making it a quasi-real time procedure, is described. Through the local analysis of the data density a criterion to set a reasonable value of the parameters and to initialise the position of the Reference Vectors (Hyper-Box preprocessing), allows to cut about 75% of the iterations and to make the computational cost of each iteration, constant, independent of the number of sampled points. Moreover, it makes Soft VQ of possible implementation on parallel machines. Overall the processing time with Hyper-Box pre-processing can be brought down to 3%. This method, in conjunction with Delaunay tessellation, has been extensively applied to the construction of 3D triangular meshes from dense noisy data. Results on the reconstruction of 3D models of Human faces are reported and discussed.

1. Introduction

3D meshes are a representation of 3D models which is gaining popularity in many fields ranging from virtual architectural to image processing, video conferencing, reversing engineering, 3D fax, CAD/CAM and virtual simulations [1]. A common approach is to collect a set of 3D points sampled over the surface through 3D digitisers and to fit a suitable 3D polygonal mesh to the points [2][3]. To get reliable models, one of the main problems to be faced is cleaning the noise introduced by the digitising process. In fact, although a polygonal mesh can be obtained in a very little time through a direct tessellation of the sampled points (e.g. through Delaunay triangulation, [4]), this procedure would maintain all the noise: the reconstructed surface will appear rather wavy (cf. Fig. 2b) with an unrealistic aspect, despite the great number of polygons used. The second problem is data compression: the mesh which has all the sampled data points as its vertexes would be constituted of an exceedingly high number of faces (cf. Fig. 2a), much more than those actually required for an accurate representation of the model. This may easily cause memory and bandwidth overflow, especially when transmission is involved (e.g. VR in collaborative environment).

For these reasons, a host of mesh simplifying techniques have been developed aimed to preserve a high quality appearance with a reduced number of polygons [1]. In these approaches the mesh at full resolution is created first (by connecting all the data points), in a second step the mesh is simplified according to different local error functions. As these approaches do not consider noise on the data points, they are little suitable to clean measurement errors. In a novel solution proposed here, based on Vector Quantisation (VQ), the two steps are interchanged: first the N sampled data points are substituted by a set of M points ($M < N$), called *Reference Vectors* (RV), such that a "representation error" is minimised. In a second step the mesh is created by triangulating this reduced (filtered) set of points. Of the two steps, the far more time consuming is VQ. Since this is a NP-hard problem, faster sub-optimal solutions, which can be obtained through an iterative soft adaptation of the RVs position which exploits the information redundancy contained in the neighbourhood of each data point [5, 6], is generally accepted. We have based our mesh construction algorithm on Neural-Gas VQ procedure (NG) [18] but the reasoning can be extended to other Soft Clustering techniques as well. NG combines a soft-max strategy to move the RVs with a deterministic annealing scheduling for the parameters. However, it has two main drawbacks for applications like mesh construction. Its annealing-based optimisation procedure needs a very long time to converge as small parameters changes should be given at each step, and the parameter setting is critical, even to achieve a sub-optimal result.

In this paper it is shown how, with a simple pre-processing based on the analysis of the sampled points distribution, called Hyper-Box (HB) pre-processing, VQ can be greatly sped up. This allows to reconstruct reliable 3D models from range data in a little time. In particular, 3D models of human faces obtained from the noisy input data obtained from the Autoscan 3D digitiser [4] are reported and discussed.

2. Soft Vector Quantisation and Neural Gas

Vector Quantisation (VQ) techniques [7] are widely used as a tool for data compression in a wide range of applications. They use a set of M Reference Vectors (RV), $\{w_j\}$, to approximate an input data set $\{v_k\}$ of greater

cardinality (N > M). The RVs are positioned such that the following reconstruction error, E(v,w):

$$E(v,w)=\frac{\sum_{k=1}^{N} d\left(v_k, w_j(v_k)\right)^2}{N}=\frac{\sum_{k=1}^{N}\left\|v_k - w_j(v_k)\right\|^2}{N} \tag{1}$$

is minimised. $w_j(v_k)$ is the RV closest to v_k or "winning" RV. To determine the optimal position of the $\{w_j(v_k)\}$. Soft techniques have been proposed to exploit local correlations in the data. Among these Neural-Gas (NG) [5] which proved superior to other methods, has been adopted here. It is a two phases procedure. In the first phase, initialisation, the $\{w_j\}$ are distributed randomly inside the input domain, and in a second phase, optimisation, they are moved around to minimise Eq. (1) according to the following soft-max updating rule:

$$\Delta w_j(t)=\varepsilon\cdot h_{\lambda(t)}\left(k_j\left(\tilde{v}(t),w_j\right)\right)\left(\tilde{v}(t)-w_j\right) \tag{2}$$

The procedure is iterative: at each iteration, t, a sampled point $\tilde{v}(t)$ is randomly extracted from the input data set and the position of all the RVs $\{w_j\}$ is updated according to Eq. (2). $k_j\left(\tilde{v}(t),w_j\right) \in \{0, 1, 2, ..., M\}$ is the ranking of w_j with respect to the actual data vector $\tilde{v}(t)$, measured according to their Euclidean distance, $d_{\tilde{v}(t),w_j}=\|\tilde{v}(t)-w_j\|$, and:

$$h_{\lambda(t)}(.)=e^{-\left(\frac{k_j(\tilde{v}(t),w_j(t))}{\lambda(t)}\right)} \tag{3}$$

is a soft-max weighting function. $\lambda(t)$ and $\varepsilon(t)$ decrease as learning progresses according to:

$$\lambda(t)=\lambda_i\left(\frac{\lambda_f}{\lambda_i}\right)^{t/T_{\max}} \qquad \varepsilon(t)=\varepsilon_i\left(\frac{\varepsilon_f}{\varepsilon_i}\right)^{t/T_{\max}} \tag{4}$$

where T_{max} is the number of iterations. The role of $\lambda(t)$ and $\varepsilon(t)$ is to reduce, respectively, the number of reference points which are effectively displaced at each iteration and the amount of displacement induced by $\tilde{v}(t)$ as optimisation progresses. If, at the beginning, each sampled point induces the movement of most RVs, at the end, h(.) gets close to the Dirac function and it will be meaningfully different from zero only the winning RV. Therefore, the first part of the learning is devoted to a homogenisation of the reference vectors inside the input space; only later, there is a refinement towards the optimal distribution.

3. HyperBox Pre-processing

In the asymptotic condition, the statistical distribution of the RVs is proportional to that of the data vectors according to [5][8]:

$$\rho_w(P)\propto\rho_u(P)^{\gamma} \qquad \text{with} \quad \gamma=D/(D+2) \tag{5}$$

where $\rho_w(P)$ and $\rho_u(P)$ are the density respectively of the RVs and of the sampled points, and D is the dimensionality of the data manifold. In our case D = 3 and γ = 0.6. From this relationship, an efficient initial distribution of the RVs is derived here. Let us call τ the desired compression rate:

$$\tau=\frac{M}{N} \quad \Rightarrow \quad M=\tau N \tag{6}$$

and B the region which contains all the sampled points. Partitioning B into N_H disjointed regions, $\{B_k\}$, gives:

$$\bigcup_{k=1}^{N_H} B_k = B \qquad \text{with} \qquad B_k\cap B_j=\varnothing \quad \forall k\neq j \qquad V_B=\sum_{k=1}^{N_H} V_k \tag{7}$$

where V_B and V_k are the volumes respectively of B and B_k. It results:

$$N=\sum_{k=1}^{N_H} N_k \qquad\qquad M=\sum_{k=1}^{N_H} M_k \tag{8}$$

where N_k and M_k are respectively the number of the sampled points and the RVs inside B_k. Applying (5) to each region B_k, the RVs mean density inside B_k has to satisfy the relationship:

$$\rho'_w(P)\propto\rho'_u(P)^{\gamma} \qquad \Rightarrow \quad \frac{M_k}{V_k}=\beta\left(\frac{N_k}{V_k}\right)^{\gamma} \tag{9}$$

where $\rho'_w(P)$ and $\rho'_u(P)$are the density of sampled points and RVs inside B_k. From Eqs. (7), (8), it holds:

$$\beta=\frac{M}{\sum_{k=1}^{N_H} N_k^{\gamma} V_k^{1-\gamma}} \tag{10}$$

The number of RVs to be inserted inside each region B_k, M_k, is computed through (6), (9) and (10) as:

$$M_k=M\frac{N_k^{\gamma}V_k^{1-\gamma}}{\sum_{k=1}^{N_H} N_k^{\gamma}V_k^{1-\gamma}}=\tau N\frac{N_k^{\gamma}V_k^{1-\gamma}}{\sum_{k=1}^{N_H} N_k^{\gamma}V_k^{1-\gamma}} \tag{11}$$

When the B_k are all Hyper-Boxes (HB) have the same dimensions, Eq. (11) can be simplified as:

$$M_k = M \frac{{N_k}^\gamma}{\sum_{k=1}^{N_H} {N_k}^\gamma} = \tau N \frac{{N_k}^\gamma}{\sum_{k=1}^{N_H} {N_k}^\gamma} \tag{12}$$

which will be termed *partitioning function*. The number of RVs, M_k, to be placed in each HB, B_k, in the initialisation phase does not depend on any length but only on the number of data points sampled in the region. Once the number of RVs per HB has been determined, these have to be placed inside their own Box. The HB data structure is represented as a D-dimensional table, where each entry contains a pointer to the HB structure (which contains all the RVs of the Box). For each sampled point, $v(x_1, x_2, ..., x_D)$, the associated pointer can be easily found in the table through a set of D indices, $(i_1, i_2, \ldots, i_D)$, computed as:

$$i_j = \text{int}\left(\left(x_j - x_{j\min}\right)/L_j\right) \tag{13}$$

where x_{jmin} is the minimum coordinate along the j-th dimension of the input space and L_j the side length of the Box along that direction.

HB pre-processing is particularly suitable to data belonging to low dimensionality. In fact, a proper ordering of the RVs by their distance from a data point v, requires to explore radially the data space starting from v. This defines spherical search regions with variable centers which would not allow to partition the input space efficiently. The use of Boxes is justified by two considerations. First, the probability of considering a RV in Eq. (2) which belongs to the same influence region of v, R(v), but which is further away from v than another RV lying outside R(v), greatly increases with the space dimensionality, and it is quite low for low dimensionalities. Second, as these RVs would lie close to the border of R(v), they are usually far from the actual sampled point, and have a low ranking. It follows that they will receive almost no updating through (2) with no deterioration in the reconstruction.

4. Speeding Up Vector Quantisation and Setting the Parameters

Taking full advantage of HB, a drastic reduction in the processing time has been obtained in two ways. First, when the initial position of the RVs is randomly generated, some of the optimisation time has to be spent to move them around to get close to their optimal position. This phase can be skipped as the RVs are already close to their final optimal position when they are positioned according to the partitioning function (Eq. (12)). Second, there is no need to apply function in Eq. (2) to all the RVs, but only those RVs which are close to the actual sampled point have to be displaced. Moreover, the RVs do not need a large displacement during the optimisation phase as they have to approximate the sampled points in their neighbourhood. These observations allow to drastically reduce the computational time and guide the choice of a proper value for the parameters: ε_i, λ_i and the HBs side L_k.

The HB side, L_k, is a critical parameter: if it were chosen too large, little advantage is gained from HB pre-processing. On the other side, if L_k were too small, the statistical meaning of Eq. (9) could be lost due to an excessive parcellation of the sampled points. For sake of simplicity all the HBs are assumed to have the same L_k, but the reasoning can be easily extended to sides of different lengths. The strategy followed here is to set, on the average, a certain small number of RVs, M_g, is placed inside each Box. To achieve this, an iterative procedure has been adopted. L_k is initially set to: $L_k = \sqrt[D]{V/M} > M_g$ which is the right value for a uniform distribution of the sampled points. The RVs are then distributed inside the HBs according to Eq. (12), and their mean number, Mg, $\overline{M}(L_k)$, is determined. If $\overline{M}(L_k) > M_g$, L_k is increased, viceversa, if $\overline{M}(L_k) < M_g$, until $\overline{M} \approx M_g$. Few iterations are sufficient to converge. In all our experiment, $M_g = 10$ has been assumed.

With HB pre-processing, the RVs are placed inside their right Box already in the initialisation phase. It follows that the distance to reach their final optimal position can be upper limited by the diagonal of the HB. This observation has been used here to speed-up the iterative optimisation phase in two ways. First, in NG, the value of $\lambda(t)$ (Eq. (4)) has to be very large at the beginning to allow all the RVs to move through the data space and get to the region of their final destination [9]. It follows that in the first iteration steps, the vectors tend to cluster around the centroid of the data distribution, and, only in a second phase, they distribute towards their final destination. With HB pre-processing instead, the first phase can be skipped as the RVs have been already distributed close to their final destination. This results in saving about 75% of the iteration steps. Another large saving in computational time is obtained from the analysis of Eq. (2): in NG, the most computationally expensive step is the ordering of the RVs, carried out in each optimisation step. This operation is of the order of M log(M). However, as the RVs are used to approximate the data in their neighbourhood (cf. Eq. (1)), and they lie inside their right Box already from the beginning, the utility of updating their position with the data points which lie far from them (cf. Eq. (2)) can be questioned. This has been incorporated in our algorithm by defining for each sampled data point v, an *Influence*

Region, R(v). This is the region that includes all the RVs which should be updated by a data point v and it has been defined somhow arbitrary as the region constituted of the $N_H = 2^D$ Boxes, (eight in the 3D space) closest to v. As a result, $\Delta w_j(t)$ (in Eq. (2)) and k(.) (in Eq. (3)) are computed only for those RVs which belong to the HB of v and the neighbour ones. Ordering the RVs does not scale anymore with the number of Reference Vectors, M (and therefore with the number of the sampled points), but it remains of the order of $2^D \overline{M}\ \log(2^D \overline{M})$. R(v) furnishes also a criterion to set a reasonable value for λ_i. To the purpose, we assume that at the first iteration (t = 0) only a certain fraction, η, of the RVs inside R(v) should be updated meaningfully. Somehow arbitrarily, a meaningful displacement is considered when $h_\lambda(.) > e^{-1}$ (cf. Eq. (2)). From Eq. (3), it follows that:

$$\lambda_i(v) = \eta \sum_{k:B_k \in R_w(v)} M_k \Big/ N_H \tag{15}$$

$\lambda_i(v)$ is therefore a function of the number of RVs containted in R(v) and it can be different for different HBs. A value of $\eta = 0.1$ has been experimentally proven efficient for a broad spectrum of compression rates ($\tau = 0.01 \div 0.4$).

As λ_f is concerned, $\lambda(t)$ goes to zero asymptotically for $t \to \infty$. Given the limited number of iterations available, for $t = T_{max}$, λ_f will assume a small value, still larger than zero. For the sake of simplicity, this value is assumed a small fraction, θ_λ, of λ_i ($\lambda_f = \theta_\lambda \lambda_i$), with $\theta_\lambda = 0.01$. With this choice, $h_\lambda(k(.))$ for the second closest RV (k = 2), in the last optimisation step, is weighted with e^{-100/λ_i} (Eq. (3)) which can be assumed reasonably close to zero.

The last parameter considered is ε (Eq. (2)). We propose here to determine its values such that the length of the path covered by the reference vector w_j in the optimisation phase, $\Delta w_{j_{TOT}}$, is not larger than the Box diagonal:

$$\Delta w_{j_{TOT}} = \sum_{t=0}^{T_{max}} \Delta w_j(t) < L_k \sqrt{D} \tag{16}$$

Adopting a probabilistic approach [10], it can be shown that:

$$\Delta w_{j_{TOT}} = \frac{1}{M} \sum_{t=0}^{T_{max}} \sum_{p=0}^{(2^D \overline{M})-1} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i} \right)^{1/T_{max}} e^{-p \Big/ \lambda_i \left(\frac{\lambda_f}{\lambda_i} \right)^{1/T_{max}}} E\left[\left\| w_j - v \right\| \,\middle|\, k_j(w_j, v(t)) = p \right] \tag{17}$$

where $E\left[\left\| w_j - v(t) \right\| \,\middle|\, k_j(w_j, v(t)) = p \right]$ is the estimated mean distance between a sampled point, v(t) and the generic j^{th} RV, w_j, when this is the p^{th} closest to v(t). Considering that the contribution of the RVs besides the winning one (p = 0) decays rapidly to zero with the number of iterations, Eq. (17) can be further simplified as:

$$\Delta w_{jTOT} \approx \frac{1}{M} \sum_{t=0}^{T_{max}} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i} \right)^{1/T_{max}} E\left[\left\| w_j - v \right\| \,\middle|\, k_j(w_j, v(t)) = 0 \right] \tag{18}$$

where only the contribution of the winning RV is taken into account. Hypothesising that the mean distance between a RV and the closest sampled point is approximately constant in the optimisation phase (small displacements of w_j), and equal to $\overline{W} = \overline{w}_k(\overline{M}, D) L_k$ [10]:

$$\Delta w_{jTOT} \approx \frac{\overline{W}}{M} \sum_{t=0}^{T_{max}} \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i} \right)^{1/T_{max}} \tag{19}$$

Substituting Eq. (19) into Eq. (16) and giving ε_f as a fraction of ε_i ($\varepsilon_f = \theta_\varepsilon \varepsilon_i$), ε_i can be computed as:

$$\varepsilon_i \cong \frac{M}{\overline{W}} \frac{1 - \vartheta_\varepsilon}{1 - (\vartheta_\varepsilon)^{1 + \frac{1}{T_{max}}}} L_k \sqrt{D} = \frac{M}{\overline{w}_1(\overline{M}, D)} \frac{1 - \vartheta_\varepsilon}{1 - (\vartheta_\varepsilon)^{1 + \frac{1}{T_{max}}}} \sqrt{D} \tag{20}$$

θ_ε has been assumed equal to: $\theta_\varepsilon = \theta_\lambda = 0.01$. As expected, ε_i does not depend on the size of the HBs.

5. Experimental results

This method has been extensively applied to the reconstruction of 3D faces starting from sets of 3D points sampled over them through the Autoscan digitiser [11]. This collects the 3D position of laser spots directed over the surface. The scanning is carried out manually allowing a denser sampling in the most critical regions like the mouth, the eyes and the nose. A typical ensemble of sampled points are the vertexes of the 3D triangular mesh reported in Fig. 2a. This is constituted of 33,622 triangles with 16,851 vertexes. As can be seen from Fig. 2b a direct tessellation of the sampled points produces an unusable result. A much better and smooth result is achieved when the tessellation is performed over the set of the RVs obtained through VQ-NG compression: in Figs. 2c-2d the same face has been reconstructed from only M = 1,685 RVs. All the parameters have been set according to Eqs. (14), (15) and (20) in Section 4. Besides this qualitative evaluation of the reconstruction, a quantitative assessment has been carried out through the *reconstruction error*, E(w,v), defined in Eq. (1), and the number of "*Dead Units"*, N_{du}. The latter are those RVs which at the end of the optimisation phase are never the closest to any sampled point and therefor are

useless or even harmful for the reconstruction quality. For a safe reconstruction Dead Units are usually removed.

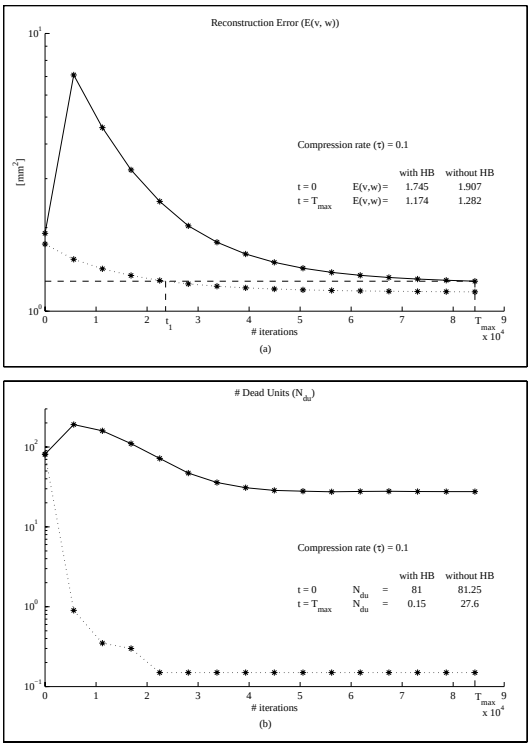

Figure 1. The reconstruction error, E(v,w), and the number of dead units, N_{du}, are reported as a function of the number of iterations for standard VQ-NG (continuous line) and NG with HB (dotted line). $t_1 = 1{,}4 * N = 23{,}629$ iterations .

To get a reliable estimate of E(w,v) and N_{du}, these are averaged over twenty different runs, each with a different initialisation of the RVs position. The mean value of E(w,v) and N_{du}, is reported in Fig. 1 as a function of the number of iterations. It is evident that with HB pre-processing, the convergence is much faster. As in each iteration only few RVs are considered, the processing time per iteration greatly decreases: for example, the total time required by T_{max} = 84,225 (5N) optimisation steps was of 113.8s with HB pre-processing versus the 585.0s required by standard VQ-NG on a SGI Indigo2, R4400, 250Mhz machine. It should be remarked that both the final Reconstruction Error and the final Number of Dead Units with HB-NG are much lower ($1.17mm^2$ versus $1.28\ mm^2$ and 0.15 versus 27.6 respectively). More interestingly the same figures of standard NG, are obtained with HB-NG only after t_1 = 23,229 (1.4N) iterations which require only 20s. Therefore HB allows to increase the algorithm speed by a factor close to sixty times.

To test the adequacy of the parameters ε_i, and λ_i, E(v,w) and N_{du} have been computed when VQ-NG with HB has been run using pairs of values generated randomly in the range of 0.1 ÷10 times the value set according to Section 4. Results have shown that the obtained error is comparable with the smallest ones [10].

HB pre-processing is also useful to obtain models with different compression rates and therefore get different Level of Details (LOD) in different regions of the input space. Although Autoscan already allows to sample more points in the most difficult regions, an even more dense tessellation may be required there.

This is easily accomplished with HB partitioning. The user has only to define manually the regions where the higher compression rate is required. First, a VQ of the input data set is carried out at the lowest compression rate, τ_1, obtaining a first set of RVs, W_1. Then, the RVs of W_1 which belong to the manually selected regions are removed and substituted with a second set of RVs, W_2, such that the desired higher compression rate, τ_2, is achieved there. HB-VQ is run again only for the set W_2. No modification is required to the RVs in the other regions. The overall procedure work therefore locally on the data set.

6. Conclusion

VQ has been historically introduced as a technique for data lossy compression [10]; it has been used here to *loose* the digitisation noise as well as to reduce the number of vertices. In conjunction with Delauney tessellation it is able to produce a mesh of very high quality with few well-placed vertexes. The introduction of HB pre-processing greatly reduced the computational time making VQ suitable to those applications which require the construction of meshes. Its ability to easily handle a multi-resolution reconstruction and its simplicity make VQ with HB a powerful tool to be used with modern 3D digitisers. However, the method is of general interest for al the clustering problems in low dimensional spaces.

References

[1] P.S. Heckbert and M. Garland, Survey of Polygonal Surface Simplification Algorithms. *Tech. Rep.,* Carnegie Mellon University, 1997.

[2] R. Mencl and H. Müller, "Interpolation and Approximation of Surfaces from Three-Dimensional Scattered Data Points," EUROGRAPHICS '98*, STAR - State of The Art Report*, pp. 51-67, 1998.

[3] M. Petrov, A. Talapov, T. Robertson, A. Lebedev, A. Zhilayaev and L. Polonskiy, "Optical 3D Digitisers: Bringing Life to the Virtual World," *IEEE Computer Graphics*, vol. 18, no. 3, pp. 28-37, 1998.

[4] C. Oblonšek and N. Guid, "A Fast Surface-Based Procedure for Object Reconstruction from 3D Scattered Points," *Computer Vision and Image Understanding*, vol. 69, no. 2, pp. 185-195, 1998.
[5] T.M. Martinetz, G. Berkovich, K.J. Schulten, "Neural-Gas Network for Vector Quantization and its Application to Times-Series Prediction," *IEEE Trans. on Neural Networks*, vol. 4, no. 4, pp. 558-569, 1993.
[6] N.A. Karajannis, "An Axiomatic Appraoch to Soft Learning Vector Quantisation and Clustering," *IEEE Trans. Neural Networks*, in press.
[7] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publisher, 1995.
[8] P.I. Zador, "Asymptotic quantisation error of continuous signals and the quantization dimension," *IEEE Trans. Inform. Theory*, vol. IT-28, no. 2, pp 139-149, 1982.
[9] K. Rose, F. Gurewitz and G. Fox, "Statistical mechanics and phase transitions in clustering," *Physical Rev. Lett.*, vol. 65, no. 8, pp. 945-948, 1990.
[10] S. Ferrari, G. Ferrigno and N.A. Borghese, "Vertex Clustering and Data Filtering with Vector Quantisation," *Tech. Rep.* 2000-2*,* Dept. Electronics and Informatics, Politecnico of Milano, 2000.
[11] N.A. Borghese, G. Ferrigno, G. Baroni, S. Ferrari, R. Savare and A. Pedotti, "AUTOSCAN: a Flexible and Portable Scanner of 3D Surfaces," *IEEE Computer Graphics and Applications*, vol. 18, no. 3, pp. 38-41, 1998.

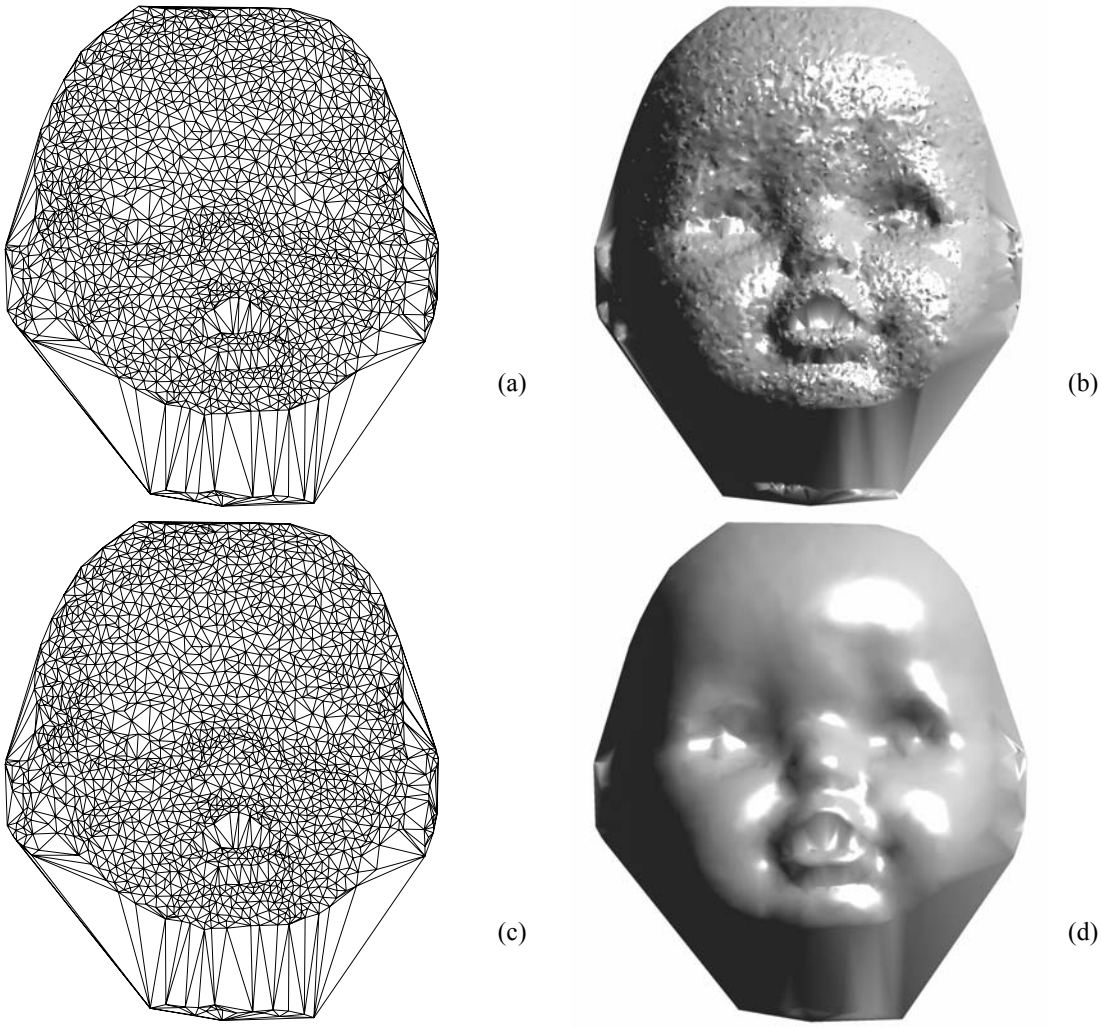

Figure 2. M constructed tessellating the set of the N = 16,851 sampled (33,622 triangles): wire frame (a) and Gouraud shaded (b), and the set of M = 1,685 RVs (3,361 polygons) obtained from NG with HB ($\tau = 0.1$, T_{max} = 3 x N = 50,553, L_k = 13.5mm, ε_i = 0.9990, $\eta = 0.1$, $\overline{M}$ =9.912, Number of HBs: $10 \times 12 \times 7$): wire frame (c) and Gouraud shaded (d).

(a) ((