Adaptive Polygonal Mesh Simplification With Discrete Centroidal Voronoi Diagrams

Sébastien Valette, Ioannis Kompatsiaris, Jean-Marc Chassery

Adaptive Polygonal Mesh Simplification With Discrete Centroidal Voronoi Diagrams

Sébastien Valette
Informatics and Telematics Institute
Thessaloniki
Greece
Email: valette@iti.gr

Ioannis Kompatsiaris
Informatics and Telematics Institute
Thessaloniki
Greece
Email: ikom@iti.gr

Jean-Marc Chassery
LIS
Grenoble
France
Email: jean-marc.chassery@lis.inpg.fr

***Abstract*— In this paper, we propose an adaptive polygonal mesh coarsening algorithm. This approach is based on the clustering of the input mesh triangles, driven by a discretized variationnal definition of centroidal tesselations. It is able to simplify meshes with high complexity i.e. meshes with a large number of vertices and high genus. We demonstrate the ability our scheme to simplify meshes according to local features such as curvature measures. We also introduce an initial sampling strategy which speeds up the algorithm, an on-the-fly checking step to guarantee the validity of the clustering, and a post-processing step to enhance the quality of the approximating mesh. Experimental results show the efficiency of our scheme both in terms of speed and visual quality.**

I. Introduction

3D meshes are used in a vast majority of 3D applications such as Computer Aided Design, Medical Imaging, Virtual Reality and Video Games. 3D models are constructed by designers, or can be generated automatically from real objects using 3D scanners. Nowadays, the models can have up to several million or even billion elements (vertices) and sometimes need a preprocessing step to match a given application requirements. The processing step sometimes consists in reducing the complexity of the mesh (in terms of number of elements, topology or smoothness) to accelerate rendering or transmission, increasing its elements aspect ratio (for accurate finite elements analysis), or remeshing (to meet a given connectivity constraint). As a consequence, automatic or semi-automatic geometry processing becomes increasingly important for interactions between various applications. We propose in this paper an adaptive surface mesh coarsening algorithm, which resamples the surface to a mesh with many fewer elements than the original mesh. Our approach extends the work of Valette and Chassery [1] to non-uniform Centroidal Voronoi Diagrams. The complexity of our algorithm (in terms of calculations and memory requirements) is low, allowing the processing of large meshes, as shown in the results section, where processing meshes with up to 1 million triangles within a minute.

II. Previous Work

Coarsening a mesh consists in resampling the original surface with a lower number of vertices. The number of existing approaches for mesh resampling is very high. For simplicity, we can split the existing approaches in three categories: refinement, decimation, or direct approaches, which will be described more precisely, due to promising recent advances.

Refinement approaches [2], [3], [4], approximate the original surface with a coarse mesh which is iteratively refined until a given precision is reached.

Decimation approaches, such as [5], [6], [7], [8], [9] also process the mesh iteratively, constructing several resolution levels. For a given mesh, several resolution levels are constructed by means of elementary simplifications (edge collapse or face merge, as an example), until the approximation error reaches a user-defined maximum. A survey of coarsening approaches is made in [10].

In opposition to the first two categories, direct approaches (or remeshing approaches) compute a mesh with a given number of elements or approximation error budget in a single resolution way. Some approaches remesh the original surface in a global parametric space [11], [12], [13], [14] They provide good results, but are limited in practice by the parametrization step, involving heavy calculations and numerical instability. To overcome these problems, some approaches [15], [16] were proposed, involving local parametrization and optimization of the remeshed model. Other works [17], [18] distribute new vertices directly on the original surface mesh, to build a new tessellation which can be further optimized.

In [19] and [20] the authors propose to remesh the model using geodesic distances: the new vertices are created using geodesic front propagation. Note that the vertices distribution can also be adapted to local curvature.

Note that remeshing approaches allow the construction of meshes with as many vertices as wanted. Indeed, mesh coarsening is not the main goal of remeshing approaches, as they permit other improvement (in terms of triangles aspect ratio) and shape adapted remeshing (e.g. adaption of the sampling according to the local curvature).

In [21] and [1], the triangles of the input mesh are clustered and a new coarsened mesh is build based on the clustering. These approach are efficient when the number of triangles of the output mesh is much lower than the number of triangles of the input mesh. The approach of Cohen-Steiner et al. [21] aims to create approximation-efficient meshes, whereas the approach of Valette and Chassery [1] aims to create uniform

output triangluations.

In [22], Nooruddin and Turk propose to simplify the mesh topology by a volumetric approach: the mesh is converted to a volumetric representation (voxels) which topology is simplified by means of morphological operations. Afterwards, the volume is re-converted to a polygonal model which is further simplified.

We can also mention out-of-core approaches for coarsening [23], [24], used for large models which do not fit entirely inside the computer RAM.

III. Our Approach

In this paper, we propose an algorithm for mesh coarsening, which produces adaptive triangulations. Our approach can be applied to manifold meshes with any genus and any number of holes. The first step is a clustering of the mesh cells (triangles) into a discrete Centroidal Voronoi Diagram (CVD), according to a desired density function.

The second step consists in replacing each cluster by a single vertex, and constructing the triangulation according to the clusters adjacency relations. We assume that the subsampling factor of the coarsening is high i.e. the ratio between the number of original vertices and the number of vertices of the resulting mesh is high. In this paper, we display results with meshes which number of vertices is at least divided by 20. Those high subsampling ratios enable us to formalize a clustering approach, noticing that even if the input surface is a discrete set (the union of several polygons), it can be seen as a continuous space, as the input polygons will be small compared to the output ones. Note that our approach simultaneously simplifies the mesh geometry and its topology, and thus can be seen as a topological and geometric filter.

IV. Technical Background

In this section, we make an overview of Centroidal Voronoi Diagrams (CVD) in terms of energy minimization, both for their continuous and discretized versions. Supplementary details can be found in [25] and [1]

A. Voronoi Diagrams

Given an open set Ω of $\mathbb{R}^a$, and n different sites (or seeds) $z_{i;i=0,1,\ldots,n-1}$, the Voronoi Diagram can be defined as n distinct regions C_i such that:

$$C_i = \{w \in \Omega | d(w, z_i) < d(w, z_j) j = 0, 1, \ldots, n-1, j \neq i\} \quad (1)$$

where d is a function of distance. These diagrams are well known in the literature. The dual of a Voronoi Diagram is a Delaunay triangulation, which has the property that the outcircle of every triangle does not contain any other site.

B. Centroidal Voronoi Diagrams

A Centroidal Voronoi Diagram is a Voronoi Diagram where each Voronoi site z_i is also the mass centroid of its Voronoi Region:

$$z_i = \frac{\int_{C_i} x.\rho(x)dx}{\int_{C_i} \rho(x)dx} \quad (2)$$

where $\rho(x)$ is a density function of C_i

Moreover, Centroidal Voronoi Diagrams minimize the energy given as:

$$E = \sum_{i=0}^{n-1} \int_{C_i} \rho(x)\|x - z_i\|^2 dx \quad (3)$$

Constructing a Centroidal Voronoi Diagram (CVD) can be done using K-means clustering and Lloyd's relaxation method [26], as an example. CVDs have intrinsic properties which make them optimal for a wide range of applications[25] because they optimize the compactness of the created Voronoi Regions (see equation 3).

C. A discretized Central Voronoi Diagram definition

In [1], a discrete definition of CVD is given. Ω is no longer a continuous space, but a polygonal mesh M. Subsequently we will only consider triangular meshes, but extension to the polygonal case is straightforward. The discrete definition of the CVD falls into this constraint: the boundaries of each Voronoi Region C_i is a subset of the edges of M. As a consequence, a Voronoi region is the union of several mesh triangles T_j. Note that with such restriction, the regions C_i are no more Voronoi regions in the strict sense. Constructing such diagram comes now as a clustering problem: we want to merge the Triangles T_j of the mesh M into n clusters (which look like Voronoi regions) C_i, each cluster having only 1 connected component.

D. Discrete minimization

The discrete definition of the CVD consists in reformulating the energy term E (equation 3) and trying to find the clustering minimizing E, which is now defined by:

$$E = \sum_{i=0}^{n-1} \left(\sum_{T_j \in C_i} \int_{T_j} \rho(x)\|x - z_i\|^2 dx \right) \quad (4)$$

It is easy to demonstrate that the individual contribution of each triangle T_j to the global energy term E can be simplified to:

$$\int_{T_j} \rho(x)\|x - z_i\|^2 dx = \rho_j \|z_i - \gamma_j\|^2 + Aj \quad (5)$$

where

$$A_j = \int_{T_j} \rho(x)\|x - \gamma_i\|^2 dx \quad (6)$$

$$\rho_j = \int_{T_j} \rho(x)dx \quad (7)$$

$$\gamma_j = \frac{1}{\rho_j} \int_{T_j} \rho(x)xdx \quad (8)$$

A_j depends only on the geometry of T_j and on the density function $\rho(x)$, ρ is the global weight of T_j according to $\rho(x)$ and γ_j is the center of gravity of T_j. Note that in [1] the term A_j was omitted, which was the first approximation made by

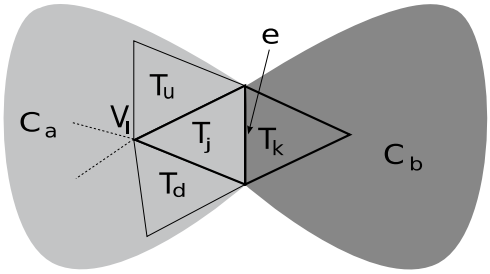

Fig. 1. Local neighbouthood used for the clustering evolution. The triangles Tj and T_k originally belong to the clusters C_a and C_b, and the test consists in checking if changing the configuration (puting T_j in C_b or T_k in C_a will decrease the global energy term.

Valete and Chassery. But we will prove that excluding A_j from the computation does not influence the quality of the results. By suming each triangle individual contribution to E, following equation 5, it comes:

$$E = \sum_{i=0}^{n-1} \left(\sum_{C_j \in V_i} \rho_j \|z_i - \gamma_j\|^2 \right) + \sum_j A_j \qquad (9)$$

which proves that whatever the clusters configuration consists in, the contribution of the terms A_j will always be the same. We can then safely omit their computation to minimize a new energy term:

$$F = \sum_{i=0}^{n-1} \left(\sum_{T_j \in C_i} \rho_j \|\gamma_j - z_i\|^2 \right) \qquad (10)$$

It is possible to efficiently minimize this energy term with an iterative algorithm that updates the clustering according to tests on the boundaries between the different clusters. Assuming that a given edge e is on the boundary between two clusters C_a and C_b (see figure1), e has two adjacent triangles T_j and T_k belonging respectively to C_a and C_b, three values of F are computed:

- F_{init} (the initial configuration) : T_j belongs to C_a and T_k belongs to C_b.
- F_1 (C_a grows and C_b shrinks) : both T_j and T_k belong to C_a.
- F_2 (C_a shrinks and C_b grows): both T_j and T_k belong to C_b.

the clusters configuration is updated according to the lowest computed energy term between F_{init},F_1 and F_2. By looping in the boundary edge set (the set of edges between two different clusters), we iteratively minimize F. As F is always positive and each local modification reduces F, the convergence of the algorithm is guaranteed.

Moreover, it was shown that when processing one edge, the comparison of the three values of F for the three cases is not needed, and instead of computing F, we only need to compute:

$$L = -\frac{\left\| \sum_{T_j \in C_a} \rho_j \gamma_j \right\|^2}{\sum_{T_j \in C_a} \rho_j} - \frac{\left\| \sum_{T_j \in C_b} \rho_j \gamma_j \right\|^2}{\sum_{T_j \in C_b} \rho_j} \qquad (11)$$

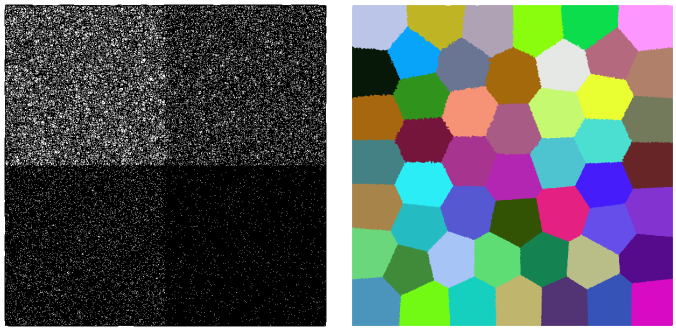

Fig. 2. A triangular plane falls into 4 parts having different vertices density (left). Despite the sharp density changes, the clustering (right) remains uniform over the plane

for each of the three cases. A fast and efficient computation is possible by storing the values $\sum_{T_j \in C_a} \rho_j \gamma_j$ and $\sum_{T_j \in C_a} \rho_j$ in accumulator arrays.

Figure 2 shows an example of clustering on a randomly triangular plane. The original plane (right) consists in 4 areas with a different sampling density. The four regions contain respectively (from top left to botom right) 10000, 20000, 40000 and 80000 vertices. Notice that despite the sharp density changes in the original sampling, the resulting clustering (right) is still uniform.

V. Our approach

A. Curvature indicator as density function

In sharp contrast with [1], we propose to cluster the mesh triangles in a non-uniform way. Adaptivity is a key feature for many applications, when some parts of the mesh must contain more vertices than other parts. As an example, it is well known that approximating schemes must provide a high vertices budget to regions with high curvature features. In this paper, we propose to mimick an approximating scheme, by giving to each triangle T_j a weight ρ_j according to local curvature measures. As we aim at applying our scheme to very complex meshes, the curvature measure has to be very robust against bad sampling conditions that may be encountered when processing such models. We propose to compute a curvature indicator with such properties. To do so, we calculate the matrix $A_{2\times 2}$ of the Weingarten map of the surface using a polynomial fitting of the local neighbourhood of each triangle, as explained in [27]. The local principal curvatures $E_{j,1}$ and $E_{j,2}$ are the eigenvalues of A. In all our experiments, we chose the neighbourhood of a triangle to be the union of the 2-ring of its three vertices. Finally, we set each triangle weight ρ_j to:

$$\rho_j = |T_j| \left(\sqrt{E_{j,1}^2 + E_{j,2}^2} \right)^{\gamma} \qquad (12)$$

where $|T_j|$ is the area of T_j and γ is a gradation parameter which controls the curvature adapted bahaviour of our scheme. In spirit with [12], setting $\gamma = 0$ will produce uniform clustering whereas higher values of γ will give more and more importance to the regions with high curvatures. Figure 3 shows the curvature indicator computed on the Happy Budda model

Fig. 3. Curvature indicator for the Happy Buddha model. Left: original model, right: curvature indicator (higher values are brighter)

with $\gamma = 1.5$. Note that as expected, the regions with high curvature depict higher values than relatively flat regions.

B. Efficient initial sampling

To begin the clustering process, an initial sampling step must be done, to associate at least one triangle to each cluster. In [1], the initial sampling is done by randomly picking one triangle of the mesh for each cluster. As a consequence, the clusters will be equally distributed over the original mesh. This is convenient for uniform coarsening, as the goal is to build clusters with the same surface. But this is not appropriate for adaptive clustering, since the regions with higher density should contain more clusters than regions with low density. Indeed, if we randomly distribute the clusters during the energy minimization process, the clusters in low density regions will slowly move towards regions with high density, resulting in very low convergence speed. To alleviate this problem, we propose to distribute the clusters according to the density function. To do so, we first compute a global average cluster density:

$$D = \frac{1}{n} \sum_j \rho_j \tag{13}$$

where n is the number of desired clusters. This density corresponds to the average cumulated density that each cluster should have at the end of the clustering process. We try to initialize the clustering with clusters having such a cumulated density. For each cluster, we randomly pick a free triangle T_f (a triangle which was not previously associated to any cluster) and grow a region around T_f until its cumulated density reaches D. If at some point some clusters remain to be initialized and no more triangles are free (which can happen, as we operate on a discrete set), we randomly pick one non-free triangle for each non initialized cluster. This initial sampling strategy was proven to be efficient in accelerating the convergence of the approach. As an example, clustering the David model to 20k vertices with randomly picked initial clusters took 44 seconds, and clustering with our initialization algorithm took only 31 seconds.

C. Convergence issues

The clustering is based on the minimization of a positive energy term defined on a discrete set. Each clustering evolution decreases the energy term, and the convergence of the clustering is then theoretically guaranteed. In practice, some numerical issues can appear. The computation of a curvature indicator gives a different weight ρ_j to each triangle T_j. Very low values of ρ_j could prevent the clustering algorithm convergence, as the acumulator arrays might not have the required precision range. This can also happen when one or several triangles of the mesh have null or almost null area: they could move from one cluster to an other one without any noticeable consequences for the energy term, preventing the clustering step to converge. To solve those problems, we slightly modify the approach : We first compute the average triangle weight ρ_{av}, and a threshold $Th_\rho = 10^{-5}\rho_{av}$. All the clusters weights below Th_ρ are set to Th_ρ, which will give a significant value to the clusters having almost null weight.

D. Guaranteed valid clusters

Once the clustering done, each cluster has to be to be a connex set of cells. One way to respect this constraint, after the convergence algorithm, is to "clean" the clusters falling into several connected components, and to restart the clustering step again. These two steps can be repeated until the constraint is respected. Although this approach works well in practice, there is no theoretical proof that it will always succeed, and running alternatively the clustering step and the cleaning step can be computationally expensive. To overcome these drawbacks, we run a three step algorithm. First, we run the clustering algorithm as described above. Afterwards, we run the cleaning step. If some cleaning was done (meaning that some clusters did not respect the connexity constraint), we then re-apply the clustering step, with an additionnal embedded checking step. Figure 1 displays a local boundary context used during clustering evolution. Each time a triangle T_j has to move from one cluster C_a to an other cluster C_b, we check if this modification does not break the connexity property of the cluster C_a. To do so, we perform two verifications:

- if one of the two neighbour triangles T_u or T_d does not belong to C_a, then associating T_j to C_b will not affect the connexity of C_a, and the modification is allowed
- if both triangles T_u and T_d belong to C_a, there still must be a path between them to keep C_a connex. A sufficient condition (but not mandatory) is to check that all the triangles in the 0-ring of the vertex V_l belong to C_a. If this this condition is not true, we forbid the modification.

With this constraint, after the second clustering step, all the clusters are guaranteed to have only one connex component. Note that we do not take this constraint into account during the first minimization process, as it would significantly decrease the speed of the algorithm, and it could prevent the removal of the input mesh topological noise.

E. Meshing and geometry enhancement

Once the clustering step is finished, the output mesh connectivity can be build, based on the clustering. We create one vertex for each cluster. We also create output triangles whenever an input mesh vertex is adjacent to 3 or more different clusters. Valette and Chassery [1] propose to set the output vertices geometry to the centroid of their respective clusters, followed by a projection on the original mesh. We propose an other approach: instead of projecting the vertices on the original mesh, a more efficient is to use a quadric-based placement, in spirit with [28]: As explained in [5], each triangle T_j of the input mesh can be associated with a 4×4 quadric matrix Q_j which represents a bilinear form usefull to compute the weighted quadratic distance between any point and the plane containing T_j. The quadrics of all the triangles belonging to a given cluster C_i are sumed together, and an "optimal" vertex position for C_i is computed, according to this sum, in accordance with [28].

Figure 4 shows a comparison between the approach proposed in [1] (left) and the quadrics-based approach (right), on the Bunny model uniformly coarsened to 600 vertices. It is clear that the quadric approach performs better approximation, most significantly for the Bunny ears.

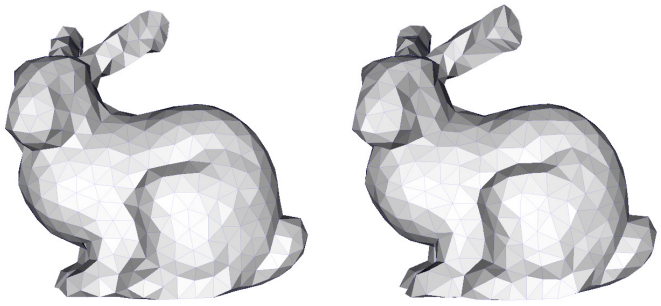

Fig. 4. Comparison between 2 approaches for vertices placement. Left: approach from [1]. Right : quadric based placement

VI. Results

We have tried our approach on a large set of reference meshes having up to several hundred thousands of vertices. We insist on the fact that many previous approaches were unable to process such big meshes, due to their complexity (high number of vertices, presence of topological noise). Moreover, the David model has 65 non-manifold edges, but this did not cause any problem to our algorithm.

Figure 5 shows the curvature indicator computed for the Bunny model, and the corresponding coarsened version to 3k vertices. Note that the Input Bunny only has 70k vertices. As our approach works well when the number of output vertices is much lower than the number of input vertices, we first subdivided the Bunny model with a Butterfly subdivision scheme, in order to obtain a 280k vertices of the Bunny, well suited for our algorithm. Figure 6 shows the results obtained when coarsening the Turbine blade model to 20k vertices. Note that the topology of the original model is well respected, as the small holes remain intact.

Figure 7 shows a coarsened version of the david model with 20k vertices (γ=1). Figure 8 is a close-up view of the original and coarsened versions of the David model to 20k vertices (γ=0 and 1.5). Figure 9 shows two versions of the Dragon model coarsened to 5k vertices (γ=0 and 2). Figure 10 shows four coarsened versions of the original Happy Buddha model with 10k vertices (γ: 0, 0.5, 1 and 1.5).

In all the presented results, when $\gamma > 0$, the vertices are clearly concentrated in regions of high curvature. As a consequence, the sharp details of the original mesh are well preserved.

Table I shows results obtained for all the models presented in this paper. The first column is the number of vertices of the original mesh. The second one is the number of vertices of the coarsened mesh. We computed two different objective criteria to measure the quality of the output meshes. One is based on the angles of the resulting triangles (the minimal angle $\angle_{min}$, the average minimal angle $\angle_{av}$, and the percentage of angles which are less than 30 degrees $\angle < 30^o$) and the second one is based on the triangles shape (minimal quality Q_{min}, average quality Q_{av}, which ranges between 0 and 1, as defined in [29]). The processing times were measured on a computer with an Intel Pentium-M 1.5Ghz and 512 Mb RAM. Timings are given only for the curvature indicator computation and for the clustering step, as the timings for the meshing step are negligible. The last column gives memory consumption of our approach for all the tested meshes

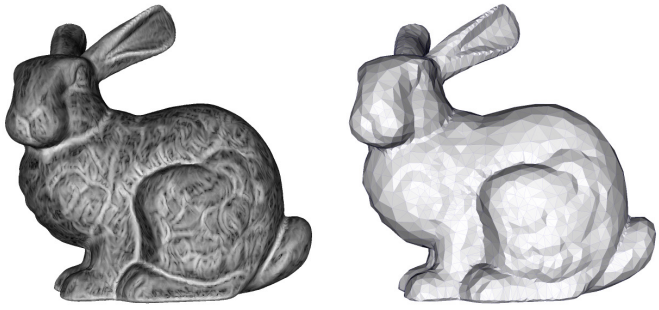

Fig. 5. Processing the Stanford Bunny. Left : curvature indicator ($\gamma = 2$). Right : coarsened model with 3k vertices

VII. Conclusion and perspectives

We proposed in this paper an efficient algorithm for adaptive mesh coarsening. Based on a discrete definition of Centroidal Voronoi Diagrams, the clustering can be fastly computed with low memory requirements, and is therefore able to process

Model	#v (original)	#v2 (coarsened)	γ	$\angle_{min}$ (deg)	$\angle_{av}$ (deg)	$\angle < 30^o$ (%)	Q_{min}	Q_{av}	curvature (s)	clustering (s)	Memory MB
Blade	883k	20k	1.5	0.08	36.7	10	0.02	0.69	142	108	308
Buddha	543k	10k	0	2.1	46.4	1.12	0.05	0.82	0	36	207
Buddha	543k	10k	0.5	0.49	39.1	7.32	0.01	0.73	92	35	207
Buddha	543k	10k	1	0.49	39.1	7.32	0.01	0.73	92	48	207
Buddha	543k	10k	1.5	0.49	39.1	7.32	0.01	0.73	92	45	207
David	507k	20k	0	0.85	46.8	1.1	0.02	0.83	0	24	161
David	507k	20k	1	0.24	40.2	6.4	0.01	0.74	82	31	161
David	507k	20k	1.5	0.24	40.2	6.5	0.01	0.74	82	31	161
Dragon	437k	5k	0	3	47	1.2	0.09	0.84	0	22	166
Dragon	437k	5k	2	0.77	40	6.5	0.02	0.73	73	31	166
Bunny	70k	3k	1	1.26	38.1	8.1	0.04	0.71	21	8	68

TABLE I

TIMINGS AND QUALITY MEASUREMENTS FOR THE TESTED MESHES

large triangular meshes. In the future, we plan to give this approach an anisotropic behaviour, for fast and efficient approximation of large meshes.

ACKNOWLEDGEMENTS

This work was founded by the Marie Curie Fellowship Association. The models presented in this paper are courtesy of Stanford University and the Digital Michelangelo Project [30].

REFERENCES

[1] S. Valette and J.-M. Chassery, "Approximated centroidal voronoi diagrams for uniform polygonal mesh coarsening," *Computer Graphics Forum (Eurographics 2004 proceedings)*, vol. 23, no. 3, pp. 381–389, 2004.

[2] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," *Computer Graphics*, vol. 29, no. Annual Conference Series, pp. 173–182, 1995.

[3] H. Delingette, M. Hebert, and K. Ikeuchi, "Shape representation and image segmentation using deformable surfaces," *Image and Vision Computing*, vol. 10(3), pp. 132–144, April 1992.

[4] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "Maps: Multiresolution adaptive parameterization of surfaces," *ACM SIGGRAPH Conference Proceedings*, pp. 95–104, 1998.

[5] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," *Computer Graphics*, vol. 31, no. Annual Conference Series, pp. 209–216, 1997.

[6] H. Hoppe, "Progressive meshes," *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 99–108, 1996.

[7] G. L. Miller, D. Talmor, and S. H. Teng, "Optimal good-aspect-ratio coarsening for unstructured meshes," in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1997.

[8] J. Rossignac and P. Borrel, "Multi-resolution 3d approximations for rendering complex scenes," in *Geometric Modeling in Computer Graphics*, B. F. Springer Verlag and T. Kunii, Eds., 1993, pp. 455–465.

[9] S. Valette and R. Prost, "Wavelet-based multiresolution analysis of irregular surface meshes," *IEEE Transactions on Visualization ans Computer Graphics*, vol. 10, no. 2, pp. 113–122, 2004.

[10] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," *SIGGRAPH 97 courses notes*, 1997.

[11] P. Alliez, M. Meyer, and M. Desbrun, "Interactive Geometry Remeshing," *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, vol. 21(3), pp. 347–354, 2002.

[12] P. Alliez, É. C. de Verdière, O. Devillers, and M. Isenburg, "Isotropic surface remeshing," in *Proceedings of Shape Modeling International*, 2003, pp. 49–58.

[13] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun, "Anisotropic polygonal remeshing," *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, pp. 485–493, 2003.

[14] X. Gu, S. Gortler, and H. Hoppe, "Geometry images," *ACM SIGGRAPH Conference Proceedings*, pp. 355–361, 2002.

[15] V. Surazhsky, P. Alliez, and C. Gotsman, "Isotropic remeshing of surfaces: a local parameterization approach," in *Proceedings of 12th International Meshing Roundtable*, 2003.

[16] V. Surazhsky and C. Gotsman, "Explicit surface remeshing," in *Proceedings of the ACM/Eurographics Symposium on Geometry Processing*, June 2003.

[17] J. Lötjönen, P.-J. Reissman, I. E. Magnin, J. Nenonen, and T. Katila, "A triangulation method of an arbitrary point set for biomagnetic problems," *IEEE Transactions on Magnetics*, vol. 34, no. 4, pp. 2228–2233, 1998.

[18] G. Turk, "Re-tiling polygonal surfaces," *Computer Graphics*, vol. 26, no. 2, pp. 55–64, 1992.

[19] G. Peyré and L. Cohen, "Geodesic remeshing using front propagation," in *IEEE workshop on Variational, Geometric and Level Set Methods in Computer Vision*, 2003.

[20] O. Sifri, A. Sheffer, and C. Gotsman, "Geodesic-based surface remeshing," in *International Meshing Roundtable*, 2003.

[21] D. Cohen-Steiner, P. Alliez, and M. Desbrun, "Variationnal Shape Approximation," *ACM Transactions on Graphics. Special issue for SIGGRAPH conference*, 2004.

[22] F. Nooruddin and G. Turk, "Simplification and repair of polygonal models using volumetric techniques," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, April-June 2003.

[23] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink, "Large mesh simplification using processing sequences," in *IEEE Visualization conference proceedings*, 2003.

[24] J. Wu and L. Kobbelt, "A stream algorithm for the decimation of massive meshes," *Graphics Interface Proceedings*, pp. 185–192, 2003.

[25] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tesselations: applications and algorithms," *SIAM Review*, no. 41(4), 1999.

[26] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Trans. Inform. Theory*, vol. 28, pp. 129–137, Mar. 1982.

[27] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22(2), pp. 121–146, 2005.

[28] P. Lindstrom, "Out-of-core simplification of large polygonal models," in *Siggraph 20000, Computer Graphics Proceedings*, K. Akeley, Ed. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000, pp. 259–262.

[29] P. Frey and H. Borouchaki, "Surface mesh evaluation," in *6th International Meshing Roundtable*, 1997, pp. 363–374.

[30] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, , and D. Fulk, "The digital michelangelo project," in *SIGGRAPH Conference Proceedings*, 2000, pp. 131–144.

Fig. 6. Coarsened version of the Turbine Blade Model (20k vertices, $\gamma = 1.5$)

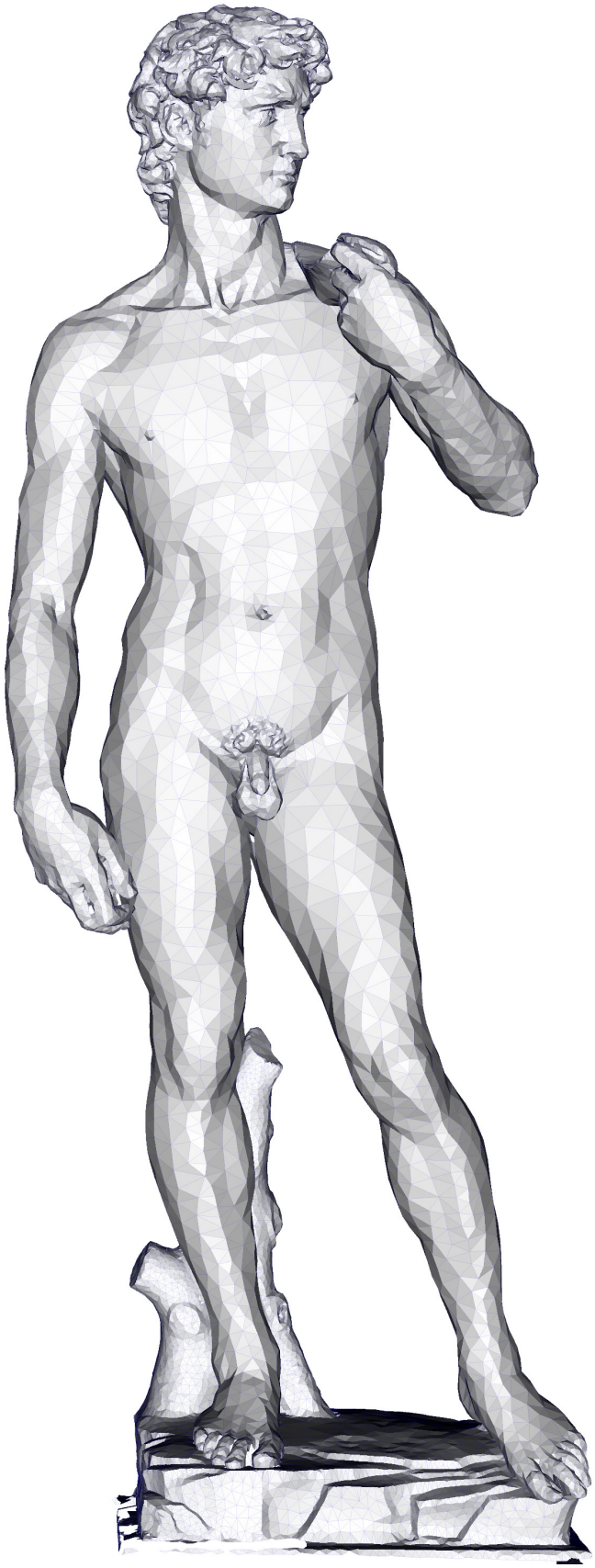

Fig. 7. Coarsened version of the David model (20k vertices, $\gamma = 1$)

Fig. 8. David model coarsened to 20k vertices. Left: original mesh. Center: uniform coarsening ($\gamma = 0$). Right: curvature adapted coarsening ($\gamma = 1.5$)

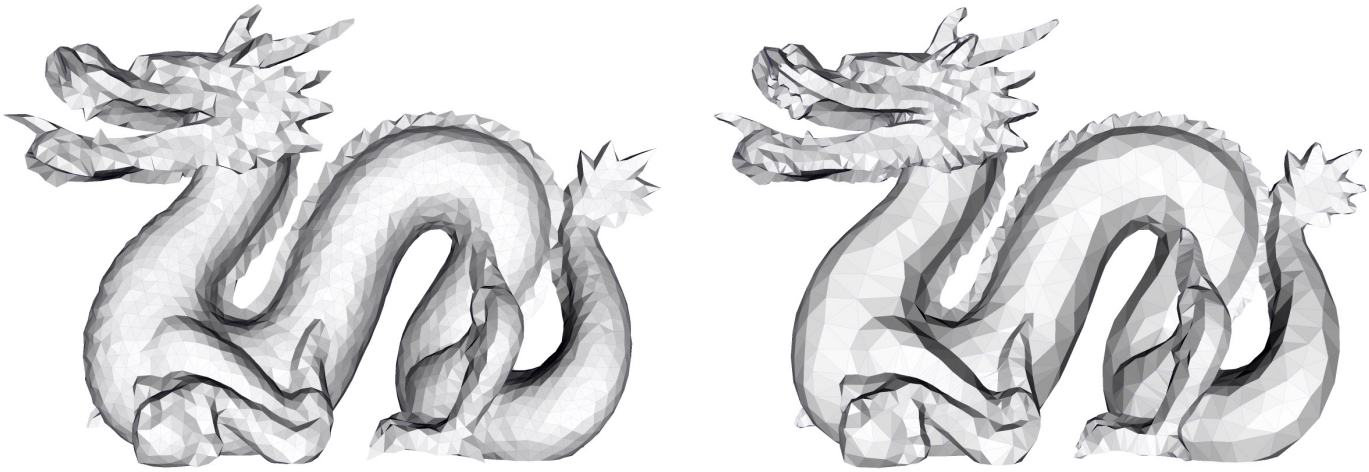

Fig. 9. Dragon model coarsened to 5k vertices. Left: uniform coarsening ($\gamma = 0$). Right: curvature adapted coarsening ($\gamma = 2$)

Fig. 10. Coarsened versions of the Happy Buddha model to 10k vertices ($\gamma = 0, 0.5, 1, 1.5$)