Technical Report

Mesh Coarsening based upon Multiplicatively Weighted Centroidal Voronoi Diagrams

Iurie Chiosa and Andreas Kolb

Version: 1.0 (24.04.2006)

Faculty 12 (Electr. Engineering & Comp. Science)
Institute for Vision and Graphics (IVG)
Computer Graphics and
Multimedia Systems Group
Prof. Dr. Andreas Kolb

Contents

Abstract

An approximated Centroidal Voronoi Diagram (CVD) construction provides an efficient and fast way for coarsening polygonal meshes. In this article a new generalization of the CVD for 3D meshes is presented - a Multiplicatively Weighted Centroidal Voronoi Diagram. Based on this concept we propose a mesh coarsening algorithm which is intended to capture the mesh features as good as possible. To achieve this, we propose three different types of cluster's weights. The mesh clusters are obtained by minimization of the underlying energy functional. To guarantee cluster connectivity, a specific vertex-boundary-count approach is introduced. The final coarse mesh is obtained by triangulating the optimized clustered model.

Along with that, we present a new and fast algorithm for initial seeds generation that can also be used for various types of applications which require an approximatively uniformly distributed initial set of seeds.

1. Introduction

In many situations a 3D model is represented by a polygonal mesh consisting of triangular faces. For a wide range of applications the input mesh is too complex, i.e. in terms of the number of vertices, therefore a simplified mesh is required. In general, reducing the mesh for a given number of output vertices requires maintaining its original fidelity. According to Heckbert and Garland [HG97], there are two major classes of simplification methods. *Refinement methods* start from a base mesh adaptively adding details to the mesh, e.g. Eck et al. [EDD*95]. On the other hand, *decimation approaches* start from the original mesh by removing mesh elements, i.e. vertices, edges or faces. One frequently used method is based on *edge collapses* [Hop96, GH97].

A third category, the so-called *remeshing techniques* can be identified. Here a mesh for a given number of output elements, e.g. vertices, or for a given error bound is computed. Explicit parametric remeshing approaches use global [GGH02] or local [SG03] parameterizations, whereas implicit or volumetric remeshing approaches construct an intermediate volume model [KJ01] [NT03].

Valette et al. [VC04] proposed a new algorithm for a fast mesh coarsening based on an approximated Centroidal Voronoi Diagrams (CVD). Due to its intrinsic properties, e.g. compactness of obtained tessellation, CVD provide an optimal strategy for resampling or, if seen from a clustering point of view, it provides an optimal k-clustering [DFG99]. That makes it most suitable for applications in which a final budget of elements, i.e. vertices, is fixed. A coarse mesh is obtained by first constructing a CVD on a surface of the input mesh and then triangulating the obtained Voronoi diagram resulting in its dual triangulation. This approach provides a triangulation with well-shaped triangles, e.g. suitable for applications such as finite elements analysis. Additionally, this approach is very fast since it is based on local quantity measures only.

However, in many cases the obtained coarse mesh fails to preserve or capture the main surface features. Valette et al. [VKC05] propose an extension of their prior technique, using a specific density function related to surface features such as mesh curvature to preserve these features.

An alternative approach to preserve surface features using a CVD technique, is presented in this paper. It is based on the expectation to have smaller sized Voronoi regions in higher curvature areas and larger sized ones in lower curvature areas, re-

sulting in more or less vertices, respectively. In order to achieve this, we introduce the *Multiplicatively Weighted Centroidal Voronoi Diagrams*.

Many clustering algorithms use a cluster growing technique which requires an initial set of seeds position. Ideally, the seed positions should be as close as possible to the final result of the clustering process, in order to have a faster convergence of the cluster optimization. In many situations a randomly chosen set of seeds is used, but in practice this is an inefficient approach. Alternatively, the seeds are selected by maximizing the distances between the seed points like in [JKS05] or distributing the seeds according to the density function [VKC05].

Contributions: We introduce a generalization of CVD, namely the Multiplicatively Weighted Centroidal Voronoi Diagrams. Based on this concept we propose a coarsening algorithm which is able to preserve the features of the input mesh based on a curvature-like measure. For that propose, we defined three different types of cluster weights. We also propose an approach for checking the cluster's connectivity and a new algorithm for initial seed generation. The later one is solely based on topological tests and can be used for many kind of applications that require a fast generation of an initial set of seeds that are approximatively uniformly positioned.

The rest of the paper is organized as follows: After discussing related work (Section 2) and the basic concept of *Multiplicatively Weighted Centroidal Voronoi Diagrams (MWCVD)* (Section 3), an overview of the algorithm is given (Section 4). Details on the related energy functional are represented in Section 5. The methods for the initial seed generation and for the connectivity check are discussed in Section 6 and Section 7, respectively. Details on the implementation are given in Section 8. Sections 9 and 10 present results of the proposed algorithm and draw some final conclusions.

2. Related Work

In [VC04] and [VKC05] algorithms for creating a coarse mesh using an approximated Centroidal Voronoi Diagrams (CVD) construction are proposed. In the next paragraphs we review these algorithms in more detail. Additionally, the basic definitions and properties of CVD and of Weighted Voronoi Diagrams in 2D are given.

Centroidal Voronoi Diagram (CVD): CVD is a Voronoi Diagram (VD) where each Voronoi site $\mathbf{z}_i$ is also the *mass centroid* $\mathbf{z}_i^*$ of its Voronoi-region V_i defined as:

$$\mathbf{z}_i^* = \frac{\int_{V_i} \mathbf{x}\rho(\mathbf{x})d\mathbf{x}}{\int_{V_i} \rho(\mathbf{x})d\mathbf{x}} \tag{1}$$

where $\rho(\mathbf{x})$ is a density function.

One of the most important property of CVD is that it minimize the following energy functional

$$E = \sum_{i=0}^{n-1} \int_{V_i} \rho(\mathbf{x}) \left|\mathbf{x} - \mathbf{z}_i\right|^2 d\mathbf{x}. \tag{2}$$

Common algorithms for constructing a CVD are the Lloyd's method and k-means clustering. For a more comprehensive discussion see [DFG99].

The algorithm of Valette et al. [VC04]**:** Given a polygonal mesh M with cells C_j, one assumes that the boundaries of a Voronoi-region V_i is a subset of the edges of M, i.e. the V_i consist of a union of several mesh cells C_j. The energy functional equivalent to Eq. (2) can be defined as:

$$E = \sum_{i=0}^{n-1} (\sum_{C_j \in V_i} \int_{C_j} \rho(\mathbf{x}) \, |\mathbf{x} - \mathbf{z}_i|^2 \, d\mathbf{x}). \tag{3}$$

It can be shown that if approximating each cell C_j by a single point, i.e. by its centroid $\gamma_j = \int_{C_j} \rho(\mathbf{x}) \mathbf{x} d\mathbf{x} / \int_{C_j} \rho(\mathbf{x}) d\mathbf{x}$, the energy functional defined by Eq. (3) can be simplified to

$$F = \sum_{i=0}^{n-1} (\sum_{C_j \in V_i} \rho_j \, |\gamma_j - \bar{\gamma}_i|^2). \tag{4}$$

where $\rho_j = \int_{C_j} \rho(\mathbf{x}) d\mathbf{x}$ is the weight of a given cell C_j and $\bar{\gamma}_i = \sum_{C_j \in V_i} \rho_j \gamma_j / \sum_{C_j \in V_i} \rho_j$ is the centroid of the cluster V_i.

Now, constructing an approximated CVD for a polygonal mesh M can be seen as a clustering problem, in which the original mesh cells are assigned to different clusters V_i, i.e. to approximated Voronoi-regions. This is realized by minimization of the energy functional (4).

The minimization process works as follows: First, each cluster is assigned a randomly picked cell. Next, for each boundary edge of two adjacent clusters V_k and V_l the energy F is computed for the initial configuration, for the case when cluster V_k grows, i.e. both cells belong to V_k, and when cluster V_l grows, i.e. both cells belong to V_l (see Fig. 1 (a)). The case resulting in the smallest energy is chosen and the cluster configurations are updated. Thus, the energy functional is iteratively reduced. The final clustering is obtained when no further energy reduction occurs.

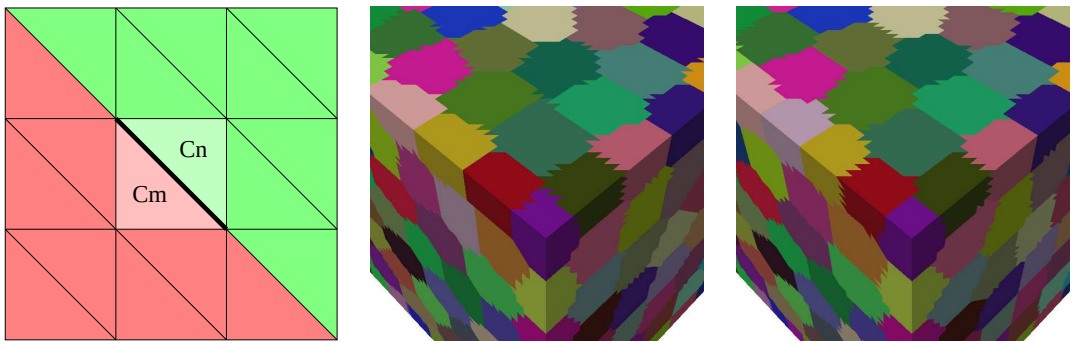
Valette et al. [VC04] also showed that comparing the energy for all three cases requires only the computation of

$$L_2 = - \frac{\left| \sum_{C_j \in V_k} \rho_j \gamma_j \right|^2}{\sum_{C_j \in V_k} \rho_j} - \frac{\left| \sum_{C_j \in V_l} \rho_j \gamma_j \right|^2}{\sum_{C_j \in V_l} \rho_j}$$

That makes the algorithm very fast because there is no need to explicitly compute the centroid position $\bar{\gamma}_i$ (see Eq. (4)). The resulting test is based on the cluster values $\sum_{C_j \in V_i} \rho_j \gamma_j$ and $\sum_{C_j \in V_i} \rho_j$, which have to be updated in case of a cluster growing or shrinking.

For a uniform density function $\rho(\mathbf{x})$ the algorithm yields a uniform output triangulation. Here the weight ρ_i (see Eq. (4) is proportional to the area of the cell C_j. The vertices in the final coarse mesh were set by identifying a vertex in the original input mesh which is the closest to the corresponding cluster centroid $\bar{\gamma}_i$.

Adaptive Mesh Coarsening: Recently, Valette et al. [VKC05] proposed an extension to their previous work heading for an adaptive polygonal mesh coarsening. To simplify the mesh according to the local mesh features, they use a curvature indicator as density function, i.e. each mesh cell C_j is given a weight ρ_i (see Eq. (4)) according to the curvature measures. To guarantee a valid clustering a three step algorithm is performed:



(a) (b) (c)

Figure 1: *(a) Local tests performed for a given boundary edge (solid line) of two adjacent clusters V_k (red) and V_l (green). (b)&(c) Result of an approximated CVD using nonuniform density function for $\rho^{new} = 50$ and $\rho^{new} = 200$, respectively.*

First, an approximated CVD using the above described algorithm is constructed. Here the density of the initial cluster seeds is related to the density function, i.e. in areas with high curvature more seeds are placed. Second, a cleaning is applied, i.e. disconnected clusters are identified. After the cleaning the minimization step is applied again together with an additional check to preserve the connectivity of all clusters. The quality of the final coarse mesh was enhanced using a quadric-based vertex placement.

Weighted Voronoi Diagrams: Weighted Voronoi diagrams are well known in 2D (see for example [OBS92]). For a given set of n different sites $\{\mathbf{z}_i\}_{i=0}^{n-1}$ in the domain Ω, the weighted Voronoi-region can be defined as:

$$V_i^w = \{\mathbf{x} \in \Omega |\ d_i(\mathbf{x}, \mathbf{z}_i) < d_j(\mathbf{x}, \mathbf{z}_j)\ \forall\ j \neq i\}$$

where d_i is a weighted distance measure for cluster i. For this type of VD the assumption is that the site has a predetermined weight which reflects an application-specific property [OBS92].

For an ordinary VD one assumes that all clusters have the same weight, i.e. $d_i(\mathbf{x}, \mathbf{z}_i) \equiv d(\mathbf{x}, \mathbf{z}_i)$, where d is the standard Euclidean distance.

Multiplicatively Weighted Voronoi Diagram: For this type of weighted Voronoi Diagrams (see [AE84], [OBS92]) d_i is given by

$$d_i(\mathbf{x}, \mathbf{z}_i) = w_i\,|\mathbf{x} - \mathbf{z}_i|$$

where $\{w_i\}_{i=0}^{n-1}$ are predetermined positive weights. In this case, we refer to V_i^w as *multiplicatively weighted (MW) Voronoi-region* V_i^{mw} and to the set $\{V_i^{mw}\}_{i=0}^{n-1}$ as *multiplicatively weighted (MW) Voronoi diagram*.

Generally, MW-Voronoi regions are not necessarily connected or convex. In some situation they may also contain holes, depending on the associated weights.

3. Multiplicatively Weighted Centroidal Voronoi Diagram (MWCVD)

The concept of CVD can be also extended to the weighted Voronoi diagrams. In this section we introduce the Multiplicatively Weighted CVD (MWCVD) as an extension of

a CVD. This concept is the basis of our algorithm for mesh coarsening.

Given a set of n different seeds $\{\mathbf{z}_i\}_{i=0}^{n-1}$ together with predetermined positive weights $\{w_i\}_{i=0}^{n-1}$ and a positive density function $\rho(\mathbf{x})$ in the 2D domain Ω. A *Multiplicatively Weighted Centroidal Voronoi Diagram (MWCVD)* is a MW-Voronoi diagram for which the sites $\{\mathbf{z}_i\}_{i=0}^{n-1}$ of the regions $\{V_i^{mw}\}_{i=0}^{n-1}$ are their centroids (see Eq. (1) for the definition of the centroid).

Similar to the CVD the following property holds:

PROPOSITION 3.1 *Given a set of n different sites $\{\mathbf{z}_i\}_{i=0}^{n-1}$ with associated positive weights $\{w_i\}_{i=0}^{n-1}$ and a density function $\rho(\mathbf{x})$ in the domain Ω. Let $\{V_i^{mw}\}_{i=0}^{n-1}$ denote any tessellation of Ω into n regions. Define:*

$$E^{mw} = \sum_{i=0}^{n-1} \int_{V_i^{mw}} \rho(\mathbf{x})\, w_i\, |\mathbf{x} - \mathbf{z}_i|^2\, d\mathbf{x} \tag{5}$$

E^{mw} is minimized if and only if $\{V_i^{mw}\}_{i=0}^{n-1}$ is a MWCVD.

Proof of this Proposition is given in Appendix A.

Observe that in the case of uniform weights $\{w_i\}_{i=0}^{n-1}$ the MWCVD becomes an ordinary CVD, so that the MWCVD is a generalization of CVD.

4. Overview of the Algorithm

Our coarsening algorithm uses the same strategy as described in [VC04]: A coarse mesh is constructed by first constructing an approximated MWCVD. The triangulated model is obtained according to the clusters adjacency relations based on the centroids of the clusters.

In a more general context, this kind of clustering can be viewed as a weighted mesh clustering, where the cluster size is related to the density ρ, e.g. the curvature. This results in a triangulation with smaller or larger triangles in regions with higher or lower density, i.e. curvature, respectively.

The steps of our algorithm are as follows:

Initialization: Using our k-neighborhood initialization (Section 6) the initial number of clusters along with the initial seeds and the k-neighborhoods are determined.

Initial Clusters Grow: Starting with the k-neighborhoods, we grow each cluster at a time, looping over the boundaries of a cluster, until the entire model is covered. This step does not require any energy computation. The initial cluster weights, according to Section 5.3, are computed using this initial configuration .

Energy Minimization: At each iteration, the algorithm loops over the boundaries of each cluster. For a given boundary edge, a check is performed if one of the clusters will be disconnected by growing or shrinking (see Section 7). If growing is allowed the configuration which is energetically minimal will be chosen and the cluster configuration will be updated accordingly. This process continues until there is no change in the energy between two consecutive iterations.

Triangulation: The final triangulation is created by inserting one vertex per cluster, i.e. the cluster centroid, and connecting them according to the cluster's adjacency.

5. Energy Functional

Clustering a mesh by minimizing the *Multiplicatively Weighted CVD (MWCVD)* energy functional (5), one needs to define $\rho(\mathbf{x})$ and w_i accordingly. In this section, we present a simplified version of the energy functional (Section 5.1) which is used to construct an approximated MWCVD. Along with a specified density function (Section 5.2) we propose three different types of cluster's weight (Section 5.3) that can be used to obtain a desired clustering.

5.1. Approximation of the MWCVD Energy Functional

In the case of a continuous space the energy functional for a MWCVD is defined by Eq. (5). The goal is to achieve a local energy comparison as in the case of the approximated CVD (see Section 2).

Assuming $\rho(\mathbf{x})$ to be uniformly defined over the cell C_j, i.e. $\rho(\mathbf{x}) = \rho_j, \forall \mathbf{x} \in C_j$. In this case the energy functional (5) becomes

$$E^{mw} = \sum_{i=0}^{n-1} \Big(\sum_{C_j \in V_i} \rho_j A_j w_i \left| \gamma_j - \bar{\gamma}_i \right|^2 \Big) \tag{6}$$

with the site of each cluster V_i chosen as its centroid

$$\bar{\gamma}_i = \frac{\sum_{C_j \in V_i} \rho_j A_j \gamma_j}{\sum_{C_j \in V_i} \rho_j A_j}. \tag{7}$$

Here A_j and ρ_j are the area and the density function of the cell C_j, respectively. Eq. (6) together with Eq. (7) can be simplified to:

$$E^{mw} = \sum_{i=0}^{n-1} w_i \Big(\sum_{C_j \in V_i} \rho_j A_j \left| \gamma_j \right|^2 - \frac{\left| \sum_{C_j \in V_i} \rho_j A_j \gamma_j \right|^2}{\sum_{C_j \in V_i} \rho_j A_j} \Big) \tag{8}$$

Thus the computation of the energy for a given cluster is base only on three values for each cluster V_i, $\sum_{C_j \in V_i} \rho_j A_j \left| \gamma_j \right|^2$, $\sum_{C_j \in V_i} \rho_j A_j \gamma_j$ and $\sum_{C_j \in V_i} \rho_j A_j$. The update of these values can be done using local tests only.

5.2. Density Function

Because of the assumption that $\rho(\mathbf{x})$ is uniformly defined over the cell C_i, i.e. equal to ρ_i, the density function is no longer a continuous function but a discrete one defined for each cell C_i. As we try to capture the mesh feature, ρ_i needs to have larger values in higher curvature regions than in lower ones. In this case one possibility to define ρ_i is to compute the curvature on the mesh and relate it with ρ_i, as done in [VKC05].

We use a slightly more simple approach and define ρ_i based on the observation that generally normal field discontinuities directly indicate a mesh features, e.g. a cell which is on a high curvature feature will have large deviation of its normal w.r.t. its neighbors. Thus, we set ρ_i equal to the mean of normals difference of the cells in the 1-ring of cell

C_i. For the cell C_i with vertices $\mathbf{V}_{i_1}, \ldots, \mathbf{V}_{i_m}$ we have

$$\rho_i = \frac{1}{n_i} \sum_{j=1}^{m} \sum_{C_k \text{ adj. to } V_j} \|\mathbf{n}_i - \mathbf{n}_k\|$$

where $\mathbf{n}_k$ is the normal of cell C_k and $n_i = \sum_{j=1}^{m} \sum_{C_k \text{ adj. to } V_j} 1$.

Note that, with this definition of ρ_i, one gets a value of zero for planar regions. A mapping of obtained values to a new interval is applied in order to prevent division by zero in Eq. (8). For our application we used a linear mapping from $[0; \rho^{max}]$ to $[1; \rho^{new}]$.

In general, ρ^{new} is a user defined parameter. For large values of ρ^{new} the cluster centroids move towards cells with higher density. This is a positive effect in the context of feature preservation.

Remark:

Assume that the multiplicatively weights w_i are uniformly defined for all clusters i, i.e. one has an approximated CVD with nonuniform density function (see Eq. (6) and Eq. (4)). Constructing an approximated CVD using this setup will result in a clustering as depicted in Figure 1(b)-(c).

Observe that even if one uses a larger value for ρ^{new}, one still gets approximatively the same result, i.e. the cluster's size is not affected considerably. The explanation can be seen from Eq. (4). For a boundary face f between two adjacent clusters V_i and V_j the energy contribution will be $\rho_f |\gamma_f - \bar{\gamma}_i|^2$ and $\rho_f |\gamma_f - \bar{\gamma}_j|^2$, respectively. Obviously, the energy contribution of f is mainly driven by the distance to the cluster centroids $\bar{\gamma}_i$ and $\bar{\gamma}_j$, thus the size of a cluster will not be affected by the density ρ_f of cell f. Therefore, we introduce methods to determine appropriate multiplicative weights for the clusters, that take the cell densities into account.

5.3. Clusters Weights

To have a complete energy formulation, the cluster weights w_i, which define the multiplicative weight in the distance measurement, have to be specified. To define the clusters weights, we directly relate w_i to the cell densities ρ_j in the cluster. As the cluster configuration changes during different iterations, the weight will be updated accordingly.

We propose three different types of cluster's weight:

Weighted Area:

$$w_i^A = \sum_{C_j \in V_i} \rho_j A_j \tag{9}$$

i.e. the area of the cluster weighted with the cell densities. This means, clusters with higher density get smaller in size compared to clusters with low density.

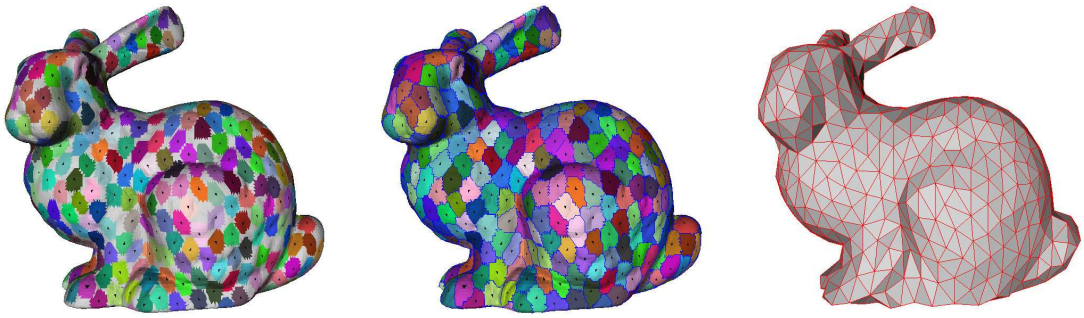
Maximum Density:

$$w_i^M = max_{C_j \in V_i} \{\rho_j\} \tag{10}$$

Assigning the maximum density to the cluster weight is based on the idea, that cells with low or high density should be contained in clusters with low or high weight, respectively.

Density Variance:

$$w_i^V = 1 + \sum_{C_j \in V_i} (\rho_j - \bar{\rho}_i)^2 \tag{11}$$



(a) (b) (c)

Figure 2: *Result of the k-neighborhood initialization for the Bunny model with level $k = 7$. (a) Obtained seeds and their k-neighborhood, time: 1 sec.; (b) Result of the initial cluster growing, time: 2sec.; (c) Triangulated result using the cluster centroid as vertex, time: 1sec.*

Observe that in the case of a uniform density function our MWCVD will result in an ordinary CVD for the maximum density based and density variance cluster's weight. This is not the case for w^A and could be solved by normalizing w^A with the cluster area $\sum_{C_j \in V_i} A_j$. This, however, yields insufficient clustering results.

6. k-Neighborhood Initialization

In this section we present a simple algorithm for generating an initial set of seeds that is approximatively equal distributed over the input mesh. It does not use any distance computation, it merely use the cell adjacencies across edges (*edge-neighborhood*).

Given a polygonal mesh M with cells C_j. Assume that all valid cells, i.e. mesh faces, are contained in an array (C_j^v). Holes are considered to be invalid. We also assume that each cell has references to its neighbors, which is a standard data element in mesh data structures such as the half-edge data structure.

For a given level k, the algorithm works as follows:

```
∀ cell in (C_j^v): setValid(cell)
∀ cell ∈ (C_j^v) do:
  if ( isValid(cell) && isValid(k-neighborhood of cell))
    setInvalid(k-neighborhood of cell)
    addToSeedSet(cell)
```

For example, for level $k = 2$ a given cell is reported as a seed if its neighboring cells are `valid` and, at the same time, each neighboring cell also has its neighboring cells `valid`. The result of this algorithm is presented in Fig. 2 (a).

The algorithm selects a set of seed cells, whose k-neighborhoods do not overlap. In a brute-force implementation, for a triangulated mesh this algorithm has a runtime complexity of 3^k to identify a single seed or k-neighborhood (see Fig. 3). The number of seed is of order $m/(k^2)$, where m is the total number of cells, yielding an overall complexity of $(m \cdot 3^k)/(k^2)$. Therefore, the k-neighborhood initialization should be used only for small values of k. We use $k = 7$, resulting in neighborhoods of some 80 cells.

Observe in Fig. 2 (a) that there are some cells that are not assigned to any given seed's k-neighborhood. This is due to the fact, that the algorithm simply works on the array of

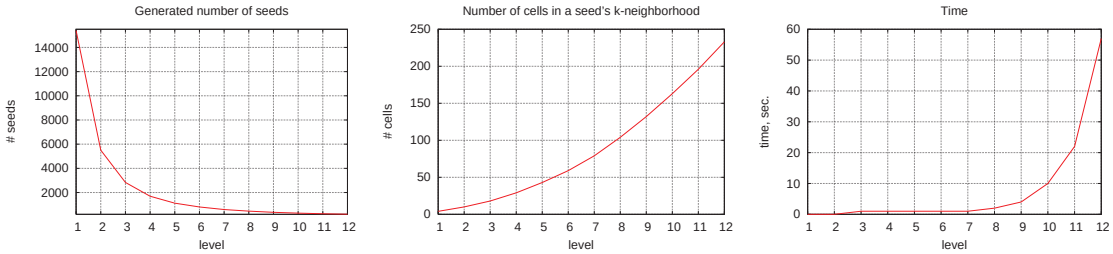


Figure 3: *Result of k-neighborhood initialization for the Bunny model.*

cells trying to identify the next cell with valid, i.e. not yet covered, k-neighborhood.

In case of uniformly sized triangles in the input mesh, the resulting k-neighborhoods are nearly disk-shaped. This yields very good starting seed sets for algorithms that try to construct equally sized and compact clusters, as the original CVD algorithm [VC04]. Additionally, in some situations the result of the k-neighborhood initialization is already good enough for a final mesh reduction (see result in Fig. 2 (b-c)). This is especially true, if the user wants to evaluate the final mesh resolution interactively, before deciding on the level to be used in the overall optimization.

Some applications may require a fixed number of seeds. In this case one can use a modified version of the algorithm to reduce the number of obtained seeds. First perform an initial growing based on the given k-neighborhoods (see Section 8.1). After that, check the area of the resulting clusters and reduce the number of seeds by merging the three neighboring clusters having a common vertex with the smallest summed area. The seed's new position is assigned to the common vertex of the three original clusters. The same idea can be used to avoid inefficient initializations for large k.

7. Connectivity Check

In the final clustering, each cluster has to be a 1-connected set of cells in order to obtain a valid triangulation. In this section we present an algorithm that prevents any violation of this constraint during the clustering process.

In the following, consider a boundary edge c between two adjacent clusters (see Figure 4 (a)). We check, if the red cluster can grow in the direction of the edge c, i.e. whether the triangle $\Delta(\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3)$ may switch to the red cluster. Let n and p denote the next and previous edges w.r.t. c for the considered triangle, respectively.

Our algorithm is related to the recently published work of Valette et al. [VKC05]. They relate their approach solely on the check of edges n and p and on vertex $\mathbf{V}_2$. In their implementation growing is allowed if: n or p are exclusively boundary edges. Otherwise, if n and p are not boundary edges, all triangles in the 0-ring of the vertex $\mathbf{V}_2$ should be of the same cluster. This approach has the disadvantage, that the check fails to handle the disconnectivity as pointed out in Fig. 4 (c).

Our algorithm is based on an additional counter for each vertex storing the number of adjacent cluster boundary edges, referred to as *boundary-edge count*. After the initial growing, the counter is set to the according number of cluster boundary edges. Thus, for example, in Fig. 4, case 0, vertex $\mathbf{V}_1$ will have count 2, $\mathbf{V}_3$ will have count 3, whereas $\mathbf{V}_2$ still has a count equal to zero.

Clearly, growing does not lead to a disconnected cluster, if the boundary-edge count for

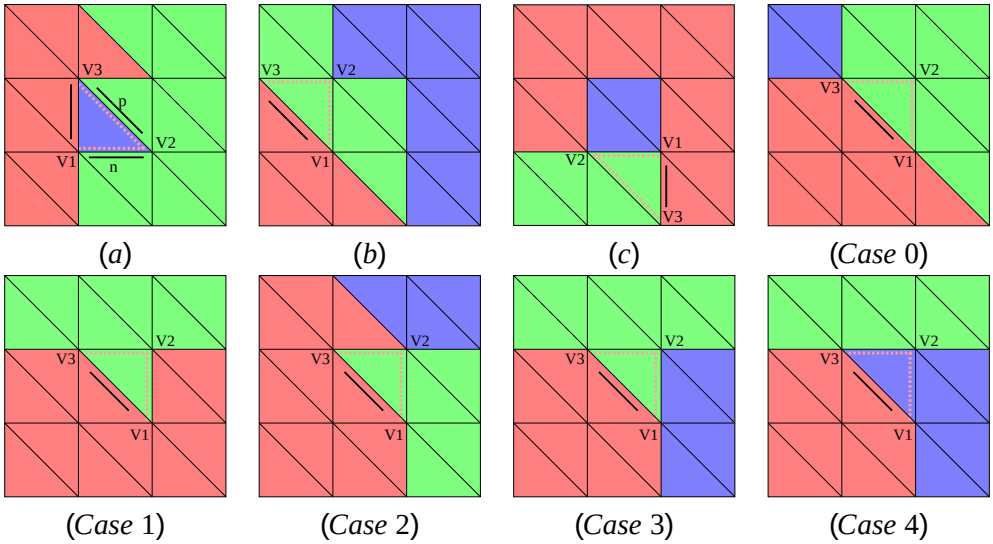


Figure 4: *All configurations. Boundary edge c is one which points from vertex V_1 to V_3; n and p denote the next and previous edges w.r.t c.*

$\mathbf{V}_2$ is zero. Also, as pointed out by Valette et al. [VKC05], if only one of the edges n or p is a boundary to the *current cluster*, i.e. w.r.t. edge c, growing does to lead to an invalid situation. Otherwise, if n XOR t is a boundary to the *different cluster*, we check, if vertex $\mathbf{V}_2$ has no adjacent boundary to the current cluster. Only in this situation, growing is permitted.

Putting this together, we have the following connectivity check:

```
( count(V2) = 0 )          OR // Case 0
( isBoundaryOfCurr(n) XOR isBoundaryOfCurr(p) )   OR // Case 1,2
( (isBoundary(n) XOR isBoundary(p)) AND
   !V2.hasBoundaryEdge(curr) )  // Case 3,4
```

where `curr` denotes the current cluster, thus `V2.hasBoundaryEdge(curr)` checks for boundary edges for the current cluster at $\mathbf{V}_2$.

Using this check the following situations are avoided (see Fig. 4 (a), (b) and (c)):

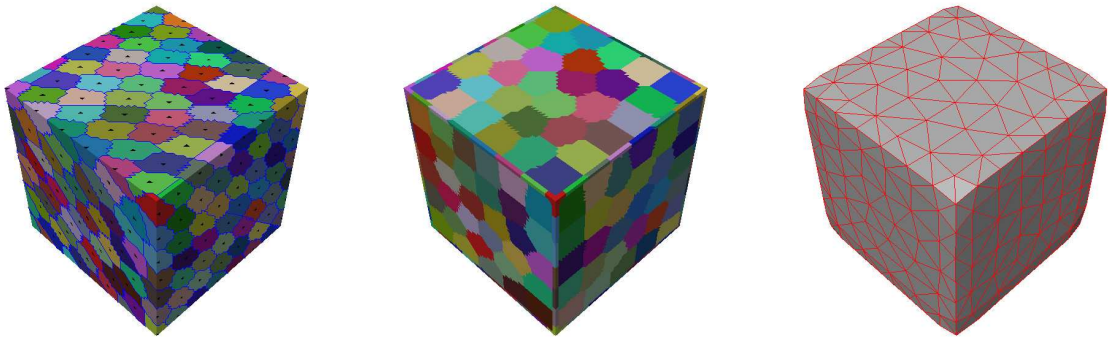
- The adjacent cell represent a single cluster . Here, a growing would lead to an implicit deletion of one cluster.
- Splitting the cluster into two parts. This is the only situation that can possibly cause a splitting of green cluster.
- Growing may result in nonzero genus clusters.

The update of the boundary-edge counter for the vertices after a growing is quite simple. With respect to the cases noted in the above algorithm, we have

`Case 0:` Update: `count(v2) = 2`.

`Case 1,` n is a boundary of the current cluster.
Update: `count(V1) = 0`.

`Case 2,` p is a boundary of the current cluster.
Update: `count(V3) = 0`.



(a) (b) (c)

Figure 5: *Algorithm stages: (a) Result of the initial cluster growing. (b) Final clustering obtained after the energy minimization. (c) Triangulation of the final clustering (b).*

`Case 3,` n is a boundary of a different cluster.
Update: `count(V1) --; count(V2) ++.`

`Case 4,` p is a boundary of a different cluster.
Update: `count(V3) --; count(V2) ++.`

Note, that case 0 is the most encountered situation during the cluster optimization, which leads to a very efficient check in this situation. Cases 3 and 4 are the most expensive checks, but they do not occur very frequently.

8. Implementations Details

In this section we will describe implementation details on the algorithm outlined is Section 4.

8.1. Initial cluster grow

After the determination of the k-neighborhoods for a given value of k, the obtained seeds, together with their neighborhoods, are considered as initial clusters. Then, the algorithm starts to loop over the boundaries of the cluster, one at a time, and assigns the free cells to the given cluster. That process is repeated until there are no more free cells (see Figure 5 (a)).

Using the obtained clusters, we specify the weight w_i according to the specified weight (see Section 5.3). These weights will be used in the first iteration of the minimization process.

8.2. Energy minimization

The energy minimization process can be described as follows: For a given boundary edge, we first check if the cluster can grow or shrink w.r.t. this edge without violating the connectivity constraint, using the algorithm described in Section 7. The energy computation is done only for configurations where growing or shrinking is permitted. Finally, the case resulting in smallest energy is chosen and the cluster's configuration is updated accordingly. In order to prevent a recalculation of the cluster's energy, we store the energy of the cluster configuration.

The energy E^{mw} is minimized at each iteration and the minimization process stops when there is no more change to the configurations of the clusters between two consecutive iterations. The result of this minimization process is presented in Fig. 5 (b).

8.3. Triangulation

A final triangulated coarse mesh is obtained by regarding each cluster V_i as a Voronoi-region. Therefore the dual of the MWCVD is the resulting triangulation (see Fig. 5(c)). The vertices of the final coarsened mesh are chosen to be equal to the centroid of the cluster, which were already computed during clustering process. That makes the final triangulation a simple and fast operation.

8.4. Efficient Cluster Weight Computation

A proper reformulation of the cluster weight (see Eq. (9),(10),(11)) results in local computation rules which can be efficiently evaluated.

For the weighted-area based approach w_i^A (see Eq. (9)) the value $\sum_{C_j \in V_i} \rho_j A_j$ is already computed for the energy functional (8). Therefore, this cluster weight is obtained with no additional computational cost.

The density-variance approach (see Eq. (11)) can be simplified to

$$w_i^V = \sum_{C_j \in V_i} \rho_j^2 - (\sum_{C_j \in V_i} \rho_j)^2 / n_i \; + 1 \tag{12}$$

where n_i is the number of cells in a cluster. This allows to compute w_i^V also using only local computations based on the additional quantities $\sum_{C_j \in V_i} \rho_j^2$ and $\sum_{C_j \in V_i} \rho_j$ per cluster.

For the maximum-density approach (see Eq. (10)), the current maximum needs to be identified by checking all cells in a cluster. To overcome this problem we propose to use the following approximation:

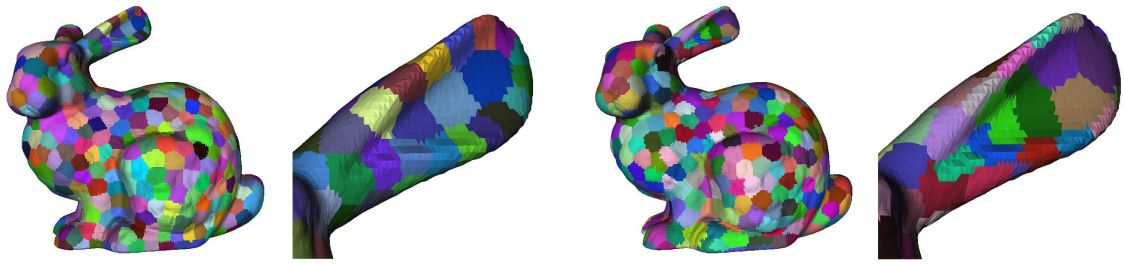
$$max_{C_j \in V_i}\{\rho_j\} = \lim_{p \to \infty} (\sum_{C_j \in V_i} \rho_j^p)^{1/p} \tag{13}$$

This approximative value of the maximum ρ in a cluster can be computed using the additional value $\sum_{C_j \in V_i} \rho_j^p$ which again requires only local updates. To get a good approximation of the maximum ρ in a given cluster, the value of p must be large enough. In our implementation we use a value of 10, since larger values may lead to numerical problems.

9. Results

A comparison between a CVD clustering with varying density and the MWCVD technique utilizing the density variance cluster weight for the Bunny model exhibits the desired result in sense of the feature elongated clusters (see Fig. 6).

Fig. 7 (top) shows different clustering results obtained for the Fandisk model, using proposed cluster's weights (see Section 5.3 and 8.4). Observe that, in all three cases the obtained clustering provides smaller-sized clusters in higher curvature regions where larger-sized one are covering lower curvature areas. Using the maximum-density or



(a) (b)

Figure 6: *Comparison of CVD clustering (a) with density in the range of* $[1, 200]$ *and the density variance approach (b).*

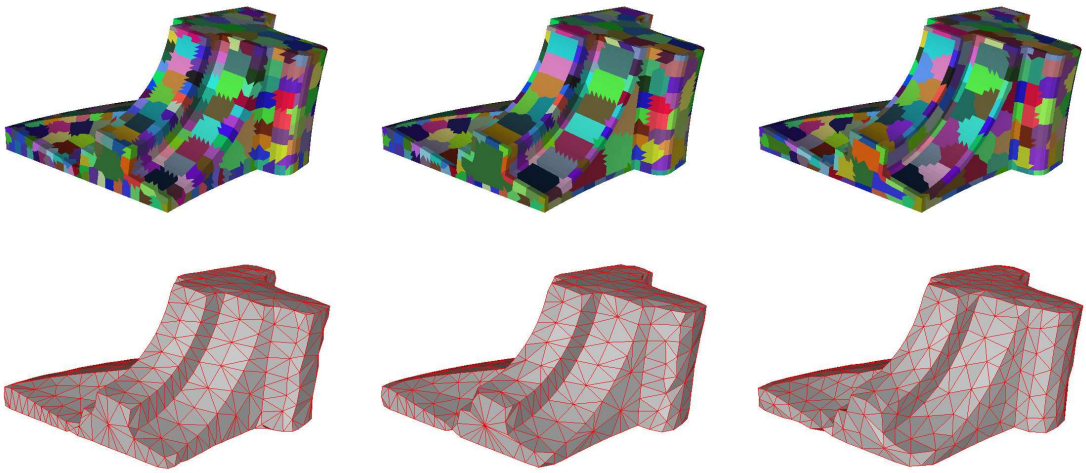


Figure 7: *(Top) Clustering results and (bottom) corresponding triangulation for the Fandisk model. Left: result using* w^A*, Center: result using* w^M*, Right: result using* w^V

density-variance approach, i.e. w^M or w^V respectively, result in clusters which are elongated along the feature lines. For the weighted-area based, i.e. w^A, the clusters are more compact and also containing a significant number of cells with low curvature. From this perspective, the clustering results obtained using former two cluster's weight, i.e. w^M or w^V, are more suitable for our final triangulation, because the cluster centroid is closer to the original surface.

Fig. 7 (bottom) shows the corresponding triangulations. Although we use only the cluster centroid as a vertex for the triangulation, the final coarse mesh still preserves the main features of the original mesh. Note that using one of the vertex positioning techniques proposed in [VC04] (closed mesh vertex) or [VKC05] (quadric-based placement), the final mesh quality around feature line could be significantly improved. Small defects which can be seen in the obtained triangulation are due to the fact that some clusters contain more than one feature point, i.e. the intersection of several feature lines. In the final triangulation these feature points are reduced to a single vertex.

Table 9 provides the results for different meshes. The first row gives the number of the vertices of the input mesh. The second one shows a chosen value for level k. For a given value of k, the k-neighborhood initialization gives the total number of seeds, i.e. the number of vertices in the final coarse mesh. The fourth row shows the time used

Model		Fandisk	Cube	Bunny	Horse	Dragon
# V(input mesh)		6475	15002	34834	48485	437645
k (level)		3	7	7	5	7
# seeds (obtained)		527	274	594	1579	6829
Init (sec.)		<1	<1	1	1	13
Clustering, w^A	ρ^{new}	50	50	50	5	50
	(sec.)	8	20	59	112	940
Clustering, w^M	ρ^{new}	50	50	50	5	5
	(sec.)	8	12	63	124	864
Clustering, w^V	ρ^{new}	3	3	3	3	3
	(sec.)	8	5	50	106	759

Table 1: *Results for different input meshes.*

for performing k-neighborhood initialization. For different cluster's weights, the scaling parameter ρ^{new} and the time required to obtain the final clustering are presented. All the results were obtained on a 3GHz Intel PC.

In the case of the density-variance approach the value for the scaling parameter ρ^{new} is usually chosen to be small compared to the value used for the weighted-area and maximum-density approach (see Tab. 9). This is due to the quadratic influence of the density function ρ on the cluster weight w^V (see Eq. 11).

At the same time, we have to point out that the timing presented in Tab. 9 should be seen as relative one due to the fact that our implementation does not use a highly optimized mesh data structure.

10. Conclusions and Future Work

We presented a mesh coarsening algorithm based on an extension of Centroidal Voronoi Diagram (CVD), namely *Multiplicative Weighted Centroidal Voronoi Diagram (CVD)*. The cluster weights are directly related to the density function of the mesh, which itself resembles the mesh features, i.e. the curvature. We proposed three different ways to define the cluster weights resulting in differently shaped clusters in the final MWCVD. The algorithm uses only local computations w.r.t. cluster boundary edges yielding a fast implementation. The MWCVD provides a natural way for feature-preserving coarsening.

Additionally, a new algorithm for defining an initial set of seeds for the clustering based on edge-adjacencies has been introduced. This algorithm yields well distributed seeds for the optimization process. Moreover, a concise connectivity check was presented, guaranteeing that the clusters are always edge-connected.

In future we plan to introduce approaches for a better positioning of the vertices in the final triangulation.

11. Appendix

A: Proof of the Proposition 3.1

Given a set of n different seeds $\{\mathbf{z}_i\}_{i=0}^{n-1}$ with associated positive weights $\{w_i\}_{i=0}^{n-1}$, a positive density function $\rho(\mathbf{x})$ in the 2D domain Ω and a tessellation of Ω into n regions $\{V_i^{mw}\}_{i=0}^{n-1}$. E^{mw}, according to Eq. (7), is minimized if and only if the regions V_i^{mw} are the MW-Voronoi regions associated to the generators $\mathbf{z}_i$ and, simultaneously, the generators $\mathbf{z}_i$ are the centroids $\mathbf{z}_i^*$ of the regions V_i^{mw} (see Eq. (6)). We follow a similar argument as in [DFG99].

First, assuming that E^{mw} is minimized, we have to show that the generators $\{\mathbf{z}_i\}_{i=0}^{n-1}$ are the centroids corresponding to the regions $\{V_i^{mw}\}_{i=0}^{n-1}$.

Examine the variation of E^{mw} with respect to $\mathbf{z}_i$, namely $E^{mw}(\mathbf{z}_i+\varepsilon\mathbf{v}) - E^{mw}(\mathbf{z}_i)$, where $\mathbf{z}_i+\varepsilon\mathbf{v} \in \Omega$. Now, dividing by ε and taking the limit as $\varepsilon \to 0$, one yields:

$$\mathbf{z}_i = \frac{\int_{V_i^{mw}} \mathbf{x}\rho(\mathbf{x})d\mathbf{x}}{\int_{V_i^{mw}} \rho(\mathbf{x})d\mathbf{x}}$$

Second, we show that E^{mw} is minimized if $\{V_i^{mw}\}_{i=0}^{n-1}$ are MW-Voronoi regions associated with sites $\{\mathbf{z}_i\}_{i=0}^{n-1}$.

Recall the fact that due to the construction of the MW-Voronoi diagram for $\mathbf{x} \in V_i^{mw}$ we get for $i \neq j$:

$$\begin{aligned} & w_i \left|\mathbf{x}-\mathbf{z}_i\right| < w_j \left|\mathbf{x}-\mathbf{z}_j\right| \\ \iff \quad & \rho(\mathbf{x}) w_i \left|\mathbf{x}-\mathbf{z}_i\right|^2 < \rho(\mathbf{x}) w_j \left|\mathbf{x}-\mathbf{z}_j\right|^2 \end{aligned}$$

Thus, for a *fixed* set of sites $\{\mathbf{z}_i\}_{i=0}^{n-1}$ the energy E^{mv} defined according to Eq. (7) will be smaller compared to the case when the tessellation is not a MW-Voronoi diagram.

References

AE84. AURENHAMMER F., EDELSBRUNNER H.: An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recognition 17*, 2 (1984), 251–257.

DFG99. DU Q., FABER V., GUNZBURGER M.: Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM Review 41*, 4 (1999), 637–676.

EDD*95. ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *Proc. SIGGRAPH* (1995), pp. 173–182.

GGH02. GU X., GORTLER S., HOPPE H.: Geometry images. In *Proc. SIGGRAPH* (2002), pp. 355–261.

GH97. GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proc. SIGGRAPH* (1997), pp. 209–216.

HG97. HECKBERT P. S., GARLAND M.: Survey of polygonal surface simplification algorithms. *Proc. SIGGRAPH* (1997). Multiresolution Surface Modeling Course.

Hop96. HOPPE H.: Progressive meshes. In *Proc. SIGGRAPH* (1996), pp. 99–108.

JKS05. JULIUS D., KRAEVOY V., SHEFFER A.: D-charts: Quasi-Developable Mesh Segmentation. *EUROGRAPHICS 24*, 3 (2005), 581–590.

KJ01. KOLB A., JOHN L.: Volumetric Model Repair for Virtual Reality Applications. In *EUROGRAPHICS Short Presentation* (2001), University of Manchester, pp. 249–256. ISSN 1017-4656.

NT03. NOORUDDIN F. S., TURK G.: Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics 9*, 2 (April-June 2003), 191–205.

OBS92. OKABE A., BOOTS B., SUGIHARA K.: *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. Wiley Publishing, 1992.

SG03. SURAZHSKY V., GOTSMAN C.: Explicit surface remeshing. In *Proc. Symp. on Geometry Processing* (2003), ACM/Eurographics, pp. 20–30.

VC04. VALETTE S., CHASSERY J.-M.: Approximated centroidal voronoi diagram for uniform polygonal mesh coarsening. *EUROGRAPHICS 23*, 3 (2004), 381–389.

VKC05. VALETTE S., KOMPATSIARIS I., CHASSERY J.-M.: Adaptive Polygonal Mesh Simplification With Discrete Centroidal Voronoi Diagrams. *2nd International Conference on Machine Intelligence ICMI 2005* (November 2005), 655–662. Tozeur, Tunisia.