Volumetric Region Guarding and Star Decomposition for Harmonic Parameterization

ABSTRACT

We present an effective framework to compute the *guarding* and the *star decomposition* for 3D volumetric regions represented by polyhedral meshes. Based on the observation that curve-skeletons extracted from 3D objects have good visibility to the object boundary, we compute guarding using skeletal nodes. A hierarchical integer linear programming framework is developed to efficiently reduce the computational complexity in finding a hierarchical approximation to these NP-complete problems. The decomposition framework proposed in this work could potentially benefit many volumetric data processing tasks. One important motivation for this work is the seeking of bijective harmonic volumetric parameterization. We demonstrate the application of star-shaped decomposition in the non-degenerated 3-manifold mapping by converting the problem to solvable sub-domains with mathematically guaranteed bijectiveness.

Keywords

Volumetric Data Segmentation, Star Decomposition, Harmonic Volumetric Mapping, Polyhedron Guarding

1. INTRODUCTION

Shape decomposition is a ubiquitous technique facilitating various geometric processing and analysis tasks. The rapid advancement of 3D scanning techniques provides us massive geometric data sets nowadays with great ease. When the size of input data are very large, direct computation can be expensive; and when the topology and geometry of the input data are very complicated, the processing of the entire domain can be infeasible. A common approach for above difficulties is through a divide-and-conquer strategy that partitions the problem into solvable sub-domains. Also, many applications such as object recognition, classification, matching, and etc. need to solve the problem of segmenting a shape into a set of salient parts.

In this paper, we study the star-shape decomposition that segments a 3D volumetric region to a set of sub-regions, each of which is visible from a guarding point. Such a sub-region is called a star shape. The motivation for this work comes from computing lowly distorted volumetric parameterization which seeks bijective harmonic functions defined on 3D-manifolds that embed in $\mathcal{R}^3$. Low-distortion parameterization facilitates many geometric modeling tasks in graphics and vision research. For example, in computer graphics, it is used for texture mapping, texture transfer, and animation morphing; in geometric processing, it is used for detail transfer, surface editing, and mesh simplification; in CAGD, it is used to construct domains for scientific functions such as splines; and in vision and medical imaging, it has been used for surface matching, data completion. Surface parameterization has been extensively studied (see the survey papers [10, 34, 16]). Recently, the parameterization of 3-manifolds has also gained great interests due to its important applications in volumetric data processing. However, its study is still in infancy, and mapping techniques developed are far from adequate. One reason is due to the much more complicated topological structure and sizes of volumetric data. Another reason is that the existence of bijective harmonic functions on general volumetric domains is not theoretically guaranteed. An effective star decomposition framework could be a solution to tackle both of these challenges in harmonic volumetric parameterization.

As a commonly used divide-an-conquer strategy, shape decomposition has been widely studied. For example, surface segmentation has been thoroughly examined, with two great survey papers given in [1] and [33]. Most decomposition methods define a criterion to measure specific geometric properties of each (sub-)region, so that a complex objects can be partitioned into small sub-patches satisfying the given criterion, facilitating the target applications. Convexity, curvedness, symmetry and many other criteria have been studied in shape decomposition. However, less study has been conducted to the decomposition based on visibility. Art-gallery guarding and star-decomposition have been studied in computational geometry community on 2D planar polygonal meshes and in 3D for terrain guarding. However, very few star-decomposition methods are for 3D free-form volumetric objects. To our knowledge, the most related work is from [26], which provides an effective iterative greedy algorithm to compute approximate guarding on geometric objects represented by point clouds. However, there are two main reasons that [26] is not enough for our application: (1) [26] computes the guarding instead of the star-decomposition on the given point cloud data, while we also need the star-decomposition on polyhedra meshes, so

that different sub-regions can actually be processed separately (see Section 4 for more discussions). (2) The guarding of [26] is an approximate result, bounded by a threshold ϵ (since the point-based representation does not have explicit connectivity), while in our setting, with a watertight polyhedra mesh, we ask for an exact guarding, so that the subsequent geometric modeling tasks relaying on the star property of the sub-domains can proceed.

Main contributions of this work include:

1. We develop an effective hierarchical optimization algorithm in computing the guarding for a 3D volumetric region represented by a polyhedral mesh.

2. We study the relationship between the region guarding and star-shape decomposition. Within the same computational framework, an efficient region growing algorithm, seeded from guarding point set, is developed.

3. We demonstrate the application of the star-shape decomposition in harmonic volumetric parameterization. On each sub-region, our constructed mapping has its bijectivity guaranteed. A parametric representation is defined on the guarding skeleton graph.

The rest of this paper is organized as follows: we first discuss the related work and background in star decomposition and volumetric parameterization in Section 2. Then we introduce our algorithms for guarding computation (Section 3) and star-shape decomposition (Section 4). Its application on volumetric parameterization is discussed in Section 5. We show the experimental results in Section 6 and conclude the paper in Section 7.

2. BACKGROUND AND RELATED WORK

As a commonly used divide-an-conquer strategy, shape decomposition has been widely studied in geometric modeling and computer graphics. We studies the decomposition based on the concept of visibility.

DEFINITION 1 (VISIBILITY). Given a shape M, two points $p, q \in M$ are **visible** to each other, if and only if

$$\overline{pq}^{\circ} \cap \partial M \equiv \emptyset.$$

where ∂ denotes the boundary operator, and the $\overline{pq}^{\circ}$ denotes the open line segment (with the boundary removed): $\overline{pq}^{\circ} = (\overline{pq} - \partial\overline{pq}) = \overline{pq} \backslash \{p, q\}$.

DEFINITION 2 (STAR SHAPE). A region M is a **star shape** if and only if $\exists p \in M$ such that $\forall q \in M$, q is visible to p. p is called a **guard** of M.

DEFINITION 3 (STAR DECOMPOSITION). Given a region M, a **decomposition** $d(M)$ is defined as:

$$d(M) = \{M_i | \bigcup_i M_i = M \text{ and } \forall i \neq j, M_i^{\circ} \cap M_j^{\circ} = \emptyset\}$$

A **star decomposition** is a decomposition such that every subregion M_i is a star shape. We say a region M admits a **star-decomposition of size** n, if it has a star decomposition composed by n star subregions.

Decomposition and Region Guarding. Many different decomposition methods (e.g. Voronoi decomposition, convex decomposition, etc.) have been proposed. A thorough review on these topics is beyond the scope of this work and we refer the readers to surveys [3] and [23]. In computer graphics and visualization, surface segmentation has been studied for different applications such as object recognition, meshing, skeleton extraction, and many others. Two thorough surface segmentation surveys were given in [1] and [33], in both of which, segmentation techniques are classified as *surface-based* methods (in which surface properties of sub-surface-patches are used as the segmentation criteria) and *part-based* methods (in which volumetric properties of sub-solid-regions are the criteria).

Star-shaped decomposition is related to guarding an art gallery [4]. A region is considered to be guarded if all its interior points are visible to some guard point. In the 2D case, guarding a general polygon is known to be NP-complete, and even the approximate approaches also have high complexities. In the 3D case, [2] proposed an approximate approach to guard terrains. Decomposing simple 2D polygons into star shapes is well studied (see [23]), but when polygons have holes, the star decomposition is again NP-complete ([22]). 3D star decomposition, in contrast, has been little explored. To our knowledge, the only closely related work that deals with general 3D free-form shapes is from Lien [26], which computes the approximated guarding on 3D point cloud data using the octree structure.

Volumetric Parameterization. Surface parameterization computes a one-to-one continuous map between a 2-manifold and a target domain with low distortions. It plays a critical role in various applications in graphics, CAGD, visualization and vision. A thorough survey of surface parameterization techniques is beyond the scope of this work, and we refer readers to surveys ([10, 34, 16]) for details. Many effective surface manipulation techniques and parameterization paradigms can be generalized onto 3-manifolds. When volumetric data are to be processed, volumetric parameterization has been studied for various applications such as shape registration ([36, 25]), volume deformation ([19, 18, 28]), and trivariate spline construction ([30]). Wang et al. [36] parameterized solid shapes over solid sphere by a variational algorithm that iteratively reduces the discrete harmonic energy defined over tetrahedral meshes, the harmonic energy is rigorously derived but the optimization is prone to getting stuck on local minima and it only focuses on spherical like solid shapes such as human brain data sets. Ju et al. generalized the mean value coordinates [9] from surfaces to volumes for a smooth volumetric interpolation, Joshi et al. [18] presented harmonic coordinates for volumetric interpolation and deformation purposes, their method guaranteed the non-negative weights and therefore led to a more pleasing interpolation result in concave regions compared with that in [19]. Li et al. [25] used the method of fundamental solutions to map solid shape onto general target domains. Later, Lipman et al. [28] designed the Green coordinates based on the same kernel functions for Laplace fundamental solutions, and applied it to cage-based deformation. Martin et al. [30] parameterized genus-0 tetrahedral mesh onto a cylinder with two singularity points, and used it for trivariate spline construction. However, none theoretic guarantee on bijectiveness of the volumetric mapping has been provided in aforementioned mapping computation

methods. In fact, the Radó theorem that lays down the foundation of surface harmonic mapping does not hold on general volumetric domain M even when M has the trivial topology.

3. GUARDING VOLUMETRIC OBJECTS

Given a volumetric region M, finding suitable positions of its optimal guarding points is a challenging problem. Under the discrete triangulation setting, one common approximation is to tetrahedralize (densely enough) the D and select guards from all the vertices. However, even in 2D, finding the optimal vertex set to guard a general polygonal domain is NP hard. A common 3D scanned object or CAD model usually has more than $20K$ sampling vertices on its boundary triangle mesh, whose tetrahedral mesh could then have the vertices number reaches $3M$ easily, which is way too large to solve for an NP problem. Effective approximation schemes, such as randomly seeded iteration methods such as [26] has been proposed for point cloud data. However, decomposition computed via the approach like [26] may depend on initial selection; and the guarding skeleton could change globally due to local shape perturbations. In real geometric applications such as shape matching, although local geometric noise usually exists in scanned 3D data, we would prefer the shape decomposition is globally stabled. Therefore, we want to seek a deterministic scheme that has controllable complexity and is less sensitive to local geometric noise. Our algorithm is based on the following observations:

- As demonstrated in several medical visualization and virtual navigation applications [13, 20], curve skeletons usually have desirable visibility to boundary points (referred as the "reliability" of skeletons in these literatures). Skeletons based on medial axes are usually computed to ensure that the interior organ surface is fully examined (visibly covered).
- Hierarchical skeletons or skeletons for a progressively simplified mesh, can be effectively computed and used for reducing the size of the optimization problem, leading to a computation of not only better numerical efficiency but also better robustness.

Many effective skeletonization algorithms (see the nice survey paper of [6]) have been developed for given 3D shapes. We use the algorithm of [7] since it efficiently generates skeletons on medial-axis surfaces of the 3D shapes. Suppose the boundary surface ∂M of a volumetric region M is represented by a triangle mesh (also denoted as ∂M) with n vertices, and the output skeleton has k nodes, the guarding problem is then converted to finding a minimal-size point set G from this k points, such that all n boundary vertices are visible to G.

3.1 Visibility Detection

Suppose we want to detect the visibility of a (skeleton) point p on a triangle mesh $\partial M = \{T, V\}$ with n vertices $V = \{v_1, v_2, \ldots, v_n\}$. We shall check intersection points of the each line segment $\overline{pv_i}$ and ∂M. If an intersection point $q \in \partial M$ exists other than v_i (which indicates the Euclidean distance $|\overline{pq}| < |\overline{pv_i}|$), then v_i is not visible from p. Simply enumerating every $\overline{pv_i}$ then detecting its intersections with each triangle $t \in T$ is time consuming: for a single skeleton point p, it costs $O(n_V \cdot n_T) = O(n^2)$ time to check its visibility on all n vertices. We develop the following **sweep algorithm** to improve the efficiency.

input : A point p and a triangle mesh $\partial M = \{T, V\}$
output: The visibility of p on $\{v_i\}, v_i \in V$
For $\forall v_i \in V$, compute $\theta(\overline{Ov_i})$ and the $\varphi(\overline{Ov_i})$;
Compute min and max of $\theta(t_i)$ and $\varphi(t_i)$, $\forall t_i \in T$, conduct necessary duplication;
Sort all $\overline{Ov_i}$ to a queue $Q = \{\overline{Ov_i}\}$, following the ascending order of (θ, φ)
Let the active triangle list $L = \emptyset$;
foreach $\overline{Ov_i} \in Q$ **do**
  $c_j \leftarrow c_j + 1$ for all t_j incident to v_i;
  if $c_j = 1$ **then**
    Insert t_j to L;
  end
  Detect intersections between $\overline{Ov_i}$ and triangles in L
  if $\exists$ *intersection point* $q \neq v_i$ **then**
    v_i is **invisible** from p;
  else
    v_i is **visible** from p;
  end
  If $c_j = 3$, remove t_j from L.
end

Algorithm 1: Sweep Algorithm for Visibility Detection.

For every skeleton point p, we create a spherical coordinate system, making p the origin o. Each vertex $v_i \in V$ can be represented as $\overline{pv_i} = \overline{ov_i} = (r(v_i), \theta(v_i), \varphi(v_i))$, where $r(v_i) \geq 0, -\pi < \theta(v_i), \varphi(v_i) \leq \pi$. For every triangle $t_i \in T, 1 \leq i \leq n_T$, we define:

$$\theta_{max}(t_i) = \max\{\theta(v_{i,j})\}, 1 \leq j \leq 3$$
$$\theta_{min}(t_i) = \min\{\theta(v_{i,j})\}, 1 \leq j \leq 3$$
$$\varphi_{max}(t_i) = \max\{\varphi(v_{i,j})\}, 1 \leq j \leq 3$$
$$\varphi_{min}(t_i) = \min\{\varphi(v_{i,j})\}, 1 \leq j \leq 3$$

where $v_{i,j}$ are three vertices of the triangle t_i. The segment $\overline{ov_k}$ cannot intersect with a triangle t unless

$$\begin{cases} \theta_{min}(t) \leq \theta(v_k) \leq \theta_{max}(t) \\ \varphi_{min}(t) \leq \varphi(v_k) \leq \varphi_{max}(t), \end{cases} \quad (1)$$

and therefore we can ignore many triangles that do not satisfy this condition.

The angle function θ is not continuously defined on a circle. When a triangle t spans $\theta = \pi$, it needs special handling. In our algorithm, we find all such triangles t by checking whether $\theta_{max}(t) - \theta_{min}(t) \geq \pi$. Then we duplicate these triangles to ensure that each θ of the original t is between $[\theta_{min}(t), \theta_{min}(t) + 2\pi)$ and θ of its duplicate is between $[\theta_{max}(t), \theta_{max}(t) + 2\pi)$, by adding or subtracting θ by 2π. For each triangle t spans $\varphi = \pi$, we detect and duplicate it in the same way.

Using $\theta(v_i)$ as the primary key and $\varphi(v_i)$ as the secondary key, we then sort all line segments $\overline{ov_i}$ that we need to check. Then we sweep all segments one by one: for each segment, we filter out triangles not satisfying condition (1). Specifically, we use a counter c_i on every triangle t_i. Initially, $c_i = 0$; when the segment $\overline{ov}, v \in t_i$ is being processed, $c_i \leftarrow c_i + 1$. The following two cases indicate that the sweep has not reached the neighborhood of the triangle t_i, and we do not need to check its intersection with line segment $\overline{ov}$:

$c_i = 0$: i.e. $\theta_{min}(t_i) > \theta(\overline{ov})$, or $\varphi_{min}(t_i) > \varphi(\overline{ov})$.

$c_i > 3$: i.e. $\theta_{max}(t_i) < \theta(\overline{ov})$, or $\varphi_{max}(t_i) < \varphi(\overline{ov})$.

Therefore we maintain a list L of neighboring triangles $\{t_i\}$ whose counters have $1 \leq c_i \leq 3$. When the sweep segment hits a new triangle t_j, we have $c_j = 1$ and add t_j into L; when a counter $c_j = 3$, then after processing the current segment we remove t_j from L.

Algorithm 1 describes the above pipeline. Given a skeleton point p, for a boundary triangle mesh with n vertices it takes $O(n \log n)$ to compute and sort angles of all segments. For each segment, if the size of the active triangle list L is m, it takes $O(m)$ intersection-detecting operations. Therefore, the total complexity is $O(logn + nm)$. The incident triangles around a vertex v_i is usually very small, usually we have $m < \log n$. Therefore the algorithm computes the visibility of p in $O(n \log n)$ time. On a skeleton containing k nodes, it takes $O(kn \log n)$ pre-computation time to know the visible region for all nodes.

3.2 Greedy and Optimal Guarding

Once we have the visibility information for all skeletal nodes, we want to pick a minimum sized point set that can cover all mesh vertices. This now can be converted to a classical set-covering problem, shown to be NP-complete [21]: given the universe $V = \{v_i\}$, and a family S of subsets $S_j = \{s_{j,k}\}, s_{j,k} \in V$, a cover is a subfamily $C \subset S$ of sets whose union is V. We want to find a covering C that uses the fewest subsets in S. Here V is the set of all mesh vertices, S contains the visibility to V from each skeletal node. C indicates a subset of skeletal nodes that can guard the entire region. Skeletons generated using medial-axis based methods with dense enough nodes usually ensure S itself is a covering. This holds for all our experiments. If a coarsely sampled skeleton can not cover the entire V, we can include all those invisible vertices into the skeleton point set.

A simple **greedy** strategy for the set covering is as follows: we iteratively pick the skeletal nodes that can cover the largest number of unguarded vertices in V, then remove all guarded vertices from V (and update S accordingly since the universe becomes smaller), until $V = \emptyset$. Greedy strategies have been shown effective, and they yield $O(\log n)$ approximation [17].

An **optimal** selection can be computed by $0-1$ programming, also called Integer Linear Programming (ILP). For every skeleton point $p_i, i = 1, \ldots, m$, we assign a variable x_i such that

$$x_i = \begin{cases} 1 & \text{if } p_i \text{ is chosen;} \\ 0 & \text{otherwise.} \end{cases}$$

The *objective function* to minimize is then $\sum_{i=1}^{m} x_i$

Since V should be covered (i.e. all vertices should be visible), for $\forall v_i \in V$, visible to skeletal nodes $P_i = \{p_{(i,1)}, \ldots, p_{(i,k)}\}$, at least one node in P_i should be chosen to ensure v_i guarded.

Therefore, we minimize $\sum_{i=1}^{m} x_i$, subject to

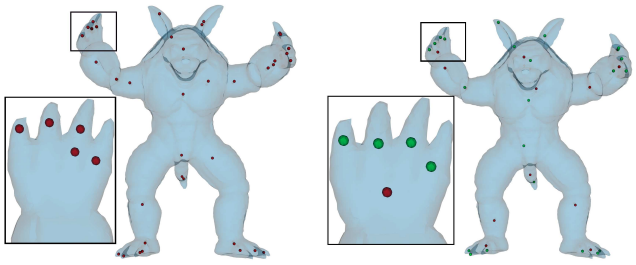
$$x_i = 0, 1, (i = 1, \ldots, n), \text{ and}$$

$$\sum_{j \in J(i)} x_j \geq 1, (i = 1, \ldots, n)$$

where $J(i)$ is the index set of skeletal nodes p_j that is visible to vertex v_i.

The above optimization can be solved using branch-and-bound algorithms. When the dimension is smaller enough, we can use the TomLab Optimization package [**?**] to solve it efficiently.

3.3 Hierarchical Guarding



(a) Greedy Guarding (b) HILP Guarding

Figure 1: Greedy and HILP Guarding. On a volumetric Armadillo model, comparison between guarding by greedy algorithm (a) and by HILP algorithm (b).

To compute the optimal guarding based on ILP we need to solve a nonlinear optimization. This limits the size of problems that we can handle. General 3D volumetric shapes can easily have a number of $20k$ to $200k$ vertices on its boundary surface, which is too large for such a nonlinear optimization. On the other hand, greedy algorithm generates the guards in a locally optimal manner. Furthermore, the greedy strategy is not robust against local geometric perturbations. For example, a small bump could lead to global structural variance of the guarding points. We propose a hierarchical guarding computation framework based on the progressive mesh [**?**], combining the $0-1$ programming optimization and the adaptive greedy refinement.

We simplify the boundary mesh ∂M into several resolutions $\partial M^i = \{T^i, V^i\}, i = 0, \ldots, m$ using progressive mesh [15]. In the coarsest level $i = m$, ILP optimization is conducted on all vertices $v \in \partial M^m$ and we get the coarsest level guard set $G^i = \{g_k^i\}$. Then we progress to $i = m - 1$ level $\partial M^{m-1} = (T^{m-1}, V^{m-1})$, we first conduct a local greedy adjustment on existing guards. Since the skeleton S^i varies, we adjust $\{g^{i+1}\}$ to local skeletal nodes that cover largest number of vertices in V^i; and then we remove these already covered vertices from current level V^i.

Current guards $\{g^i\}$ (refined from the coarser level $\{g^{i+1}\}$ might not be enough to guard all boundary vertices in the current level. We conduct four **reduction** (see below) operations, then apply the ILP optimization on uncovered regions for necessary insertion of new guards, and include them into G^i. This progressive refinement adds necessary new guards when the detail on boundary surface increases, and ends when all boundary vertices are covered on the finest level $i = 0$.

Reduction. The size of the ILP optimization can be significantly reduced in our hierarchical framework. Suppose we store the visibility information in an incidence matrix A. If the skeletal node p_i can see the vertex v_j, $a_{ij} = 1$, otherwise $a_{ij} = 0$. Originally the A is $|S| \times |V|$. The following **four rules** are applied to reduce the size of A:

1. If column j has only one non-zero element at row i, we must take p_i in order to see v_j. Therefore, add p_i into

G, remove column j. Also, for all non-zero element a_{ik}, remove column k (we take p_i, all points it sees are guaranteed to be covered, thus now can be removed).

2. If row i_1 has all its non-zero elements non-zero in row i_2, i.e. $a_{i_1,j} = 1 \rightarrow a_{i_2,j} = 1$, then p_{i_2} sees all vertices that p_{i_1} can see, and we can remove the entire row i_1.

3. If column j_1 has all its non-zero elements non-zero in column j_2, i.e. $a_{i,j_1} = 1 \rightarrow a_{i,j_2} = 1$, then guarding v_{j_1} guarantees the guarding of v_{j_2}, and we can remove the entire column j_2.

4. If the matrix A is composed by several blocks, we partition A to several small matrixes $\{A_k\}$.

In step 4, since we remove vertices that have been seen by the adjusted guards from a coarser level, remaining boundary vertices could be partitioned to several connected-components far away from each other. And these sub-components may be optimized separately. This significantly reduces the size of the optimization.

In our experiments, we simplify the boundary mesh to the coarsest level with 5k vertices for the first round ILP optimization. Generally, we make each iteration adding in another 10k vertices. When the size of constraints is around 5k, and the size of variables (skeletal nodes) is around 1k, the optimization usually takes 10-50 seconds to solve.

Our hierarchical scheme together with the reduction processing, has the following important **advantages** over both the pure greedy strategy and the pure $0-1$ optimization:

- It is **much faster** than the non-linear ILP optimization. The current framework can handle large-size geometric shapes.
- With similar performance, it usually provides **better** guarding solutions than pure greedy strategy. Especially when the boundary geometric details are not complicated, HILP computes better global guarding in comparison to greedy algorithms.
- More importantly, it is **hierarchical** and therefore is **robust** against geometric noise. Figure 1 shows an example. In our HILP framework, refined local details will not change the global structure of the previously optimized guarding graph in coarser levels.

The optimization algorithm discussed in this section is now used to **efficiently** compute the guarding for large-size 3D volumetric regions and generate a **hierarchical** guarding graph. The entire pipeline is fully automatic.

4. STAR-SHAPE DECOMPOSITION

The process of decomposing a 3D region into star shapes is usually discussed as the guarding problem and oftentimes when we get the guards, we directly go on and partition regions apart based on these seeds (guards). However, it can be shown that guarding and decomposing a region could be different. *A region M that can be guarded by n points not necessarily admits a star-decomposition of size n.* Figure 2 illustrates an example in 2D. The left figure shows that two guards are enough to see the entire region (P sees red, white and orange regions, and Q sees green, white, and brown regions). However, we cannot find a decomposition

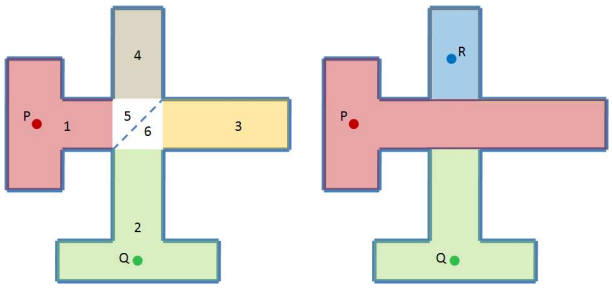


Figure 2: Guarding versus Decomposition. (Left) P and Q are enough to guard the entire region. But star-decomposition of size-2 does not exist. (Right) Star decomposition computed by our algorithm.

with just 2 star regions. The shared white region can't be included in both sub-parts, each of which has to be a 1-connected-component and visible from one point. A possible star-decomposition is shown in the right figure.

There are generally two categories of approaches for decomposition algorithms. One is **top-down**, by iteratively segmenting sub-parts to finer components. And the other is **bottom-up**, by iteratively gluing small elements/components to larger parts. For example, the Approximate Convex Decomposition [27] is a top-down approach. It iteratively measures the convexity of each (sub-)region M; if it fails to satisfy the convexity criterion, it shall be further *cut* into two sub parts M_1 and M_2. The algorithm continues until all sub-regions are convex enough. A difficult issue in **top-down** methods is to find the suitable *cut* so that shapes of smaller regions become nice in a few steps. On the other hand, popular surface segmentation techniques such as region growing ([5],[24]), watershed ([29]), or clustering ([35],[11]) algorithms are bottom-up approaches. These approaches start from a set of seeds, then expand to include neighboring primitives (vertices, faces, tetrahedra) until their unions cover the entire region.

4.1 Region Growing and the Dual Graph

We choose to use a region growing algorithm for the star decomposition. There are mainly two reasons. (1) It seems very difficult to find a suitable and robust cutting surface for top-down volumetric decomposition algorithms, yet making it efficient. (2) We can generalize the sweep algorithm discussed previously to tetrahedral mesh vertices for the efficient pre-computation on the visibility from any tetrahedral vertex to each guard. The pre-computation takes $O(m \log m)$ time, where m is the total number of vertices of the tetrahedral mesh. We can simultaneously preserve the star-property for all growing regions.

Suppose we have a star region M, guarded by K_g guard points $G = \{g_i\}, i = 1, \ldots, K_g$, each g_i has a specific color c_i. We grow outward from these seeds simultaneously. Region growing becomes a iterative procedure that assigns a unique color c_i to each tetrahedron, so that at the end, the connected component in color c_i is a star shape guarded by g_i. It can be easily shown that if M needs K_g guards to cover, then a star decomposition of M is at least of size K_g (i.e. K_g separate sub-regions). Therefore, starting from K_g seeds and starting from these guards is a nice choice (we might need to grow new sub-regions when necessary, details

discussed in Section 4.2). We conduct the region growing of the tetrahedral mesh on its dual graph. We first define the dual graph with necessary notations and operations.

Definition 4 (Visibility Dependency). We say that **a tetrahedron is visible** from a point g if all its four vertices are visible from g. With respect to g, we say a tetrahedron t is **visibly dependent** on a set of tetrahedra $T(g,t) = \{t_i\}$, if that t is visible to g if and only if $\forall t \in T(g,t)$ is visible to g.

Intuitively, $T(g,t)$ contains all tetrahedra that the four ray segments $\overline{gv_j}, v_j \in t$ from g pass through. If we denote $F_n(v)$ as the **one-ring faces** surrounding a vertex v (i.e. $F_n(v) = \{f | \forall t, v \subset t, f \subset t, v \not\subset f\}$); and denote $T_n(t) = \{\cup_{v_j \in t} t' | F_n(v_j) \subset t'\}$; then we can define $\widetilde{T}(g,t) = T(g,t) \cap T_n(t)$. Intuitively, including tetrahedra in the same sub-region prevents triangles in $F_n(v_j)$ (which could block the visibility of t) from becoming the boundary of the sub-region. It is not difficult to further show that

(a) t is visible if and only if $\forall t' \in \widetilde{T}(g,t)$ are visible.

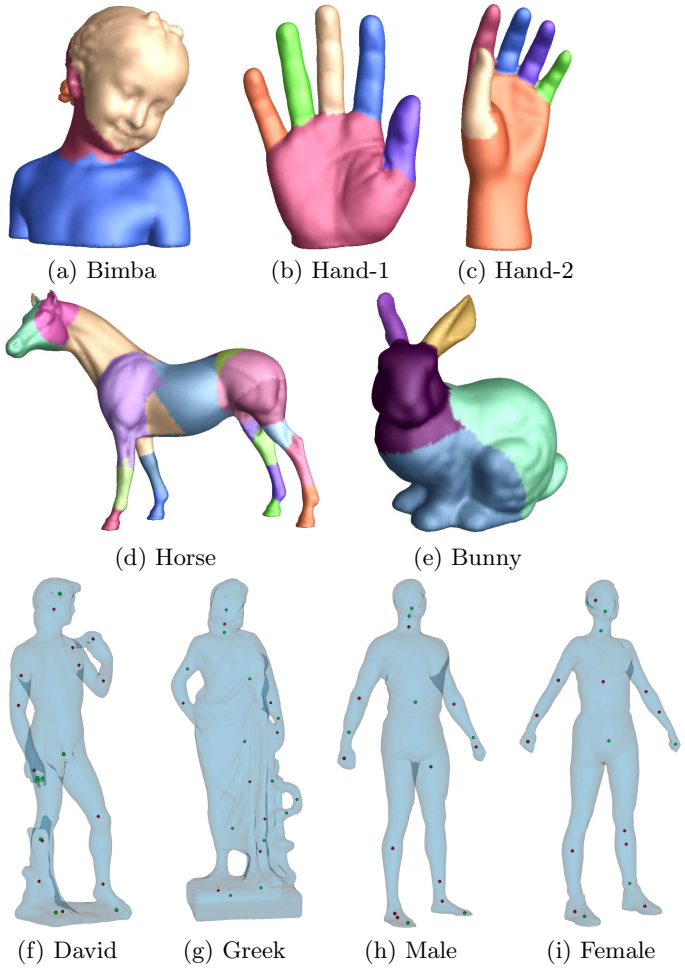
(b) t can be safely added into a sub-region M_g seeded in g without violating its star-property, if and only if all $t' \in \widetilde{T}(g,t)$ are in M_g.

Following (b), if a tetrahedron t is visibly dependent on several tetrahedra $\widetilde{T}(g_i,t)$ with respect to g_i, but one $t' \in \widetilde{T}(g_i,t)$ has been included to another sub-region (seeded on $g_j, j \neq i$), then t will be no longer visible to g_i and cannot be included into D_{g_i} during its growing.

Now we define the dual graph of the given tetrahedral mesh D. A **node** n_i is defined for each tetrahedron t_i. For a node n_i visible from g_k (with the color c_k), we connect a **directed edge in color** c_k to n_i from another node n_j if $t_j \in \widetilde{T}(g_k,t_i)$; and we call here n_j is a **color-c_k predecessor** of n_i. Since recursively, $T(g_k,t_i) = \widetilde{T}(g_k,t_i) \cup T(g_k,t_j)$, we only need to store each node's visibility dependency relationship. $\tilde{T}(g_k,t_i)$ can be computed in $O(1)$ time by checking the intersections of $\overline{g_k,v}$ and $F_n(v)$ for each $v \in t_i$.

For each guard g_k we generate a virtual node in the dual graph and connect it to nodes corresponding to all its one-ring tetrahedra. Then each guard g_k and its visible region defines a direct acyclic graph R_k. The entire 3D region guarded by K_g points $\{g_i\}$ corresponds to a big graph with K_g sources. Each source g_i has an individual color c_i, the region glowing assigns each node a unique color. A node n_j can be assigned by a color c only when all its color-c predecessors (on which n_j is visible dependent) are already assigned by color-c.

To conduct region growing in the dual graph, we take the operation of **node-merging**. When there is a color-c edge from a color-c node n_i to an uncolored node n_j, and all edges *entering* n_j are leaving from color-c nodes, then n_j can be safely colored by c, and included into the same region. Therefore we can merge them together to one node, in color-c, preserving all distinct outgoing edges. The region growing procedure is converted to iteratively merging each uncolored node to one of the "growing" K_g colored nodes. The entire procedure finally results in K_g colored nodes, and some nodes/components that can not be colored. Note that according to the visibility dependency relationship, if for a node n_k, one of its color-c_i predecessors has been assigned a different color c_j, then n_k can no longer be given the color c_i. If a node cannot be assigned to any color appeared on its entering edges, then it can not be colored and is not classified to any growing region.



(a) Bimba (b) Hand-1 (c) Hand-2 (d) Horse (e) Bunny (f) David (g) Greek (h) Male (i) Female

Figure 3: Guarding Computed using our HILP Framework. The upper row, from left to right: the volumetric Bimba, Hand and Hand(2) models. Boundary regions are colorized according to closest guards respectively. The bottom row, from left to right: the volumetric Michelangelo David, Greek, Cyberware Male, and Female. Small spheres show the guards, where green nodes are the newly inserted guards on the finest level.

4.2 Iterative Nodes-merging

We propose the following greedy strategy to merge nodes and simulate the seeds-growing.

Pre-processing Merging. Starting from K_g virtual source nodes, for each g_i, we first merge its one ring tetrahedra nodes. Then we iteratively collect and merge all neighboring nodes that only have color-c_i edges enter them (i.e. they are only visible by g_i exclusively), since they can be safely merged.

Pre-computing Cost Function. For each node n_i, we can compute how many nodes are directly or indirectly visually dependent on n_i with respect to g_k. This can be pre-

computed in linear time: after the dependency graph R_k is created from the source g_k to leafs, inversely the dependency cost can be accumulated and stored as $f(g_k, n_i)$.

Nodes Merging. After the preprocessing, we can conduct the region growing based on the cost function $f(g, n)$. For example, for an uncolored node n_i, if it has all red edges entering it come from the growing red node g, and it also has all green edges entering it come from the growing green node g'. Then at this stage, n_i can either be colored as red or green, i.e. merged to either g or g'. We then check $f(g, n_i)$ and $f(g', n_i)$, finding how many nodes are visibly dependent on n_i for g and for g' respectively. We merge n_i to the node whose corresponding cost function is larger.

Iterative Region Growing. During the region growing, each step we select the uncolored nodes with largest cost difference and colorize/merge it, we repeat this until no more node can be merged. If a node cannot be colorized to any color appeared on its entering edges, it is left unclassified after the region growing. We collect each uncolored connected components, and respectively compute their guarding and re-apply the region growing until all tetrahedra are colored. Figure 2 (right) illustrates an example. When we start the region growing from two seeds p and q, the blue region is left unclassified. We computes its guarding, and grow the region again to obtain the final star-decomposition of size 3. More decomposition examples in general volumetric models using this algorithm are shown in Figure 4.

Figure 4: Decomposing volumetric Rocker-arm, Torus-cone, David, Cyberware ware Male, Female, David, and Greek into star-shaped solid sub-regions. Different parts are rendered by different colors.

5. HARMONIC VOLUMETRIC PARAMETERIZATION

After the decomposition of a given object M, we get a set of star shapes $\{M_i\}$, each region being guarded by a point g_i. Then we can parameterize each sub-region onto a solid sphere. A key property that we will show shortly is that such a harmonic map is guaranteed to be bijective. The harmonic

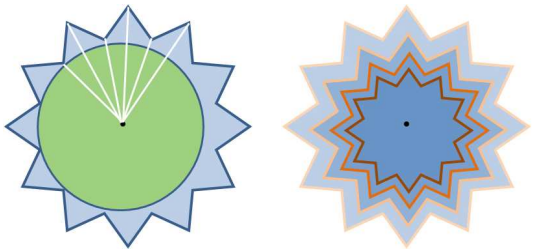


Figure 5: Parameterizing a Star Shape onto a Solid Sphere. (Left) Initial u and v coordinates can be projected from the boundary surface to the inscribed sphere. (Right) The w coordinate (depth) can be computed using methods of fundamental solutions [25].

map can be computed using the **method of fundamental solutions (MFS)** [25], by setting the boundary condition of the volumetric map to be the spherical parameterization of ∂M_i. We use the harmonic spherical parameterization [12] to get the harmonic surface map $f' : \partial M_i \to S^2$. [12] takes the normal map as the initial mapping and conduct the optimization on local tangential plane before projecting the adjusted position back to the sphere. If the initial map (like the normal map) has large flip-over regions, the optimization will be slow and could trap locally. Since each M_i now is a star shape, the following approach efficiently gets a bijective initial spherical mapping. Figure 5 illustrates our idea. In the left picture, full visibility of the local region guarantees a bijective projection from the boundary points onto the sphere. When the boundary mapping is decided, we only need to compute the third dimensional coordinate w of the parameter. Figure 5 (Right) illustrates the concept of the different iso-w level sets. Such a harmonic function w can be computed using the MFS method. We refer to [25] for technical details, and only briefly recap the idea as follows.

MFS for Volumetric Parameterization. Based on the Green's function and the maximum principle of harmonic functions, the harmonic function w defined on a region D never reaches maximal or minimal values in the interior of D, and w is fully determined by the boundary condition and can be computed by fundamental solutions of the 3D Laplace equation. The kernel function for the 3D Laplace equation coincides with the electric potential produced by point charges. Therefore, its intuitive physical explanation is to design a potential field that approximates the boundary condition. The potential field, guaranteed to be harmonic by the fundamental solutions, is the function w that we seek for. The parameterization therefore is converted to a boundary fitting problem for the potential field, and can be solved using a linear system effectively.

5.1 Bijectiveness of Harmonic Mapping on Star Regions

In the surface case, a harmonic map is a minimizer of the Dirichlet energy and indicates a minimal surface [32]. It can be effectively approximated through FEM analysis of harmonic energy [8]. The theoretic foundation for harmonic surface mapping is built upon the Radó Theorem, which states that, on a simply connected surface M with a Riemannian metric, suppose a harmonic function $f : M \to D \subset \mathbb{R}^2$ maps M to a convex planar domain D, if f on the boundary

is a homeomorphism then f in the interior of M is also a diffeomorphism.

FEM analysis of harmonic energy can also be conducted [36] on 3-manifolds using tetrahedral meshes. However, the Radó theorem does not hold for 3-manifolds. Therefore, fundamental theoretic obstacles remain for volumetric harmonic mapping. We tackle this fundamental parameterization problem for volumetric data through star decomposition. It can be proved that for specific domains such as convex shapes, bijective harmonic parameterization exists. Then in order to compute a bijective mapping, we can first decompose volumetric data into a set of solvable subdomains for local piecewise mapping computation. We show ([14] gives a rigorous mathematic proof through analyzing the induced foliation) the existence of harmonic volumetric parameterization on a star-shaped region and that the bijective map can be constructed using the MFS-based framework effectively. The idea is as follows.

LEMMA 1. *In a star-shaped domain M guarded by a point g, the harmonic function $w : M \to [0, 1]$ has its only critical point at g.*

When M is visible to a guard g and a harmonic function w is defined over M, with $w|_{\partial M} = 1$ and $w|_g = 0$. We can use g as the origin O and create a local coordinate system. Then we analyze the gradient of the harmonic function ∇w within this local coordinate system. At $p(x_1, x_2, x_3)$, we can define another harmonic function $h(p) =< p, \nabla w >$, where $<,>$ denotes the dot product. Since w is harmonic, $\frac{\partial w}{\partial x_i}$ is also harmonic, then we can verify $h(p)$ is harmonic by:

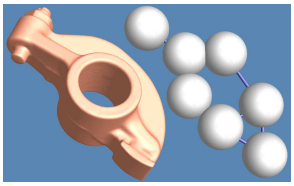
$$\Delta h = (\sum_k \frac{\partial^2}{\partial x_k^2})(\sum_i x_i \frac{\partial w}{\partial x_i}) = 2\Delta w + \sum_i x_i \Delta(\frac{\partial w}{\partial x_i}) = 0.$$

The maximum principle of harmonic map guarantees h reaches its max and min values only on the boundary of the harmonic field, i.e. surface boundary ∂M and the infinitely small ball bounding O, $\partial B(O, \epsilon)$. ∇w has same direction of p so it is easy to see that $f > 0$ in all the region bounded by $\partial B(O, \epsilon)$ and ∂M. Therefore, for arbitrary ϵ, $f \neq 0$, we have $\nabla w \neq 0$ in $M/\{O\}$, the harmonic potential is proved to have no critical point in M except O.

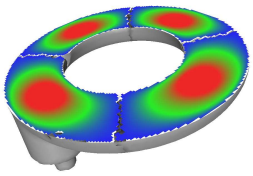
THEOREM 1. *Given a potential value r, the level set $w^-1(r)$ is a topological sphere.*

This is guaranteed by Morse theory [31], which says that two level sets share the same topology if there does not exist critical point between these two layers. Therefore, based on Lemma 1, all interior iso-layer $w^{-1}(r)$ has the same topology with the region boundary. This demonstrates the harmonic function w computed using MFS, together with the other two coordinates u, v defined by surface mapping, induce a bijective spherical map $M \to S^2$, which is also a diffeomorphism.

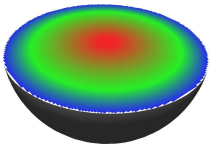
We conduct our harmonic volumetric parameterization on decomposed sub-regions. And verify the signed volume of each tetrahedron when mapped onto the target object. Under the mapping, all the deformed tetrahedra still preserve the positive volume. Which demonstrates the non-degeneracy of our parameterization. In the interest of space, we do not show the improvement in the mapping distortion (where the quasi-conformality of the harmonic mapping also effectively increases), which will be further discussed in our future work (see Section 7).



(a) Parameterization of the volumetric Rocker-arm



(b) Torus-cone (4-parts)



(c) Spherical level set

Figure 6: Harmonic Parameterization of Rocker-arm and Torus-cone. (a) The volumetric rocker-arm model, its spherical parametric domain, and the volumetric parameterization visualized in one cross-section. (b) The volumetric torus-cone model is decomposed to 4 star regions, also, the color-encoded w distance field is visualized in one cross-section. (c) Each sub-region is parameterized onto a solid sphere, whose color-encoded level set is visualized.

Models	Size (#(Tet))	Time (s)	Parts
Rocker-Arm	130,069	72.3	7
Torus-Cone	112,930	30.1	4
David	76,170	51.6	24
Greek	79,951	62.1	25
Female	81,067	49.5	21

Table 2: Star-Decomposition Runtime Table. The size (number of tetrahedra) of each mesh, and the computational time (in seconds) for iterative region growing (initially seeded on guards) are shown.

6. EXPERIMENTAL RESULTS

We conduct experiments on several volumetric objects. The computation are run on an AMD Athlon(tm) 64 X2 Dual Core Processor 5000+, 2.60GHz, with 2.0G memory. The statistics for guarding computation are illustrated in Table 1. We compare our framework with the pure *Integer Linear Programming* and the pure *Greedy Algorithm*. Although the *ILP* provides the optimal guarding, it is too time-consuming for large data. Large models like Amadilo, with 20k vertices on the boundary triangle mesh, could not be handled. The *Greedy* Algorithm and our *Hierarchical Integer Linear Programming (HILP)* has similar computation performance. However, HILP generally provides better guarding. Specifically, due to its ILP optimization component, it usually guards the region using less points globally. When the object has complex boundary details, the multi-resolution refinement in HILP might introduce more points for these refining details and result in more guarding points than greedy algorithms (e.g. David and Cyberware Male). However, even in such situations, a hierarchical result is still desirable, by providing better global robustness on the guarding skeleton structure and shape against geometric noise on the boundary shape.

For star decomposition, the region growing runtime table

Model Size			Number of Guards Needed			Computation Time (Second)		
Models	#(Tet)	#(B-Vert)	G_k(ILP)	G_k(Greedy)	G_k(HILP)	t(ILP)	t(Greedy)	t(HILP)
Armadillo	192,142	20,002	–	38	33	–	590.8	593.5
Female	81,067	10,002	13	18	18	2,046.2	279.1	286.1
Male	83,313	10,002	14	16	18	3,074.3	312.6	316.2
Greek	79,951	9,994	15	22	21	4,122.4	307.4	305.4
David	76,170	9,996	16	20	22	107,391.2	245.1	254.9
Hand	126,920	10,002	–	6	5	–	196.1	206.2
Hand-2	130,964	10,000	–	7	5	–	223.7	223.8
Bimba	226,126	10,002	–	5	4	–	198.2	210.1
Rocker-arm	130,069	11,350	–	7	5	–	251.5	274.3
Bunny	194,111	10,002	–	5	5	–	229.5	231.5
Horse	227,781	19,850	–	–	19	–	–	523.2

Table 1: Statistics for the Guarding of Different Volumetric Objects. #(Tet) and #(B-Vert) are the number of tetrahedra in tet-meshes and the number of vertices on the boundary triangle mesh. G_k indicates the number of guards necessary computed by different computation methods. t shows the computational time in seconds.

is shown in Table 2, excluding the initial guarding computation. As we can see the iterative region growing conducted on the dual graph of the tetrahedral mesh are very efficient (the necessary consecutive guarding computation time on the unassigned region is also included).

Figure 3 illustrates guarding results on the volumetric objects such as volumetric Bimba, Hands, horse, and bunny models. In the first two rows, we visualize the guarding by colorizing differently the boundary regions that are guarded by different points. When a region is guarded by more than one point, we simply use the color of the guard with closest Euclidean distance. In the last row, Michelangelo David, Greek, Cyberware male and female models and their guards are shown.

Figure 4 illustrates the star decomposition results on Volumetric Torus-cone, Rocker-arm, Cyberware Male, Female, David and Greek models. Different sub-regions are rendered in different colors, each sub-region is guaranteed to be a star-shape.

Figure 6 illustrates the harmonic parameterization of volumetric objects. The volumetric Torus-Cone and Rocker-Arm models are decomposed into 4 and 7 star-regions, respectively. They are parameterized to a set of solid spheres using the MFS method. (a) shows the parameterization of the rocker-arm. The original model and the sphere parametric domain are shown. The graph encodes the structure and adjacency relationship of different nodes. The color-encoded harmonic field is visualized in a cross-section. On each interior point, such a color-coded level set is transferred from each corresponding point (induced by the computed volumetric map) at the spherical parametric domain as shown in (c). A parameterization of the Torus-Cone model is visualized in (b).

7. CONCLUSION

In this paper we studies the guarding problem for 3D volumetric regions represented by polyhedra meshes. We propose an effective hybrid framework that progressively conduct optimization. Important advantages of our algorithms includes that (1) it effectively breaks down the optimization problem to a solvable size so that guarding can be efficiently computed; and (2) it provides the hierarchical structure and robustness against local geometric perturbations. Therefore, subsequent shape matching or multi-resolutional geometric modeling could potentially benefit from such a hierarchical and robust decomposition. From the guarding results, we compute the star decomposition of the 3D region. The region growing algorithm is conducted on the dual graph of the tetrahedral mesh, using the visibility dependency cost function as the guidance. The region growing has $O(n \log n)$ (coming from the sweep algorithm) complexity for n being the number of tetrahedra in the mesh. Finally, we compute the harmonic spherical parameterization on decomposed star sub-regions, which is shown to have the guaranteed bijectiveness.

Limitations and Future Work. The current decomposition-based parameterization treats each subregion separately and the resultant parametric representation is not smooth along the cutting boundary. Although it is not difficulty to assure C^0 continuity along the boundary by enforcing coherent boundary condition on parameterization, it will be interested to apply stronger boundary constraints in order to obtain higher continuity across inter-subregions. We plan to explore along this direction in the near future.

Also, the effectiveness of our guarding and decomposition computation could rely on the quality of the skeletonization. How exactly low-quality skeletons could affect our computation deserves more study. Also, it might be interesting to see if inversely, checking the intersection of the visibility regions from the boundary vertices can suggest local movement of the guarding point candidates (no longer on the skeleton). A better solution might be provided.

The decomposition framework proposed in this paper is general, and we will also explore more applications for the guarding and star-shape decomposition of 3D regions in our future work.

Acknowledgements

The Michelangelo David and Bunny models are from Stanford Shape Repository. Other models are courtesy of Aim@Shape Repository. The curve-skeleton computation codes are from Tamal Dey. We thank Xianfeng Gu for insightful discussions at the early stage of this project, and we thank Huan Xu and Zhao Yin for their helps in figure generations.

8. REFERENCES

[1] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 3d mesh segmentation

methodologies for cad applications. *Computer-Aided Design*, 4(6):827–841, 2007.

[2] B. Ben-Moshe, M. J. Katz, and J. S. Mitchell. A constant-factor approximation algorithm for optimal terrain guarding. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 515–524, 2005.

[3] B. Chazelle and L. Palios. Decomposition algorithms in geometry. *Algebraic Geometry and Its Applications*, pages 419–447, 1994.

[4] V. Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975.

[5] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 905–914, New York, NY, USA, 2004. ACM.

[6] N. D. Cornea, D. Silver, and P. Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007.

[7] T. Dey and J. Sun. Defining and computing curve-skeletons with medial geodesic function. In *Proc. Eurographics Symp. on Geometry Processing (SGP)*, pages 143–152, 2006.

[8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH*, pages 173–182, 1995.

[9] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[10] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. 2005.

[11] N. Gelfand and L. J. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symposium on Geometry processing*, pages 214–223, 2004.

[12] X. Gu and B. Vemuri. Matching 3D shapes using 2D conformal representations. In *MICCAI (1)*, pages 771–780, 2004.

[13] T. He, L. Hong, D. Chen, and Z. Liang. Reliable path for virtual endoscopy: Ensuring complete examination of human organs. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):333–342, 2001.

[14] Y. He, X. Yin, F. Luo, and X. Gu. Harmonic volumetric parameterization using green's functions on star shapes. *Technical Report Manuscript*, 2008.

[15] H. Hoppe. Progressive meshes. In *SIGGRAPH*, pages 99–108, 1996.

[16] K. Hormann, B. Lévy, and A. Sheffer. Mesh parameterization: Theory and practice. In *ACM Siggraph 2007 Course*, 2007.

[17] D. S. Johnson. Approximation algorithms for combinatorial problems. In *Proc. of the fifth annual ACM Symp. on Theory of computing*, pages 38–49, 1973.

[18] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki. Harmonic coordinates for character articulation. In *SIGGRAPH '07*, pages 71–81, 2007.

[19] T. Ju, S. Schaefer, and J. D. Warren. Mean value coordinates for closed triangular meshes. *SIGGRAPH*, 24(3):561–566, 2005.

[20] D. Kang and J. Ra. A new path planning algorithm for maximizing visibility in computed tomography colonography. 24(8):957–968, August 2005.

[21] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[22] J. M. Keil. Decomposing polygons into simpler components. *PhD Thesis, Dept. Comput. Sci., Univ. Toronto*, 1983.

[23] J. M. Keil. Polygon decomposition. *Handbook of Computational Geometry*, 2000.

[24] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Rapid and effective segmentation of 3d models using random walks. *Comput. Aided Geom. Des.*, 26(6):665–679, 2009.

[25] X. Li, X. Guo, H. Wang, Y. He, X. Gu, and H. Qin. Harmonic volumetric mapping for solid modeling applications. In *Proc. ACM symp. on Solid and physical modeling*, pages 109–120, 2007.

[26] J.-M. Lien. Approximate star-shaped decomposition of point set data. In *Eurographics Symposium on Point-Based Graphics*, 2007.

[27] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polygons. *Computational Geometry*, 35(1-2):100 – 123, 2006. Special Issue on the 20th ACM Symposium on Computational Geometry.

[28] Y. Lipman, D. Levin, and D. Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27(3):1–10, 2008.

[29] A. P. Mangan and R. T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[30] T. Martin, E. Cohen, and R. Kirby. Volumetric parameterization and trivariate b-spline fitting using harmonic functions. In *Proc. ACM Solid and Physical Modeling*, pages 269–280, 2008.

[31] J. W. Milnor. *Morse Theory*. Princeton Univ. Press, 1963.

[32] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. In *Experimental Mathematics*, volume 2, pages 15–36, 1993.

[33] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[34] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[35] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum*, pages 219–228, 2002.

[36] Y. Wang, X. Gu, T. F. Chan, P. M. Thompson, and S. T. Yau. Volumetric harmonic brain mapping. In *IEEE International Symp. on Biomedical Imaging: Macro to Nano.*, pages 1275–1278, 2004.