EFFICIENT 3D REGION GUARDING FOR MULTIMEDIA DATA PROCESSING

Wuyi Yu[*]*, Maoqing Li*

Xiamen University
School of Information Science and Technology
Xiamen, Fujian, China, 361005

S. S. Iyengar, Xin Li [†]

Louisiana State University
Center for Computation and Technology
Baton Rouge, LA, USA, 70803
*E*mail: xinli@lsu.edu

ABSTRACT

With the advance of scanning devices, 3-d geometric models have been captured and widely used in animation, video, interactive virtual environment design nowadays. Their effective analysis, integration, and retrieval are important research topics in multimedia. This paper studies a geometric modeling problem called 3D region guarding. The 3D region guarding is a well known NP-hard problem; we present an efficient hierarchical integer linear programming (HILP) optimization algorithm to solve it on massive data sets. We show the effectiveness of our algorithm and briefly illustrate its applications in multimedia data processing and computer graphics such as shape analysis and retrieval, and morphing animation.

Index Terms— 3D Region Guarding, Shape Analysis, Shape Morphing, Shape Retrieval

1. INTRODUCTION

The rapid advancement of 3D scanning techniques provides 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. 3D geometric data segmentation is a ubiquitous technique. Effective partitioning complex models can benefit many computer graphics applications relying on computationally expensive geometric processing. Besides the improvement of computation efficiency, segmenting each model into a set of salient parts facilitates many shape analysis tasks such as object recognition and retrieval, and benefits animation such as inter-object morphing.

This paper studies the shape partitioning based on the visibility, called *star decomposition*. It segments a 3D volumetric region to a set of sub-regions, each of which is visible from a guarding point (such a subregion is called a star shape). It can be shown that a star shaped subregion has many advantages. For example, in solid modeling, lowly distorted harmonic volumetric parameterization can be constructed bijectively upon such domain [1]. In computer graphics and animation, star decomposition benefits (to be demonstrated in Section 4) tasks such as morphing, shape abstraction and retrieval.

Surface segmentation, generally based on metric measuring local geometric properties of surface patches, has been thoroughly examined (see great survey papers [2, 3]). Partitioning 3D objects based on their volumetric properties, such as convexity, symmetry, etc. have also been studied; however, less study has been conducted to the decomposition based on visibility. Star-decomposition is closely related to a well known *art-gallery guarding* problem. The gallery guarding problem has been studied in computational geometry community on 2D planar domains and in 2.5D for terrain guarding. But "Very little is known about gallery guarding in three dimensions" [4], especially for 3D free-form models, due to their much higher complexity.

To our best knowledge, no effective practical optimization algorithm has been proposed for general 3D models. This paper proposes an effective hierarchical optimization algorithm for the 3D Gallery Guarding problem. The main contributions of this paper are two-folded.

1. We develop an efficient hierarchical optimization algorithm to compute the guarding for a 3D region bounded by a polyhedral mesh. From the guarding we can compute star decomposition of this given model.
2. These optimization algorithmic tools that we have developed can benefit various multimedia application toolkits. Especially, we explore two applications. (a) Interpolation through star decomposition can generate better morphing animation, compared with the conventional linear interpolation approach. (b) Extracted guarding skeletal graph of the given shape encodes encodes the topological and geometric characteristics of the shape, and thus can be used as a signature for shape comparison and retrieval.

[*]This research is done when the first author is a visiting student at Department of Electrical and Computer Engineering, Louisiana State University.

[†]Iyengar is with Department of Computer Science, Li is with Department of Electrical and Computer Engineering, LSU.

978-1-61284-350-6/11/$26.00 ©2011 IEEE

2. BACKGROUND AND RELATED WORK

On a geometric region M, the optimal guarding problem is to find the smallest number of points $\{g_i\}$ (denoted as *guards*) inside M so that every point $p \in \partial M$ is *visible* to at least one guard. Here M is a 3D object whose boundary surface ∂M is represented by a polygonal mesh. For a given guard g_i, a point $p \in M$ is visible to g_i if the line segment $\overline{g_i p}$ entirely locates inside M (considering $\partial M \subset M$). Various versions of this problem is generally called the *art gallery* problem, which is famous and shown to have high complexity. Even in 2D case, the problem is known to be NP-complete. To our best knowledge, no effective approximation algorithm has been proposed for 3D regions bounded by general polygons, and this is the first practical optimization algorithm for guarding large free-form 3D domain represented by polygonal meshes.

The art gallery problem was first proposed by Victor Klee. Guards can be restricted to boundary vertices ($p \in \partial M$), interior vertices ($p \in M$), or mobile vertices (either on boundary or in the interior). When guards are not mobile, they are called *stationary guards*. If guards must be on the boundary, they are called *vertex guards*; if there is no boundary restriction, the guards are referred as *point guards*. In 2D, Chavatal [5] and Fisk [6] both showed that a simple polygon $M \subset \mathbb{R}^2$ needs at most $\lfloor n/3 \rfloor$ stationary guards, based on which, Avis and Toussaint [7] developed an $O(n \log n)$ time algorithm for positioning $\lfloor n/3 \rfloor$ guards in M. When guards are mobile, we call them *mobile guards*. Furthermore, mobile guards are called *edge guards* if they are restricted to boundary vertices.

The above theoretic work discuss the upper bounds for necessary guards on a 2D region. On the other hand, given a specific region, designing practical algorithm to find the actual optimal point guards depends on topology and geometry of this region, and this is usually very difficult. Finding minimal guards has been shown to be NP-hard for 2D polygons with holes [8], 2D simple polygons [9], and even 2D simple orthogonal polygons [10, 11], using either vertex or point guards. Approximation algorithms have been studied in 1.5 ([4]) and 2D ([12]) to get an close-to-optimal result in polynomial time complexity. Lien [13] computes guarding for 3D point cloud data, approximating visibility using ϵ-view. The algorithm is based on a randomized greedy approach.

3. 3D REGION GUARDING

Given a 3-manifold M, whose boundary is discretized by a triangle mesh $\partial M = \{V, F\}$, where $V = \{v_1, v_2, \ldots, v_{N_V}\}$ are vertices and $F = \{f_1, f_2 \ldots, f_{N_F}\}$ are triangle faces connecting them.

- A point $p \in M$ is *visible* to another point $q \in \partial M$ if the line segment $\overline{pq}$ connecting p and q is inside M, namely, it only intersects ∂M on q: $\overline{pq} \bigcap \partial M = \{q\}$.
- The set of all visible vertices $\{q\}, q \in \partial M$ to p is called p's *visible region* $S(p) = \{v | v \in V, v \text{ is visible to } p\}$.
- A set of points $G = \{g_1, \ldots, g_{N_G}\}$ can *visibly guard* the entire region, if the union of their visible regions is the entire ∂M: $\bigcup_{i=1}^{n} S(g_i) = V$; G is called a *guarding set*.
- Finding a guarding set G with the smallest size n is the *optimal gallery guarding* problem that we want to solve.

Our algorithm is based on two intuitions. (1) As demonstrated in several medical visualization and virtual navigation applications (e.g. [14]), medial axes (curve skeletons) usually have desirable visibility to boundary points (referred as the "reliability" of skeletons). An effective skeletons can guide the camera navigation, ensuring nice examination (visibly covered) of the interior of organ surfaces. (2) 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 better numerical efficiency and stability against boundary perturbations.

The medial axis (skeleton) of a given region M can be efficiently computed (see the survey [15]) and represented as a 1D-graph with k sampling nodes. We can then seek an *approximate optimal guarding* by using fewest skeletal nodes to cover the entire ∂M.

In the following, we first introduce the efficient visibility detection via a sweep algorithm. Then we discuss the greedy and optimal guarding strategies, and propose our *Hierarchical Guarding* algorithm.

3.1. Visibility Detection

A basic operation is to detect the visible region of a given point p, namely, to check the visibility of p to each $v_i \in V$. Following the definition, intersection between line segment $\overline{pv_i}$ and ∂M should be checked. v_i is invisible from p if an intersected point q other than v_i has a smaller Euclidean distance $|\overline{pq}| < |\overline{pv_i}|$. Enumerating every $\overline{pv_i}$ and check its intersections with every triangle $\forall f, f \in F$, which takes $O(N_V \cdot N_F) = O(N_V^2)$, on each p, is time consuming. A *sweep algorithm* can be developed to improve the efficiency to $O(N_V \log N_V)$.

First, create a spherical coordinate system originated at p. Each vertex $v_i \in V$ is represented as $\overline{pv_i} = (r(v_i), \theta(v_i), \varphi(v_i))$, where $r(v_i) \geq 0, -\pi < \theta(v_i) \leq \pi, -\frac{\pi}{2} \leq \varphi(v_i) \leq \frac{\pi}{2}$. For every triangle $f_i = (v_{i,1}, v_{i,2}, v_{i,3}) \in F, (1 \leq i \leq N_F)$, its max $\theta(f_i)$ can be defined as $\theta_{max}(f_i) = \max\{\theta(v_{i,j})\}, 1 \leq j \leq 3$, the $\theta_{min}(f_i), \varphi_{max}(f_i), \varphi_{min}(f_i)$ can be defined similarly. The segment $\overline{pv_k}$ cannot intersect with a triangle f unless

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

and therefore we only check triangles (called active triangles) satisfying this condition. The angle functions θ and φ are not continuously defined on a sphere. When a triangle f spans $\theta = \pi$, we duplicate it to ensure that each θ of the original f

is between $(\theta_{min}(f) - 2\pi, \theta_{min}(f)]$ and θ of its duplicate is between $[\theta_{max}(f), \theta_{max}(f)+2\pi)$, by adding or subtracting θ by 2π. For each triangle f spans $\varphi = \pi$, we duplicate it similarly. Then we sort all line segments $\overline{pv_i}$ using $\theta(v_i), \varphi(v_i)$, and sweep all segments in the ascending order of angle functions to check intersections. Given a skeleton point p, for ∂M with N_V vertices it takes $O(N_V \log N_V)$ 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. The total complexity is $O(\log N_V + N_V m)$. The incident triangles around a vertex v_i is generally very small (i.e. $m < \log N_V$). Therefore the algorithm finishes visibility detection of p in $O(N_V \log N_V)$ time. On a skeleton containing k nodes, it takes $O(kN_V \log N_V)$ pre-computation time to compute the visible region for all its nodes.

3.2. Greedy and Optimal Guarding

Once visibility regions for all skeletal nodes are computed, we want to pick a minimum sized point set that can cover all boundary vertices. This reduces to a set-covering problem, also shown to be NP-complete [16]: 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 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 in all of our experiments. If a coarsely sampled skeleton can not cover the entire V, we can include the vertex into the skeleton point set.

A **greedy** strategy for the set covering problem is to iteratively pick the skeletal nodes that covers the largest number of unguarded elements in V, then remove covered ones from V (and update S accordingly since the universe becomes smaller), until $V = \emptyset$. Greedy strategies have been shown effective and it yields $O(\log n)$ approximation [17] to the set covering problem, where n being the number of optimal solution.

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 all element should be visible, for $\forall v_i \in V$ visible to some 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

$$x_i = 0, 1, \text{ and } \sum_{j \in J(i)} x_j \geq 1, \forall m \in \{1, \ldots, n\},$$

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

The above optimization can be solved using branch-and-bound algorithms. When the dimension is small (e.g. a few hundreds to a few thousands), we can use the TomLab Optimization package [18] to solve it efficiently.

3.3. Hierarchical Guarding

The time complexity in ILP optimization is non-polynomial and it limits the size of problems that we can handle. General 3D models in multimedia applications can easily have a number of $20k$ to $200k$ vertices on their boundary surfaces, which prohibitive to ILP optimization. On the other hand, greedy algorithm gets trapped at local minima easily; furthermore, the greedy strategy is sensitive to local geometric perturbations (e.g. a small bump could cause big change in the guarding points selection). We propose a hierarchical guarding computation framework based on the progressive mesh [19], combining ILP and the adaptive greedy refinement.

We simplify the boundary mesh ∂M into several resolutions $\partial M^i = \{V^i, F^i\}, i = 0, \ldots, m$ using progressive mesh [19]. In the coarsest level $i = m$, ILP optimization is performed on all elements $v \in \partial M^m$ and we get the coarsest level guard set $G^i = \{g_k^i\}$. Then we progress to $i = m - 1$ level $\partial M^{m-1} = (V^{m-1}, F^{m-1})$: (a) map existing guards $G^{i+1} = \{g_k\}$ to closest finer-level skeletal nodes $G^i = \{g_k'\}$, locally adjust it to maximize its visible region $S(g_k')$ (b) remove least significant guards $\{g || S(g)| < \epsilon N_V\}$ from G^i; (c) remove covered vertices $\{v | v \in S(g), g \in G^i\}$.

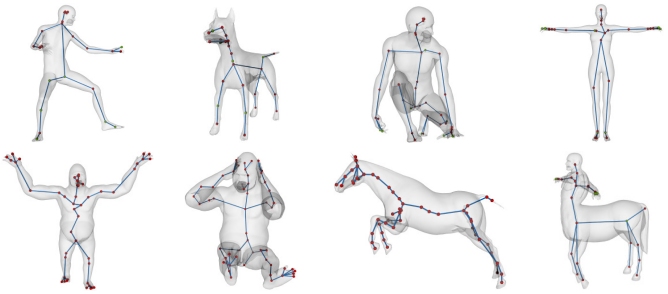


Fig. 1. HILP Guarding of Models from Shape Retrieval Contest (SHREC) Datasets: First row: David, Dog, Michael, and Victoria; Second row: Gorilla 1, Gorilla 2, Horse, and Centaur. Nodes indicae guards.

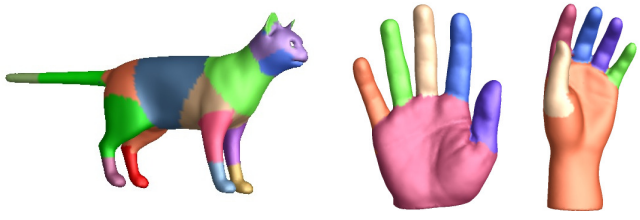


Fig. 2. Decomposition based on Visibility of Models from SHREC: Cat, Hand, and Hand-2. Different subregions are colorized differently.

Table 1. Statistics for the Guarding of Models from Shape Retrieval Contest (SHREC) Datasets. N_V is the number of vertices on the boundary triangle mesh. N_I, N_G, N_H indicate the number of guards necessary computed by ILP, Greedy, and HILP approaches, respectively. t shows the computational time in seconds.

Models	N_V	N_I	N_G	N_H	t_I	t_G	t_H
Cat	10,004	14	19	15	3173	399.4	413.5
Centaur	10,002	–	34	28	–	315.3	336.5
Wolf	10,005	13	18	15	8044	348.4	354.9
David	14,999	–	36	23	–	280.2	287.4
Dog	15,002	–	39	27	–	433.0	444.2
Victoria	15,000	–	35	27	–	425.7	449.3
Horse	20,002	–	38	29	–	386.5	394.6
Michael	20,002	–	45	36	–	333.0	342.9

Then we solve ILP again on uncovered boundary vertices. With details increase in finer levels, new guards will be inserted into G^i. The algorithm ends when all boundary vertices are covered at the finest level $i = 0$. We call this pipeline the hierarchical integer linear programing (HILP). Within this hierarchical pipeline, a set of reduction rules can be further applied on each level to reduce the dimensions of the optimization problem without changing its solution.

In our experiments, we simplify the boundary mesh to the coarsest level with 5k vertices for the first round ILP optimization. On each iteration, we refine to next level with additional 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.

HILP has the following **advantages** over both the pure greedy strategy and the pure ILP:
(1) It is **much faster** than the ILP optimization. HILP can handle massive geometric shapes.
(2) With similar performance, HILP gets **better** guarding solutions than the greedy strategy.
(3) It is **hierarchical** and therefore is **robust** against geometric noise. In HILP, refined local details tend not to significantly change the guarding graph optimized in coarser levels.

We perform our experiments on the standard benchmark: Shape Retrieval Contest (SHREC) 2010 Datasets. Figure. 2,1 illustrate HILP guarding and decomposition on part of these models; Table 1 shows the runtime statistics.

After the guarding graph $G = \{g_i\}$ is computed for a given 3D region M, we can conduct the region growing algorithm seeded from these guards $\{g_i\}$, on the dual graph of the tetrahedral mesh of M. During the region growing we require the subregion to preserve visibility from each seed g_i. With an iterative scheme, the star decomposition of M can be efficiently computed.

4. APPLICATIONS

We propose a shape descriptor based on region guarding. The descriptor has two parts: the guarding graph and histograms

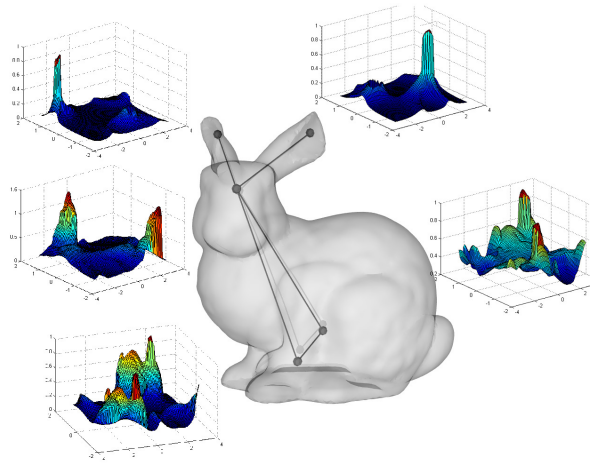


Fig. 3. Guarding Graph Shape Descriptor of the Bunny Model. Each histogram stores distance from each node to the boundary surface.

defined on nodes. The guarding graph is extracted from the skeletal graph, whose nodes are the guarding points. At each guard g_i, we compute a histogram storing the distances from g_i to the object boundary surface, at a set of sampling directions. Fig. 3 illustrates the signature of the Stanford's Bunny model as an example. Note that since the entire region is visible to G, the surface is completely encoded. As a concise and complete signature, this shape descriptor can benefit several applications such as shape analysis and retrieval.

Shape Retrieval. Given a model, we can use this shape descriptor to search for its most similar shapes in the database. Shapes are described and compared by their descriptors. Given two shapes M_1 and M_2, we match their guarding graphs G_1 and G_2 using a weighted energy $E(G_1, G_2)$ composed of three terms ([20]): the node fitting error (histogram similarity), smoothness error (similarity of transformations on adjacent nodes), and length-preserving error (graph edge length changes under transformation). Transformation defined on every node can be solved by minimizing $E(G_1, G_2)$ globally, this minimized energy $E(G_1, G_2)$ characterizes the non-rigid mapping between G_1 and G_2. We then use the symmetric energy $(E(G_1, G_2) + E(G_2, G_1))/2$ as the shape distance between M_1 and M_2. Guarding graphs can be pre-computed (offline) for all shapes in the database; then given a subject shape we match its guarding graph with graphs in the database, and return the one with smallest matching energies. We test this algorithm on a small data set containing seven models: David, Greek, Male, Female, Horse, Camel and Bunny. The shape comparison results are shown in Table 2, from where we finds the following pairs are more similar to each other Male and Female, Horse and Camel, David and Greek. We also perform shape retrieval on the Shape Retrieval Contest (SHREC) 2010 benchmark, please see our accompanying slides for experimental results.

Fig. 4. Consistent Guarding Multiple Models. Left two: two horse models and their extracted skeletal nodes; right two: consistent guarding.

Shape Morphing. Morphing between two surfaces M_1 and M_2 can be generated through *consistent guarding*. The consistent guarding indicates two **isomorphic** graphs G_1 and G_2 (which guard M_1 and M_2 correspondingly). The consistent guarding $\{G_1, G_2\}$ can be computed in two steps: (1) compute one-to-one surface mapping $f_M : M_1 \rightarrow M_2$ using surface mapping techniques (e.g. [21, 22]); (2) extracting compatible skeletons (e.g. [23]) that maps the first curve skeleton C_1 (of M_1) to the second C_2 (of M_2), $f_C : C_1 \rightarrow C_2$; (3) applying HILP simultaneously. We define $v_i \in M_1$ and $f_M(v_i) \in M_2$ are *simultaneously visible* to $p_j \in C_1$ and $f_C(p_j) \in C_2$, if both v_i is visible to p_j and $f_1(v_i)$ is visible to $f_2(p_j)$. Then we can get consistent guarding $\{G_1, G_2\}$ for $\{M_1, M_2\}$, G_i may require more guards than necessary to cover M_i, but the guarding points and their images consistently cover both models. This consistent guarding can generate consistent star decomposition which can benefit many applications. An example is the morphing animation. Conventionally, shape morphing can be generated by linear interpolation: given inter-surface mapping $f_M : M_1 \rightarrow M_2$, the morphing for each vertex is generated by linear interpolation between $v_1 \in M_1$ and its image $f_M(v_1) \in M_2$: $v(t) = (1-t)v_1 + tv_2$. With star decomposition, we can interpolate the spherical coordinates to generate the morphing. A similar idea in 2D, based on star decomposition for 2D polygons, is introduced in [24]. Now on each star region, the transformation with respect to the center is decomposed into rotation and scaling (along the radial direction), and interpolated separately. Compared with direct linear interpolation, this new morphing scheme can cause less self-intersection, therefore generating more natural interpolation. A comparative example is shown in Figure 5.

5. CONCLUSION

We propose a 3D guarding algorithm that can cover a given complicated region using as few as possible points. To our best knowledge, the proposed hierarchical integer linear programming (HILP) algorithm is a first efficient optimization algorithm for finding good approximate solutions to this NP-hard problem. We demonstrate the effectiveness of this algorithm on two graphics applications: morphing animation and shape retrieval. In the near future, we will explore more applications of 3D region guarding. On example is in robotic environment inspection [25].

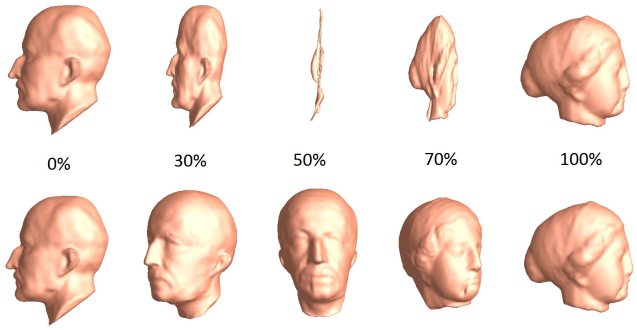


Fig. 5. Spatial Linear Interpolation (upper row) VS Spherical Interpolation on Star Region (lower row). Each row, from left to right: original shape, 30%-interpolation, 50%-interpolation, 70%-interpolation, and target shape. Spatial linear interpolation may causes unnecessary collapsing-and-expanding; spherical interpolation on star region leads to more visually-natural interpolation.

Acknowledgements

This work has been supported by Louisiana Board of Regents Research Competitiveness Subprogram (RCS) LEQSF(2009-12)-RD-A-06, PFund: NSF(2009)-PFund-133, LSU Faculty Research Grant 2010, NSF CNS-0963793. We would like to thank Tamal Dey for their software, and AIM@Shape, SHREC, and Stanford Shape Repository for their datasets.

6. REFERENCES

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

[2] A. Shamir, "A survey on mesh segmentation techniques," *Computer Graphics Forum*, vol. 27, no. 6, pp. 1539–1556, 2008.

[3] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis, "3d mesh segmentation methodologies for cad applications," *Computer-Aided Design*, vol. 4, no. 6, pp. 827–841, 2007.

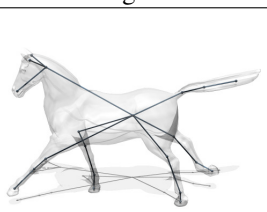
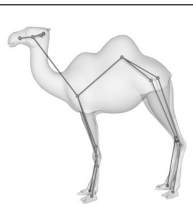
[4] B. Ben-Moshe, M. J. Katz, and J. S.B. Mitchell, "A constant-factor approximation algorithm for optimal terrain guarding," in *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 515–524.

[5] V. Chvatal, "A combinatorial theorem in plane geometry," *Journal of Combinatorial Theory*, pp. 39–41, 1975.

[6] S. Fisk, "A short proof of chvatal's watchman theorem," *Journal of Combinatorial Theory, Series B*, vol. 24, no. 3, pp. 374 – 374, 1978.

[7] D. Avis and G. Toussaint, "An efficient algorithm for decomposing a polygon into star-shaped polygons," *Pattern Recognition*, vol. 13, pp. 395–398, 1981.

Table 2. Guarding Skeleton Registration Error

							
1	0	0.8945	0.8145	0.99	1.115	1.558	2.902
2	0.8945	0	0.905	0.7445	1.1825	1.5445	2.509
3	0.8145	0.905	0	0.6695	1.526	1.3315	5.702
4	0.99	0.7445	0.6695	0	1.46	1.26	6.0925
5	1.115	1.1825	1.526	1.46	0	0.806	3.46
6	1.558	1.5445	1.3315	1.26	0.806	0	4.102
7	2.902	2.509	5.702	6.0925	3.46	4.102	0

[8] J. O'Rourke and K.J. Supowit, "Some np-hard polygon decomposition problems," vol. 29, pp. 181–190, 1983.

[9] D T Lee and A K Lin, "Computational complexity of art gallery problems," *IEEE Trans. Inf. Theor.*, vol. 32, no. 2, pp. 276–282, 1986.

[10] D. Schuchardt and H. Hecker, "Two np-hard art-gallery problems for ortho-polygons," *Math. Log. Q.*, vol. 41, pp. 261–267, 1995.

[11] M. Katz and G. Roisman, "On guarding the vertices of rectilinear domains," *Comput. Geom. Theory Appl.*, vol. 39, no. 3, pp. 219–228, 2008.

[12] A. Efrat and S. Har-Peled, "Guarding galleries and terrains," *Inf. Process. Lett.*, vol. 100, no. 6, pp. 238–245, 2006.

[13] J. Lien, "Approximate star-shaped decomposition of point set data," in *Eurographics Symposium on Point-Based Graphics*, 2007.

[14] T. He, L. Hong, D. Chen, and Z. Liang, "Reliable path for virtual endoscopy: Ensuring complete examination of human organs," *IEEE Trans. on Visualization and Computer Graphics*, vol. 7, no. 4, pp. 333–342, 2001.

[15] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Trans. on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.

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

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

[18] "TOMLAB v3.0 User's Guide," Technical Report IMaTOM-2001-01, 2001.

[19] H. Hoppe, "Progressive meshes," in *SIGGRAPH*, 1996, pp. 99–108.

[20] L. Wei, W. Yu, M. Li, and Xin Li, "A non-rigid registration algorithm for compatible skeletonization," in *Proc. International Conference on Computer Science and Education*, 2010, pp. 209 – 214.

[21] X. Li, X. Gu, and H. Qin, "Surface mapping using consistent pants decomposition," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 4, pp. 558–571, 2009.

[22] X. Li, Y. Bao, X. Guo, M. Jin, X. Gu, and H. Qin, "Globally optimal surface mapping for surfaces with arbitrary topology," *IEEE Trans. on Visualization and Computer Graphics*, vol. 14, no. 4, pp. 805–819, 2008.

[23] Q. Zheng, A. Sharf, A. Tagliasacchi, B. Chen, H. Zhang, A. Sheffer, and D. Cohen-Or, "Consensus skeleton for non-rigid space-time registration," *Computer Graphcis Forum (Special Issue of Eurographics)*, vol. 29, no. 2, pp. 635–644, 2010.

[24] M. Shapira, "Shape blending using the star-skeleton representation," *IEEE Computer Graphics and Applications*, vol. 15, pp. 44–50, 1995.

[25] X. Li, W. Yu, X. Lin, and S. S. Iyengar, "An optimal region guarding algorithm for autonomous pipeline inspection," *IEEE Trans. on Robotics*, p. to appear, 2011.