Computing 3D Shape Guarding and Star Decomposition

Wuyi Yu¹,²  Xin Li²*

¹Xiamen University, China
²Louisiana State University, USA (Email: xinli@cct.lsu.edu)
Star Decomposition and Guarding

- **Shape Decomposition**
  - shape decomposition is a ubiquitous geometric processing tool
  - complex models can be partitioned into solvable subdomains for effective computation and processing
  - Convex decomposition $\rightarrow$ partition shapes into convex subparts

- **Star Decomposition**
  - to partition a 3D model to a set of *star-shaped sub-regions*
  - a **star shape** is a region that is visible from an interior point
  - Less strict than convex decomposition, fewer parts
  - e.g. bijective harmonic volumetric parameterization can be constructed on star-shaped domain [Xia et al. 2010]

- **Computation of Star Decomposition**
  - closely related to the visibility guarding problem.
Art Gallery Problem: Guarding in 2D

Art Gallery Problem:
To guard an art gallery with minimum number of cameras:

- the layout of the art gallery → a region $M$ bounded by a polygon $\partial M$;
- a guard (ceiling camera) → a point $p \in M$;
- A point $p \in M$ is visible to another point $q \in M$, if the line segment $pq \subseteq M$ (assuming $\partial M \subset M$).
- A point set $G$ can guard $\partial M$, if $\forall q \in \partial M$, $\exists p \in G$ s.t. $q$ is visible to $p$.
- Optimal Gallery Guarding : to find a $G$ with minimal points that guards $\partial M$. 

\[ \text{Art Gallery Problem: } \]
Theoretic Bounds for Guarding 2D Polygons

- simple polygon: $\leq \left\lfloor \frac{n}{3} \right\rfloor$ guards [Chavatar 1975]
- simple orthogonal polygon: $\leq \left\lfloor \frac{n}{4} \right\rfloor$ guards [Kahn et al 1983]
- polygon with h holes: $\leq \left\lfloor \frac{n+h}{3} \right\rfloor$ guards [Hoffmann et al 1991]
- orthogonal polygon with h holes: $\leq \left\lfloor \frac{3n+4h+4}{16} \right\rfloor$ guards [Gyori et al 1996]

Computing optimal guarding of a given polygon:

- optimal guarding for 2D polygons with holes is NP-hard [O’Rourke and Supowit 1983]
- optimal guarding for 2D simple polygons is NP-hard [Lee and Lin 1986]
- optimal guarding for 2D simple orthogonal polygons is NP-hard [Schuchardt and Hecker 1995], [Katz and Roisman 2008]
- approximation algorithm in 1.5D region [Ben-Moshe et al 2005]
- approximation algorithm in 2D region [Efrat and Har-Peled 2006]
- “Very little is known about gallery guarding in 3-dimensions” ([Ben-Moshe et al 2005])
Related Work: Shape Decomposition

Mesh Segmentation:
- Extensively studied, two surveys: [Agathos et al 2007] [Shamir 2008]
  - Region Growing Algorithms ([Lai et al 2009], [Cohen-Steiner et al 2004])
  - Watershed Algorithms ([Mangan and Whitaker 1999])
  - Clustering-based Algorithms ([Shlafman et al 2002], [Gelfand and Guibas 2004])

3D Shape Decomposition
- 2D Polygon Star Decomposition for Morphing [Shapira and Rappoport 95]
- Some 2D methods on polygon decomposition are generalizable [Chazelle and Palios 94] [Keil 2000]
- Approximate Convex Decomposition [Lien and Amato 2006]
- Approximate Star Decomposition of Point Cloud Surfaces [Lien 2007]
Our Idea: Solving Guarding on Skeleton

Problem: 3D Region Guarding:
Given a 3D region $M \subset R^3$, represented by a tetrahedral mesh \( \{V_M, T_M\} \), to find a minimal point set $G$ inside $M$ that guards \( \partial_M = \{V_{\partial M}, F_{\partial M}\} \).

Intuition:
- Medial axis (curve skeleton) $S(M)$ of $M$ has great visibility to $\partial M$
- Densely sampled curve skeleton nodes \( \{p_i\} \subset S(M) \Rightarrow \) Candidates to guard $M$
Visible Region Detection

- Visible Region (of a skeletal node $p \in S(M)$) on $\partial M$
  - A vertex $v_i \in V$ is visible to $p$ if $p v_i$ doesn’t intersect with any other face $f_j \in \partial M$
  - A face $f_k \in F$ is visible to $p$ if all its three vertices are visible to $p$
  - The visible region of $p$ contains all its visible faces $V(p) = \{ f_k | f_k \text{ is visible to } p \}$

- Efficient Detection of Visible Region of $p$
  - Direct Detection: $O(n^2)$: checking all $p v_i$ and $f_j$
  - We propose a sweep algorithm: $O(n \log n)$.

Algorithm: sweeping a straight line on a point $p$:

1) Sort all the vertices $v_i \in \partial M$ according to the spherical coordinates with respect to $p$: $p v_i = (r(v_i), \theta(v_i), \phi(v_i))$;
2) Sort all the faces $f_i \in \partial M$ using the max and min of $(\theta, \phi)$ of each triangle’s vertices;
3) Sweep the line $pv_i$ and only check its intersection within its nearby triangles $\{f_j\}$. 
Converting Guarding to Set-Covering and a Greedy Solving Approach

- Guarding $\rightarrow$ Set-Covering
  - Each skeleton point $p_i$ is associated with its visible region on $\partial M$
  - Guarding becomes the following set-covering problem

- Set-Covering
  - Given a universe $U = \{u_1, \ldots, u_n\}$ and sets $S_1, \ldots, S_m \subseteq U$, a cover is a collection $G$ of some of the sets from $\{S_k\}$ whose union is $U: \bigcup_{S_k \in G} S_k = U$
  - To find the optimal covering, i.e. the cover $G$ contains the fewest necessary subsets
  - Finding its optimal solution is still NP-complete [Kahn et al 1983]

- A Greedy Approach to Solve the above Set-Covering Problem
  - Iteratively select a subset $S_i$ that contains the most uncovered points in $U$
  - Repeat until all points in $U$ are covered;
  - Efficacy: produce a good approximation; the result is at most $\log C$ times worse than the optimal solution, $C = |G_{opt}|$.
An optimal result can be found by the following **Integer Linear Programming (ILP)**:

Assign a variable $x_j$ to each skeleton point $p_j$ (i.e. $S_j$ correspondingly):

- The **objective function** to minimize: $\sum_{j=1}^{m} x_j$.
- All boundary faces should be covered: $\forall f_i \in F_{\partial M}$, at least one skeletal node visible to $f_i$ should be selected;
- Let $J(i)$ be the index set of all nodes $p_j$ visible to $f_i$: $J(i) = \{ j | p_j \text{ is visible to } f_i \}$.
- Therefore, the **constraints** are: $x_j = \{ 0, 1 \}, (j = 1, \ldots, m)$, and,
  $$\sum_{k \in J(i)} x_k \geq 1, (i = 1, \ldots, n).$$
- This optimization can be solved using a branch-and-bound algorithm, if the problem size is small.
Solving ILP is too expensive for general 3D models. We propose a **Progressive Integer Linear Programming (PILP)** Algorithm:

- Progressively simplify $\partial M$ into multiple resolutions;
- Compute guarding using ILP in the coarsest level;
- Iteratively progress to finer levels,
  1) remove some least significant guards
  2) apply reductions;
  3) then solve ILP again on this level.
Reduction Rules

- Reducing the problem size, without changing the optimality of the solution
- Consider the visibility matrix $A_{|S| \times |F|}$:
  
  \[
  (a_{ij} = 1, \text{ if } i\text{-th skeletal point } p_i \text{ is visible to } j\text{-th face element } f_j)
  \]

1) If for row $i_1$ and $i_2$: $a_{i_1,j} = 1 \rightarrow a_{i_2,j} = 1$, then remove row $i_1$;
2) If for column $j_1$ and $j_2$: $a_{i,j_1} = 1 \rightarrow a_{i,j_2} = 1$, then remove row $j_2$;
3) If $a_{ij} = 1$ and column $j$ has only this non-zero element, then (a) add $p_i$ to $G$, (b) remove column $j$, (c) for $\forall a_{ik} = 1$ remove column $k$;
4) If the matrix is composed of several blocks, we partition them and solve them individually.

\[\begin{array}{c}
\includegraphics[width=0.2\textwidth]{image1}
+ \includegraphics[width=0.2\textwidth]{image2}
= \includegraphics[width=0.2\textwidth]{image3}
= \includegraphics[width=0.2\textwidth]{image4}
\end{array}\]
2) If for column $j_1$ and $j_2$: $a_{i,j_1} = 1 \rightarrow a_{i,j_2} = 1$, then remove row $j_2$;
3) If $a_{ij} = 1$ and column $j$ has only this non-zero element, then (a) add $p_i$ to $G$, (b) remove column $j$, (c) for $\forall a_{ik} = 1$ remove column $k$;
4) If the matrix is composed of several blocks, we partition $A$ to several smaller ones and solve them individually.
Guarding 48 models from the TOSCA dataset [TOSCA]
Experimental Results (cont.)
A region $M$ is a *star shape* if $\exists p \in M$ such that $\forall q \in M$, $q$ is visible to $p$.

Given a region $M$, a decomposition $d(M)$ of $M$ is defined as

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

where $M^\circ = M \setminus \partial M$.

A star decomposition of $M$: each $M_i$ is a star shape.
Region Growing for Decomposition

With guarding computed, star decomposition can be computed by region growing:

1) Compute the dual graph of the tetrahedral mesh;
2) Use guards as seeds (each → a star subregion);
3) Start from all seeds simultaneously, and grow the regions (each region keeps merging adjacent unclassified nodes on the dual graph, while preserving its star property).

 Efficiency: visibility pre-computed in $O(n \log n)$, region growing in $O(n)$
Experimental Results (decomposition)
$N_V$ is the boundary surface mesh vertex number. $N_I, N_G, N_P$ indicate the number of necessary guards computed by ILP, Greedy, and PILP approaches, respectively. $t$ shows the computational time in seconds. Guarding of big models cannot be solved directly using ILP, so their statistics are not applicable.

### Experimental Results (Statistic Table)

<table>
<thead>
<tr>
<th>Models ($N_V$)</th>
<th>$N_I$</th>
<th>$N_G$</th>
<th>$N_P$</th>
<th>$t_I$</th>
<th>$t_G$</th>
<th>$t_P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greek (9,994)</td>
<td>15</td>
<td>22</td>
<td>18</td>
<td>4,122.4</td>
<td>290.2</td>
<td>293.1</td>
</tr>
<tr>
<td>David (9,996)</td>
<td>16</td>
<td>22</td>
<td>17</td>
<td>107,391.2</td>
<td>233.9</td>
<td>235.2</td>
</tr>
<tr>
<td>Female (10,002)</td>
<td>13</td>
<td>18</td>
<td>14</td>
<td>2,046.2</td>
<td>264.2</td>
<td>281.1</td>
</tr>
<tr>
<td>Male (10,002)</td>
<td>14</td>
<td>16</td>
<td>15</td>
<td>3,074.3</td>
<td>298.6</td>
<td>310.2</td>
</tr>
<tr>
<td>Cat (10,004)</td>
<td>14</td>
<td>19</td>
<td>15</td>
<td>3173</td>
<td>375.4</td>
<td>393.1</td>
</tr>
<tr>
<td>Wolf (10,005)</td>
<td>13</td>
<td>18</td>
<td>15</td>
<td>8044</td>
<td>328.1</td>
<td>349.9</td>
</tr>
<tr>
<td>Dog (15,002)</td>
<td>–</td>
<td>39</td>
<td>27</td>
<td>–</td>
<td>412.3</td>
<td>433.2</td>
</tr>
<tr>
<td>Victoria (15,000)</td>
<td>–</td>
<td>35</td>
<td>27</td>
<td>–</td>
<td>408.7</td>
<td>421.2</td>
</tr>
<tr>
<td>Horse (20,002)</td>
<td>–</td>
<td>38</td>
<td>29</td>
<td>–</td>
<td>376.1</td>
<td>384.2</td>
</tr>
<tr>
<td>Michael (20,002)</td>
<td>–</td>
<td>46</td>
<td>31</td>
<td>–</td>
<td>321.0</td>
<td>332.9</td>
</tr>
<tr>
<td>Gorilla (30,004)</td>
<td>–</td>
<td>60</td>
<td>46</td>
<td>–</td>
<td>462.4</td>
<td>490.1</td>
</tr>
<tr>
<td>Centaur (30,002)</td>
<td>–</td>
<td>52</td>
<td>32</td>
<td>–</td>
<td>488.1</td>
<td>514.5</td>
</tr>
</tbody>
</table>
Greedy vs PILP on 48 models in TOSCA dataset. The x-axis lists the 48 models, the y axis indicates the necessary guards computed. The blue bar indicates the PILP result and red is the greedy result. PILP has similar computational performance with the greedy approach, but generate better guarding: on average, PILP uses 20% less guards.
Experimental Results (Hierarchical Guarding)

Greedy (left, 49 guards) and PILP (right, 39 guards) Guarding of Centaur.
Applications

- Region guarding and star decomposition are ubiquitous in geometric computing and processing

- We explore two applications in computer graphics:
  - Geometric signature for shape retrieval
  - Morphing
Shape Descriptor Defined on Guarding Graph

Shape Descriptor ➔ the guarding skeleton + histogram on each node

Histogram Computation:
On each guard $p$, shoot rays $\{r_i\}$ towards all spatial directions defined on a unit sphere. Each ray $r_i$ intersects with $\partial M$ on a point $q_i$. The lengths $\{|pq_i|\}$ compose the histogram $H_p$

This shape descriptor is
- Complete: can reconstruct $M$
- Efficient: guarding skeleton is simplest
- Discriminative: local geometric deviation can be detected

Applications: Shape Retrieval
Shape comparison of 48 models in the TOSCA dataset.

- **Black** indicates the smaller difference.
- **Black blocks** indicate similar groups of models.
Morphing

Morphing by Linear Interpolation: $M(0) = M_1, M(1) = M_2 \Rightarrow M(t), 0 < t < 1$

1) Compute bijective surface mapping $f : M_1 \rightarrow M_2$;
2) For $\forall p \in M_1$, $p(t) = (1 - t)p + tf(p)$.

In 2D: Star Decomposition can be used to prevent self-intersection in Morphing

- Proposed Shape Interpolation based on Star Decomposition
  1. Compute bijective surface mapping $f : M_1 \rightarrow M_2$;
     [Li et al. 2008]
  2. Compute compatible skeletonization of $M_1$ and $M_2$;
     [Zheng et al. EG10]
Morphing

Morphing by Linear Interpolation: \( M(0) = M_1, M(1) = M_2 \Rightarrow M(t), 0 < t < 1 \)

1) Compute bijective surface mapping \( f : M_1 \rightarrow M_2 \);

2) For \( \forall p \in M_1, p(t) = (1 - t)p + t f(p) \).

In 2D: Star Decomposition can be used to prevent self-intersection in Morphing

\[\text{[Shapira and Rappoport 95]}\]

➢ Proposed Shape Interpolation based on Star Decomposition

1. Compute bijective surface mapping \( f : M_1 \rightarrow M_2 \);

   \[\text{[Li et al. 2008]}\]

2. Compute compatible skeletonization of \( M_1 \) and \( M_2 \);

   \[\text{[Zheng et al. EG10]}\]

Compatible Skeletons
Morphing by Linear Interpolation: \( M(0) = M_1, M(1) = M_2 \Rightarrow M(t), 0 < t < 1 \)

1) Compute bijective surface mapping \( f : M_1 \rightarrow M_2 \);
2) For \( \forall p \in M_1, p(t) = (1 - t)p + tf(p) \).

- Proposed Shape Interpolation based on Star Decomposition

... 

3. Compute compatible guarding and star decomposition on \( M_1 \) and \( M_2 \);
4. Solve as-rigid-as-possible interpolation on each subdomain and blend the adjacent regions (generalized from [Baxter et al 2008])

Compatible Guarding
Interpolating the Galloping Horse

Applications: Shape Morphing

- Linear Interpolation
- Center-Driven As-Rigid-As-Possible Interpolation
- As-Rigid-As-Possible Interpolation Based on Region Guarding and Decomposition
Conclusions

Contributions:
- An effective progressive integer linear programming (PILP) optimization paradigm to compute approximate optimal guarding of 3D free-form domains;
- An efficient region-growing algorithm to compute the star-decomposition of 3D models;
- Two direct applications of guarding/star-decomposition: shape matching and shape morphing.

Future Work:
- Optimize skeleton computation and skeletal nodes sampling;
- Improve our shape matching algorithm;
- Explore new applications of guarding and star decomposition, e.g. in robotics.
Acknowledgements

- Anonymous reviewers;
- Data sources: TOSCA Dataset, Aim@Shape, Robert W. Sumner and Jovan Popović, and Stanford Shape Repository
- Funding Agencies:
  - Louisiana Board of Regents Research (LA BOR) Competitiveness Subprogram (RCS) LEQSF(2009-12)-RD-A-06
  - LA BOR PFund: NSF(2011)-PFund-236
  - LSU Faculty Research Grant 2010
Thank you!

Project Webpage (with the paper, slides, video, and executable demos):

http://www.ece.lsu.edu/xinli/Research/Guarding.html
Guarding by vertices VS by skeletal nodes

| $|V_{\partial M}| / |V_M|$ : # of boundary vertices / tetrahedral vertices, respectively; |
|--------------------------|
| $|S|$ : # of nodes on extracted skeletons; |
| $|G_{V_M}| / |G_S|$ : # of necessary guards (solved by ILP) when candidates are all tet-vertices / skeletal nodes, respectively.

| Models  | $|V_{\partial M}|$ | $|V_M|$ | $|S|$ | $|G_{V_M}|$ | $|G_S|$ |
|---------|----------------|--------|------|----------|------|
| Kitten  | 400            | 1,682  | 122  | 3        | 3    |
| Beethoven | 502          | 2,895  | 88   | 2        | 2    |
| Bimba   | 752            | 5,115  | 139  | 2        | 2    |
| Buddha  | 502            | 3,002  | 155  | 2        | 2    |
| Bunny   | 998            | 8,320  | 270  | 5        | 5    |