Computing Optimal Guarding and Star Decomposition of 3D Models

Wuyi Yu¹  Xin Li²*
¹Xiamen University {yuwuyi@gmail.com}  ²Louisiana State University {xinli@cct.lsu.edu}

**INTRODUCTION**

**Optimal Guarding:** Given a 3D region $M$, to find a smallest set of points $\{g_i\} \in M$ to which the entire region $\forall p \in M$ is visible.

**Star Decomposition:** To partition a 3D region to a set of star subregions (each star shape is visible from an interior point).

**Previous Work**

- Finding optimal guarding for a given 2D region is NP-hard [Chv75].
- Lien [Lie07] presents an approximate guarding for point set data.
- To our best knowledge, no effective guarding/star decomposition computation algorithm has been developed for general 3D regions represented by polyhedra.

**METHODS**

**Overview**

- Intuition: To seek optimal guarding of 3D shapes on their medial axes (skeletons) due to their nice visibility.
- Region Guarding: An efficient multi-level integer linear programming optimization framework.
- Star Decomposition: A constrained region growing algorithm, seeded from computed optimal guarding points.

**Algorithms**

1. **Visibility Region Detection.** Given a model $M$ with boundary $\partial M$ and a node $p$ on the skeleton $S(M)$, to detect $p$’s visibility region $\forall (p) = \{f_i\}_{i \in \partial M}$: an efficient $O(n \log n)$ sweep algorithm.

2. **Converting Guarding to Set Covering.** Assign a variable to each skeleton node, the guarding problem can be converted to a Set Covering Problem: to pick the fewest nodes whose union of visibility regions is the entire $\partial M$.

- **An Optimal ILP algorithm.** Set Covering can be solved by Integer Linear Programming (ILP): optimal solution, but exponential complexity.

- **A Greedy Algorithm.** Iteratively pick the skeletal node that can see most uncovered faces and remove the covered faces from the universe, until all faces are covered: good efficiency, but approximate solution.

- **PILP.** We propose an efficient optimization algorithm: Progressive Integer Linear Programming (PILP)

3. **Star Decomposition.** Using guarding points as seeds, to perform region growing.

**Progressive Integer Linear Programming Pipeline**

1. Progressively simplify $\partial M$ into multiple resolutions;
2. Compute guarding using ILP in the coarsest level;
3. Iteratively progress to finer levels, on each level:
   - (a) map existing guards to the finer-level skeleton, remove some least significant guards, then remove covered faces,
   - (b) solve ILP again.

**EXPERIMENTAL RESULTS**

**PILP V.S. Greedy and ILP**

<table>
<thead>
<tr>
<th>Models</th>
<th>$n_t$</th>
<th>$n_f$</th>
<th>$t_{pred}$</th>
<th>$t_{greedy}$</th>
<th>$t_{ilp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eagle (10K)</td>
<td>15</td>
<td>20</td>
<td>121.6</td>
<td>280.0</td>
<td>297.9</td>
</tr>
<tr>
<td>Female (10K)</td>
<td>15</td>
<td>20</td>
<td>258.9</td>
<td>318.3</td>
<td>318.4</td>
</tr>
<tr>
<td>Male (10K)</td>
<td>14</td>
<td>19</td>
<td>323.7</td>
<td>374.3</td>
<td>374.3</td>
</tr>
<tr>
<td>Cat (10K)</td>
<td>15</td>
<td>20</td>
<td>429.7</td>
<td>467.3</td>
<td>467.3</td>
</tr>
<tr>
<td>Rhino (10K)</td>
<td>15</td>
<td>20</td>
<td>800.4</td>
<td>920.4</td>
<td>920.4</td>
</tr>
<tr>
<td>Dog (15K)</td>
<td>20</td>
<td></td>
<td>122.3</td>
<td>145.5</td>
<td>145.5</td>
</tr>
<tr>
<td>Vase (15K)</td>
<td>25</td>
<td></td>
<td>186.7</td>
<td>212.4</td>
<td>212.4</td>
</tr>
<tr>
<td>Horse (20K)</td>
<td>20</td>
<td></td>
<td>301.1</td>
<td>334.2</td>
<td>334.2</td>
</tr>
<tr>
<td>Motorcycle (25K)</td>
<td>50</td>
<td></td>
<td>152.9</td>
<td>172.8</td>
<td>172.8</td>
</tr>
<tr>
<td>Cat (50K)</td>
<td>50</td>
<td></td>
<td>488.4</td>
<td>514.5</td>
<td>514.5</td>
</tr>
</tbody>
</table>

- PILP is much faster than ILP: several magnitude of efficiency improved on large geometric models, with similar guarding size.
- PILP obtains significantly better guarding results than Greedy approach, with comparable efficiency.

**Guarding and star-decomposition of models in TOSCA dataset and Aim@shape dataset**

**APPLICATIONS**

**Shape Retrieval**

**Shape Interpolation**

**Shape Descriptor = Guarding Skeleton + Distance Histogram**

Shape Retrieval using this shape descriptor on 18 models from TOSCA dataset, based on graph matching; Black indicates better similarity.

**Autonomous Robot Planning for Pipeline Inspection**

Optimal region guarding implies the ideal spots for the autonomous robot inspection, helps planning for online detection of abnormal geometric changes of the environment.

**REFERENCE**
