Geometric Covering

Nadav Shragai
CGGC, CS, Technion, Israel

MSc Thesis
Introduction

Geometric Covering (GC) queries appear in numerous applications:

- Mold design in manufacturing
- Inspection
- Security and surveillance
- Placements of cellular antennas
- Illumination design
- Spraying of paint
Layout of the Rest of the Talk

We are focusing on mold-design and security.

- Related work in mold-design and security.
- A generic unified framework for answering geometric covering.
- **Geometric Covering** is an *NP*-hard problem.
- Examples of the generic framework as implemented in a 3D mold-design and security.
- Conclusions and future work.
Related Work I

Mold design

- 2-pieces-mold polygonal decomposition in $R^3$ [Ahn02, Khardekar06, Chen06]
- $n$-pieces-mold polygonal decomposition in $R^3$ [Liu09, Priyadarshi04, Stoyan10]
- 2-pieces-mold freeform surface decomposition in $R^3$ [Elber04]
- Algebraic analysis of visibility of freeforms in $R^3$ [Seong06]
- **Nothing** so far on automatic $n$-pieces-mold freeform decomposition in $R^3$
Related Work II

Security

- Polygonal 2.5D terrain where \( z = f(x, y) \).
- Guards on the vertices or above them [Lee91, Goodchild89]
- Edge guards [Bose96, Bose97]
- Different greedy solutions [Goodchild89, Kaucic04]
- Guards limited to strategic locations [Kim04]
- Calculating partial visibility [Franklin94, Rana03]
Set-Cover I

- Set-cover (SC) is a classic computer science query.

- SC is considered a very hard problem to solve ($NP$ hard).

- Given some universe $U$ and a family $F$ of subsets of $U$ which their union equals $U$, a cover of $U$ is a subfamily of $F$ whose union still equals $U$.

- In SC we are seeking a cover with minimal number of subsets.
Set-Cover II

- The universe $U$ is a set of circles.
- A subset of $U$ is a group of circles.
- The family $F$ is all these groups of circles.
- The subfamily $F_1$ is the brown, yellow, blue and green groups.
  $F_1$ is a cover of $U$.
- The subfamily $F_2$ is the red, purple and yellow group.
  $F_1$ is a minimal cover of $U$.

We will now show a reduction from GC problems to SC problems.
Visibility Map I

- We receive a 2-manifold geometry in $\mathbb{R}^3$, $C$, which has a parameterization $x_{uv}, y_{uv}, z_{uv}$.
- The domain $D_C$ of $C$ is a 2-dimensional box, a rectangle, possibly trimmed.
- We are creating a discrete representation of $D_C$ as an image, as a visibility map.
- The visibility map can serve as a controlled approximation for the coverage of $C$. 
Visibility Map II

The Utah Teapot with its interior curved in.

- Visible locations are set to **white**.
- Hidden locations are set to **black**.
- Trimmed away bits are set to **green** - don’t care.

Visibility Map II
A General Framework for Geometric Coverage Analysis

Center for Graphics and Geometric Computing, Technion
Linearize the visibility map, as a vector of bits as follow:

- Don’t care locations are simply skipped.
- Each bit is either 1 (visible pixel) or 0 (hidden pixel).
- Sequence the 1/0 bits in some order over the visibility map (for example: left to right, top to bottom).
Visibility Map IV

Visibility map of 8×7

Vector of 56 bits
Set-cover can be clearly applied to vectors of bits:

- The universe $U$ is the domain $D_C$.
- A subset of $U$ is a vector of bits.
- A family $F$ of subsets of $U$ is a set of vectors of bits from different views around the geometry $C$.
- A cover of $U$ is a subfamily of $F$, a set of vectors of bits which their union equals $D_C$. 
The set-cover is done in the parametric domain.

Subfamily of the set of visibility maps

The union of the visibility maps
Creating Visibility Maps I

- Input geometry $C$ can be a surface or a set of surfaces, possibly trimmed.

- Each surface has its own rectangular domain, created independently of the other surfaces.

- We rearrange the domains of all the surfaces in one large image: The visibility map of $C$. 
Creating Visibility Maps II
Given \( C \) and \( D_C \), the visibility map from direction \( V_i \) is computed as follow:

- The surface is tessellated into triangles.
- Two-rendering passes:
  - I. A regular (Z-buffer) rendering of \( C \) from \( V_i \) keeping only the Z-depth information, in \( ZBuffer(x, y) \).
  - II. Scan conversion of \( C \) in the domain, \( D_C \), and deciding visibility by comparing the Z-depths.
Creating Visibility Maps IV

Pass II

A tessellation $T = \{T_i\}$ of triangles with UV parametric coordinates is given.

For each triangle $T_i$ in $T$, scan convert $T_i$ by its UV coordinates.

For each pixel $p_{uv}$ in $T_i$

\[
 x_{uv}, \ y_{uv}, \ z_{uv} \leftarrow \text{XYZ coordinates of } p_{uv};
\]

\[
 \text{VisMap}(u, v) \leftarrow z_{uv} \approx \text{ZBuffer}(x_{uv}, y_{uv});
\]

EndFor

EndFor

UV Domain of $4 \times 2$
Creating Visibility Maps V

UV domain pass II

Euclidean space pass I

ZBuffer(x, y) ≈ z₁
Creating Visibility Maps VII

Mold Design

Orthographic projection

Security

Perspective projection
Creating Visibility Maps VIII
Perspective projection I

Far plane
Near plane
XY aperture
Z aperture

Camera

Center for Graphics and Geometric Computing, Technion
Creating Visibility Maps IX
Perspective projection II

Combining visibility maps
Pixel Collapsing I

\( n \times n \times m \)

- \( 2^m \) possible pixels vector.
- \( n^2 \) different pixels vector at most.
- In practice, much less.
Pixel Collapsing II

Subfamily of the set of visibility maps

The union of the visibility maps
Reduction from SC to GC I

- We have shown a polynomial reduction from GC to SC. For completeness we will also show a polynomial reduction from SC to GC, proving that GC is NP-hard as SC is.

- We have a standard SC as described before.

- We will create a geometry corresponding to the universe $U$.

- We will create guards corresponding to the subsets of $U$.

- Solving the GC will solve the SC as well.
Reduction from SC to GC II

Subset of $U$ - a possible guard.

$U$ - a long strip.

Elements of $U$ - regions on the strip.
Reduction from SC to GC III

- $F$ - as many guards as are subsets in the problem, spread over the entire plane.
- All the upper strips are entirely covered by each of the guards.

GC is $NP$-hard
Examples

General Notes

- The following examples were created using Visibility maps of size $4096 \times 4096$.
- Both exhaustive (exponential) set cover solution and greedy (non-optimal) solution were sought.
- All implementation is software based and with single thread.
- In the examples we seek high coverage percent rather than a complete coverage.
Mold-Design Examples

General Notes

- The following examples were created using 266 views:
  - 130 general views around $S^2$, duplicated as $V$ and $-V$.
  - 6 views of $\pm X$, $\pm Y$, $\pm Z$. 

Center for Graphics and Geometric Computing, Technion
Example – a Cup Model

99.827% cover in greedy SC in ~4 seconds.

99.995% cover in exhaustive SC in ~10 hours.

First two view directions 95% cover.
Example – The Utah Teapot I

99.7\% cover in greedy SC in \textasciitilde6 seconds.
Example – The Utah Teapot II

99.7% cover in exhaustive SC in ~433 hours.
Security Examples

General Notes

- The following examples were created using about 300 guards/cameras.
- The guards were evenly spread on a curve or a plane.
A free form shape gallery
Examples

Cameras on the walls
Cameras on the wall - 2 cameras solution
Cameras on the ceiling
Cameras on the ceiling - 2 cameras solution
A military compound
A military compound - candidates above the perimeter
Candidates above the perimeter – 3 guards solution
A military compound - candidates above the compound
Candidates above the compound– 2 guards solution
Examples

Ben Gurion airport
Ben Gurion airport - candidate cameras
Ben Gurion airport - exhaustive 4 views solution

99.399% cover in exhaustive SC.
Conclusions and Future Work I

- We solve the GC problem in the parametric domain and reduce the analysis into the pixel level.
- Though we presented the framework in $\mathbb{R}^3$, nothing prevents the use of this framework in $\mathbb{R}^n$ for arbitrary $n$.
- The reduction to the discrete SC problem allows to optimally solve only discrete GC problems with a few views.
- We are looking for the solution in the continues problem.
Use of GPU in proposed framework can benefit the computation times (expect ~two orders of magnitudes).

Viewing angle and location distance limitations can be integrated into the creation of the visibility map.

Many of the visibility maps are very similar. Can we use this property to reduce set cover calculations?

The suggested framework can be used in other GC problems beside mold design and security.
End