Geometric Multi-Covering

Rouven Strauss[a], Florin Isvoranu[b], Gershon Elber[a]

[a] *Technion - Israel Institute of Technology*
[b] *Evolute GmbH*

Abstract

We present a general, unified framework to resolve geometric covering problems. The problem is reduced to a set cover search in parametric space. We propose and implement different methods for solving the set cover problem, allowing for flexible trade-offs between solution quality and computation time. Our framework relies on computer graphics techniques and heavily exploits GPU based computations.

Results are demonstrated in two specific applications: firstly, multi-visibility/accessibility analysis of 3D scenes that guarantees coverage, possibly redundant, of the target shape(s) by a minimal number of observers. Secondly, illumination design in 3D environments that ensures the satisfaction of local constraints on illuminance levels using a minimal set of lamps.

Keywords:
Geometric multi-covering, placement, k-coverage, visibility, lighting design, surveillance

1. Introduction

Geometric covering (GC) problems arise in many different fields. As part of the manufacturing process, products have to be entirely validated by a minimal number of inspection devices. In scenarios involving visual surveillance, e.g., in banks or in museums, critical areas have to be visible to an as-small-as-possible set of cameras or guards, possibly more than once for redundancy (see Figure 1). Public spaces require sufficient illumination while keeping the number of light sources minimal for energy conservation. In mold design and given a 3D artifact, a division of the artifact into a minimal set of assemblable mold parts is desired. When considering antenna networks, the objective is to achieve certain levels of service quality at different geographical locations, making it necessary to place a minimal set of antennas guaranteeing the service quality.

The aforementioned applications pose similar covering questions of geometric nature, where covering means the satisfaction of constraints on values assigned to objects in space, such as illuminance values, visibility requirements or quality-of-service levels. Due to their importance, GC questions have attracted a considerable amount of attention from various scientific disciplines such as computer graphics [1, 2], computational geometry [3, 4], manufacturing and mold design [5, 6], surveillance [7], inspection [8] and sensing theory [9].

In this work, we introduce a framework to tackle GC problems in a general, unified way. Our approach draws heavily from computer graphics techniques and offers solutions for the abovementioned applications as well as other GC queries. Being considered in various fields, studies about GC problems typically use different terminologies. For consistency and convenience, we first introduce the terminology used throughout this work.

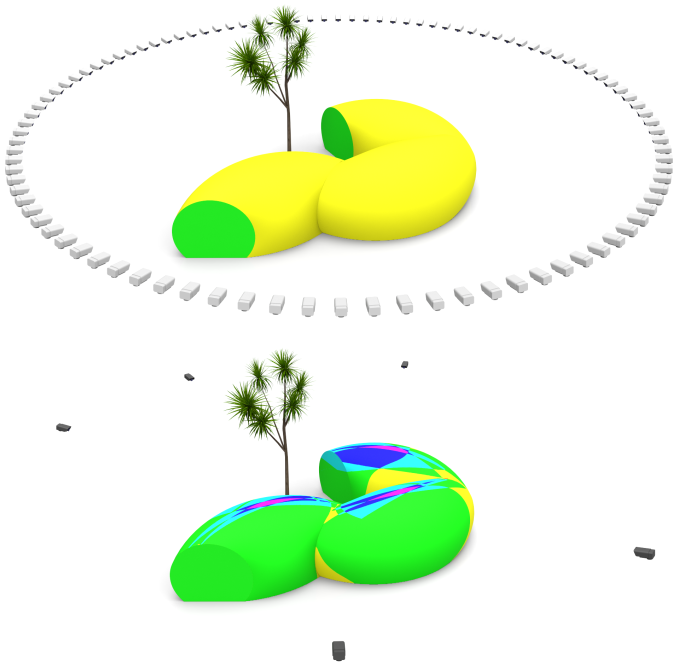

Figure 1: **Top:** A surveillance scenario in which a pavilion has to be inspected. The two entrances (in green) must be visible to at least two cameras, while the hull of the pavilion (in yellow) must be covered once. One hundred candidate cameras are shown. **Bottom:** Five selected cameras fulfill 98.5% of the coverage constraints specified on the pavilion's surface. The different colors show the actual coverage levels (0 - red; 1 - yellow; 2 - green; 3 - turquoise; 4 - blue; 5 - magenta).

Terminology. The space in which the GC problems are examined in our framework contains three types of objects:

- *targets*
- *sensors*
- *occluders*

An object in space that requires covering will be denoted as a **target**. Formally, a target, $\mathcal{T}$, is an m-manifold object in space, where $m \in \{1, 2\}$. We pose only one constraint on $\mathcal{T}$: it must possess a parametrization from an m-dimensional box, $D_{\mathcal{T}} : [d^i_{min}, d^i_{max}]^{m-1}_{i=0}$, such that it can be represented as $\mathcal{T} = \mathcal{T}(d^0, d^1, ..., d^{m-1})$. For example, in $\mathbb{R}^3$, the target can be a bivariate (trimmed) surface or a set of such surfaces, but also (a set of) 3-space curves. In Figure 1, the surface of an art pavilion serves as $\mathcal{T}$.

Any object in space that exhibits covering abilities is called a **sensor**. The sensors of a set $\mathcal{S}$ *cover* target locations, and can represent cameras or guards but also cellular antennas or light sources. Sensors are typically fixed points in space. The space in which the sensors reside is not required to equal the space in which the target resides. For example, while the target can be located $\mathbb{R}^2$, the sensors might reside in $\mathbb{R}^3$. A possible set of sensors is illustrated in Figure 1, where one hundred candidate cameras form $\mathcal{S}$.

The third class of objects, O, are called **occluders**. Based on their location in space, occluders reduce or even prevent the coverage of target locations by sensors. For instance, in GC problems dealing with visibility, occluders may be opaque objects blocking any inspection through them. In Figure 1, a tree serves as an occluder.

The sensors cover target locations with a certain **coverage level**. In turn, every target location has a **required coverage level** assigned to it. Based on the provided terminology, GC problems thus reduce to the question of *what is the best placement of a (minimal) number of sensors such that the required coverage levels are satisfied for all target locations*.

Certain GC problems share the property that the coverage levels are binary. For instance, in applications concerned with visibility, the coverage levels encode the states 'visible' and 'invisible'. We refer to such GC problems as **Binary Geometric Covering** (BGC) problems. However, in many GC problems, the covering is not necessarily binary. For instance, certain target locations may be required to be visible by several cameras, offering some redundancy. Similarly, multiple light sources may accumulatively illuminate target points to reach the desired illumination levels. A GC problem where the required coverage levels are above $\mathbb{R}$ is, therefore, denoted as a **Continuous Geometric Covering** (CGC) problem. Analogously, any GC problem in which the required coverage levels are above the natural numbers is called a **Discrete Geometric Covering** (DGC) problem.

Our Approach. The vast majority of previous work on GC problems explore the solution in the space in which the sensors and targets are situated, typically in 2D or 3D Euclidean space. In contrast, we propose an approach that reduces GC problems into a generic problem in the parametric domain of the target, $D_{\mathcal{T}}$. The problem is then discretely solved, exploiting its highly parallel nature and using computer graphics techniques.

Despite its discrete and therefore approximate character, our approach offers a simple, robust, and unified framework to address a large variety of GC problems, including the aforementioned ones. Furthermore, by reducing the problem to many simple (i.e., pixel) problems, we are able to handle and support **local** covering specifications. That is, any location on the target can have its own required coverage level. To the best of our knowledge, existing GC algorithms can only handle the global specification of a coverage requirement, i.e., achieving the same covering level for all target locations.

GC problems are considered difficult to solve. As stated, the majority of known GC algorithms operate in Euclidean space and, as part of the solution process, must resolve complex visibility and accessibility queries among different 3D objects, typically polygons. Moreover, GC problems are typically reduced to a set cover query and hence of expected exponential time complexity. In this work, we propose different methods for answering the set cover queries, each involving a different trade-off regarding solution quality, runtime and extensibility. For higher efficiency, we take advantage of GPU computation capabilities wherever possible.

Organization. The rest of this work is organized as follows: after discussing previous work concerned with different types of GC problems in Section 2, we provide a formal definition of GC problems in Section 3. In Section 4, we present our algorithmic approach for computationally solving GC problems. Two different computer graphics related GC applications are discussed in Section 5: multi-visibility analysis by cameras or guards and illuminance satisfaction by multiple light sources. Finally, we discuss the proposed approach and possible extensions in Section 6, and conclude this work in Section 7.

2. Related Work

One of the classical GC problems is arguably the famous Art Gallery Problem [7]. While an extensive amount of GC problem variations has since been addressed in literature, we restrict our discussion mostly to related work solving problems in three dimensions. Following [10], we refer to problems requiring a globally defined covering level k as k-coverage problems.

Inspection. Tarbox and Gottschlich [11] sample sensors on a sphere surrounding a three-dimensional target and choose a set of locations on the surface of the target. Different heuristic-based algorithms are used to find a small, but not necessarily minimal set of sensors such that all target locations are 1-covered. Restricting themselves to simple polyhedra, Roberts and Marshall [8] attempt to find sets of faces of a given target in $\mathbb{R}^3$ that are 1-covered by a common viewpoint while minimizing the number of viewpoints.

Computer Graphics. Fleishman et al. [1] tackle a BGC problem aiming at automatically creating a small number of rendered images of a given 3D scene such that most surfaces in the scene are 1-covered. They furthermore require that any surface be covered at most once, in order to obtain non-redundant images.

Mold Design. In mold design, 1-cover must be precisely satisfied, i.e., every target location belongs to exactly one mold part. Liu and Ramani [5] process the geometry in Euclidean space, while Shragai and Elber [6] work in parametric space. The latter work is similar to our approach and can be seen as a special case of it.

Sensor Networks. Lu et al. [9] assign to every sensor s a probability function mapping every location in $\mathbb{R}^2$ to a probability with which it is detected by s. All required coverage levels are the same, representing the expected number of sensors detecting the corresponding target location. The resulting k-coverage problem is solved using a greedy method. Becker et al. [12] consider 1-coverage of every voxel of a 3D volume, employing a greedy algorithm to select a small number of sensors.

Lighting Design. GC problems also arise in lighting design that seeks to automatically place lights such that a prescribed illumination is achieved. The problem of approximating a predefined illumination setting is tackled by Schoeneman et al. [2], Marschner and Greenberg [13] as well as Contensin [14] by means of least squares methods. However, to the best of our knowledge, no previous work aims at solving the problem of *guaranteeing* minimal light intensities at all target locations with a minimal number of light sources.

3. Continuous Geometric Coverage Problems

In this section, we provide a general definition of CGC problems. Consider a target $\mathcal{T}$ and recall that $D_\mathcal{T}$ is the parametric domain of $\mathcal{T}$. The required coverage levels of the target locations are prescribed, via $D_\mathcal{T}$, by a *coverage requirement function*:

Definition 1. *The coverage requirement function* $\mathbf{R} : D_\mathcal{T} \to \mathbb{R}_{\geq 0}$ *specifies for each parametric location, $d \in D_\mathcal{T}$, a required coverage level that has to be reached at target location $\mathcal{T}(d)$.*

The required coverage levels are defined locally, i.e., different target locations can be assigned different coverage levels. This is desirable since in real-world scenarios there may be some parts of an object that must be covered with a different coverage level than other parts.

Similar to the required coverage levels, we define the coverage levels that a sensor provides to the locations on the target, potentially influenced by occluders:

Definition 2. *The sensor coverage function* $\mathbf{C}_O : D_\mathcal{T} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ *assigns to every parametric location, $d \in D_\mathcal{T}$, a value $\mathbf{C}_O(d, s)$ with which target location $\mathcal{T}(d)$ is covered by sensor $s \in \mathcal{S}$, and which depends on the set of occluders, O.*

Coverage levels are usually additive, e.g., if they represent illuminance values [15] or the number of cameras seeing a certain target location. Hence, having defined the covering that each sensor provides to different points of target $\mathcal{T}$, we can consider the cumulative covering contribution of several sensors.

Definition 3. *A target location, $\mathcal{T}(d)$, is said to be k-covered by a set of sensors, $\mathcal{S}$, if*

$$\sum_{s \in \mathcal{S}} \mathbf{C}_O(d, s) \geq k, \quad k \in \mathbb{R}_{\geq 0}.$$

CGC problems can now be defined in the following way:

CGC Problem Statement. Given ($\mathcal{S}$, $\mathcal{T}$, $D_\mathcal{T}$, $\mathbf{C}_O$, $\mathbf{R}$), find the set $\hat{\mathcal{S}} \subseteq \mathcal{S}$ of minimal size satisfying

$$\sum_{s \in \hat{\mathcal{S}}} \mathbf{C}_O(d, s) \geq \mathbf{R}(d), \quad \forall d \in D_\mathcal{T}. \tag{1}$$

Less rigorously, CGC problems ask for a minimal cardinality set of sensors ensuring that each target location is covered at least to the extent of its required coverage level. In case that $\mathbf{C}_O : D_T \times \mathcal{S} \to \mathbb{N}$ and $\mathbf{R} : D_T \to \mathbb{N}$, the problem is a DGC problem. BGC problems are obtained if $\mathbf{C}_O : D_T \times \mathcal{S} \to \{0, 1\}$ and $\mathbf{R} : D_T \to \{1\}$.

Due to the difficulty of computationally solving CGC problems as defined above, the continuous domain of the target, $D_\mathcal{T}$, is discretized. Discretizing $D_\mathcal{T}$, being an m-dimensional box, $m \in \{1, 2\}$, yields a finite sample of parametric locations which we denote by D. Recall that the coverage requirement function assigns a required coverage level to each parametric location $d \in D_\mathcal{T}$ and hence also every $d \in D$. The discretization thus provides a discrete m-dimensional map whose values can readily be serialized into a *coverage requirement vector* $\overline{\mathbf{R}} \in \mathbb{R}_{\geq 0}^{|D|}$, e.g., via the lexicographic order of the elements in D. Further, we assume a finite set of candidate sensors, which yields the *sensor coverage matrix* $\overline{\mathbf{C}_O} \in \mathbb{R}_{\geq 0}^{|D| \times |S|}$. CGC problems can now be approximated by the following optimization problem:

$$\begin{aligned} \mathit{Minimize} \ & \|\overline{x}\|_0 \\ \mathit{subject\ to} \ & \overline{x} \in \{0, 1\}^{|S|} \\ & \overline{\mathbf{C}_O} \cdot \overline{x} \geq \overline{\mathbf{R}}, \end{aligned} \tag{2}$$

where $\|\overline{x}\|_0$ denotes the number of non-zero entries of $\overline{x}$.

A vector $\overline{x}$ solving optimization problem (2) represents the optimum choice of the sensors in this discretized solution. Let the i'th element of vector $\overline{x}$ be denoted by $\overline{x}_i$. Furthermore, assume that the elements in $\mathcal{S}$ are enumerated and let s_i denote the i'th sensor in $\mathcal{S}$. The solution $\hat{\mathcal{S}}$ of the discretized CGC problem is hence given by $\hat{\mathcal{S}} := \{s_i \in \mathcal{S} \mid \overline{x}_i = 1\}$.

Basing the definition of both the sensor coverage matrix $\overline{\mathbf{C}_O}$ and the coverage requirement vector $\overline{\mathbf{R}}$ on $D_\mathcal{T}$, the parametric domain of the target $\mathcal{T}$, allows the reduction of the problem complexity to a discrete, yet local and dense pixel analysis which yields several advantages: Firstly, complex and potentially unstable visibility computations in Euclidean space can be substituted by simple, parallelizable comparisons in parametric space. This renders an efficient computation of $\overline{\mathbf{C}_O}$ possible in various applications, as we will show in Section 4.1. Furthermore, the solution of optimization problem (2) can be computed in a parallel fashion. Last, but not least, the discreteness and locality of our approach allows the specification of different coverage levels for distinct target locations.

In the remainder of this work, we refer to optimization problem (2) as the discretized CGC problem. The discretized CGC problem contains the *minimum set cover optimization problem* [16] as a special case and is thus NP-hard which is straightforward to show. Hence, it is indicated to resort to potentially non-optimal, but efficient approaches.

4. Algorithmic Approach

The target $\mathcal{T}$ may consist of one *continuous* manifold or collections of such. Clearly, the parametric locations in the discretized set D should approximate the original shapes as good as possible. Proper sampling of the parametric domain of $\mathcal{T}$ can lead to a good selection of corresponding target locations which have to be covered, since the differential geometry of manifold $\mathcal{T}$ and the mapping of its parametrization can be taken into account. That said, the optimal sampling of the parametrization of freeform manifolds or mesh models is a question beyond the scope of this work and was investigated by many (e.g., [17, 18]). As a first order approximation, a uniform grid in the parametric space can be sampled. This sampling is used in our implementation and our examples, where the targets are (trimmed) surfaces in $\mathbb{R}^3$. Note that the definition of the discretized CGC problem is unaffected from the way the set D is computed.

Since the required coverage levels are known in advance, the discretization directly yields the coverage requirement vector $\overline{\mathbf{R}}$. However, the same often does not hold true for the sensor coverage matrix $\overline{\mathbf{C}_O}$, which is a necessary parameter required for solving the discretized CGC problem. In order to obtain $\overline{\mathbf{C}_O}$, one usually has to explicitly compute the coverage levels provided by the set of candidate sensors, $\mathcal{S}$. This computation depends on the specific application, and is discussed in Section 4.1.

With a discretized $D_{\mathcal{T}}$ and $\mathcal{S}$, one can only hope for an approximate answer. However, thanks to the discretization, both computational tasks, creating the sensor coverage matrix and solving the discretized CGC problem, are highly parallelizable and we can, therefore, perform the computations on a GPU.

4.1. Computing the Sensor Coverage Matrix

In this work, we address two GC applications in $\mathbb{R}^3$. The *multi-visibility problem* (*MVP*) pursues the minimal set of guards, out of $\mathcal{S}$, that see $\mathcal{T}$ with enough redundancy to satisfy $\overline{\mathbf{R}}$. The *illuminance satisfaction problem* (*ISP*) seeks a minimal set of lamps, from a given set $\mathcal{S}$, which satisfies some illuminance covering requirements, $\overline{\mathbf{R}}$, over some target $\mathcal{T}$. We will further elaborate on these specific problems in Section 5. We now discuss the computation of $\overline{\mathbf{C}_O}$ for these two GC applications.

Conceptually, the values of $\overline{\mathbf{C}_O}$ are easy to determine. In the *MVP*, $\overline{\mathbf{C}_{O d,s}}$ is 0 if target location $\mathcal{T}(d)$ is invisible to sensor s and 1, otherwise. Determining whether s sees $\mathcal{T}(d)$ depends on both the occluders and the target that can possibly occlude itself when viewed from certain directions. In the *ISP*, an entry $\overline{\mathbf{C}_{O d,s}}$ is an illuminance value $I \in \mathbb{R}_{\geq 0}$ received by $\mathcal{T}(d)$ from s.

Preprocessing. We assume that $\mathcal{T}$ is provided as a set of (trimmed) NURBS surfaces or as a polygonal mesh model with corresponding UV parametrization $D_{\mathcal{T}}$. The same assumption is made for the set of occluders, O, with the difference that no parametrization is required. Both $\mathcal{T}$ and O are tessellated for graphics processing, keeping the UV values of $\mathcal{T}$ at the triangle vertices. A unique ID is assigned to each triangle of $\mathcal{T}$, while all triangles of O receive the same ID, distinct from those assigned to the triangles of $\mathcal{T}$.

Rendering. The remaining step is a two-pass z-buffer rendering algorithm that is executed for each sensor s separately. Rendering pass I is performed in the original 3D Euclidean space. In this pass, z-buffer rendering is used to scan-convert the tessellated target and the occluders, employing frusta that incorporate the view properties of s. In case of the *ISP*, s is a point light source, and therefore, has a 360° field of vision, whereas in the *MVP* s represents a camera or a guard with an appropriate field of view. In both cases, an arbitrary number of frusta may be used to model the required views. At every pixel, the ID of the corresponding visible triangle is stored.

In rendering pass II, the actual values of $\overline{\mathbf{C}_O}$ are computed. $\mathcal{T}$ is z-buffer scan-converted in the UV parameter space, yielding the discrete set of target locations D in the form of pixels of a 2D texture. This computation is straightforward, efficient with regard to both time and space and requires minimal knowledge about the parametrization. For each pixel representing a parametric target location d we check whether the ID of the triangle to which $\mathcal{T}(d)$ belongs matches the ID of the corresponding target location $\mathcal{T}(d)$ projected to the stored z-buffers from pass I. This comparison enables us to decide the visibility of $\mathcal{T}(d)$ from sensor s without performing complex visibility computations in Euclidean space. In case of a match, $\mathcal{T}(d)$ is considered visible to sensor s; otherwise it is regarded as invisible. In the *MVP*, $\overline{\mathbf{C}_{O d,s}}$ is set directly to values 0 or 1 depending on the visibility. In the *ISP* and if $\mathcal{T}(d)$ is visible to s, $\overline{\mathbf{C}_{O d,s}}$ is computed according to the parameters determining the illuminance.

The required steps are described in a more concise way in Algorithm 1. Function *CoverageLevel* in line 14 returns the coverage level according to the specific problem. In the *MVP*, *CoverageLevel* always returns 1 since $\mathcal{T}(d)$ is visible to s. In the *ISP*, *CoverageLevel* computes the illuminance value of $\mathcal{T}(d)$ in accordance with the distance between $\mathcal{T}(d)$ and s, the luminous flux of s, the angle of incidence, and additional parameters.

Implementation Details. The two described rendering passes share a common property: they perform pixel operations which are independent of each other and can be performed in parallel on a GPU. Both passes involve the creation of numerous large textures. The substantial size of the textures is essential in order to prevent errors caused by the sampling deviation between the Euclidean projection and the parametric domain. In our implementation, 26 textures, are computed per sensor in pass I, covering 360°. In pass II, the entries of 2D matrix $\overline{\mathbf{C}_O}$ are computed, creating one column for each sensor, where each column is a vector of linearized 2D textures. We use OpenGL and GLSL to conduct the computations.

Algorithm 1 Computing $\overline{\mathbf{C}_O}$

```
Input: $\mathcal{T}, \underline{D}, \mathcal{S}, O \subset \mathbb{R}^3$
Output: $\overline{\mathbf{C}_O}$
1: Tesselate $\mathcal{T}$ and $O$, yielding sets of triangles $T_{\mathcal{T}}$ and $T_O$, respectively;
2: Assign ID 0 to all triangles $\triangle \in T_O$;
3: Assign unique positive IDs to all triangles $\triangle \in T_{\mathcal{T}}$;
4: for each sensor $s \in \mathcal{S}$ do
5:     Pass I:
6:     z-buffer render $T_{\mathcal{T}}$ and $T_O$ from $s$ onto corresponding projection plane $P$, keeping only the ID of the visible triangle at each pixel;
7:     Pass II:
8:     z-buffer render $T_{\mathcal{T}}$ in parametric UV space, yielding $D$ in the form of pixels of a 2D texture
9:     for each pixel $d \in D$ do in parallel
10:        $\underline{\text{ID}}(d) \leftarrow$ ID of triangle $\triangle$ s.t. $\mathcal{T}(d) \in \triangle$;
11:        $\overline{\mathbf{C}_{O}}_{d,s} \leftarrow 0$;
12:        $(x, y) \leftarrow$ coordinates of projection of $\mathcal{T}(d)$ on $P$;
13:        if $\underline{\text{ID}}(d) = \text{ID}(P(x, y))$ then
14:            $\overline{\mathbf{C}_{O}}_{d,s} \leftarrow CoverageLevel(s, \mathcal{T}(d))$;
```

Due to their high memory consumption, it is usually impossible to store all textures computed per sensor simultaneously in GPU memory. However, both the set of textures created in pass I and the texture computed in pass II, representing the coverage sensor matrix $\overline{\mathbf{C}_O}$, can be subdivided. This separability property allows the two-pass rendering to be performed in several iterations and is exploited in our implementation, enabling us to increase accuracy as much as required despite memory limitations.

4.2. Solving Continuous Coverage Problems

Having $\overline{\mathbf{C}_O}$, we can now compute a solution of the discretized CGC problem. Diverse algorithms can be applied for this purpose; in this work, we consider three different methods, each of which offers a different trade-off regarding solution quality, runtime and extensibility.

The discretized CGC problem is a linear integer program that can be solved optimally by a variety of solvers, given it is feasible. Feasibility can be verified by checking whether $\overline{\mathbf{x}} := 1^{|D|}$ solves the problem. An unfeasible problem can be transformed into a feasible one by removing the unsatisfiable constraints. In the remaining work, we assume that the unsatisfiable constraints and the corresponding target locations are removed.

1. Optimal Solution. We employ *Gurobi Optimizer 5.5* [19] in order to obtain an optimal solution of the discretized CGC problem. The sensors in the returned solution guarantee that *all* coverage constraints are satisfied by a minimal number of sensors. However, it might be the case that this optimal solution consists of many sensors, while a much smaller set of sensors achieves the required coverage levels at *almost all* target locations. Such a smaller set might be more practical if the lack of satisfaction of several required coverage levels is tolerable. In addition, one might desire more than one solution in order to be able to filter the involved sensor sets based on other criteria. It is therefore useful to exhaustively check the sets of sensors with small cardinality.

2. Exhaustive Exploration of Solution Space. Solving the discretized CGC problem up to a certain solution quality is possible by evaluating the coverage achieved by every possible choice of sensors, i.e. by performing an exhaustive search. Sensor combinations guaranteeing the required coverage levels for a sufficient number of target locations are stored, allowing for sorting or filtering according to different criteria, e.g. the sum of minimal distances between the sensors.

However, even if both $|\mathcal{S}|$ and the allowed number of sensors in the solution $\hat{\mathcal{S}}$ are rather small, the total number of potential solutions is extremely large. In addition, for each possible sensor choice, the satisfaction of $|D|$ coverage constraints has to be verified, where D might contain several millions of target locations. Thus, exhaustive computation is possible only for sensor combinations involving few sensors. On this account, fast methods allowing a flexible solution quality are desirable.

3. Greedy Algorithm. A third and very efficient algorithm computes an approximate solution of the discretized CGC problem in a greedy fashion. Before describing the greedy algorithm, an auxiliary procedure, Algorithm 2, is defined. The procedure computes how much a given sensor can decrease the remaining coverage gap, based on the L^1 distance of the corresponding column in $\overline{\mathbf{C}_O}$ from the current coverage requirement vector $\overline{\mathbf{R}}$.

Algorithm 2 *Remainder*

```
Input: $s \in \mathcal{S}, \overline{\mathbf{C}_O} \in \mathbb{R}_{\geq 0}^{|D| \times |\mathcal{S}|}, \overline{\mathbf{R}} \in \mathbb{R}_{\geq 0}^{|D|}$
Output: $r$
1: $r \leftarrow 0$;
2: for each $d \in D$ do in parallel
3:     $r \leftarrow r + \max\{0, \overline{\mathbf{R}}_d - \overline{\mathbf{C}_{O}}_{d,s}\}$;
```

The problem can now be solved by iteratively adding the sensor which reduces the coverage gap the most and updating the coverage requirement vector $\overline{\mathbf{R}}$ accordingly. This greedy method is formalized in Algorithm 3. In case that the GC problem under consideration is a DGC problem, i.e., both $\overline{\mathbf{C}_O}$ and $\overline{\mathbf{R}}$ contain only natural numbers, Algorithm 3 provides a solution of bounded size. Dobson [20] shows that in this case the solution size is at most $O(|\hat{\mathcal{S}}_{opt}| \log a)$, where $\hat{\mathcal{S}}_{opt}$ is a solution set of minimal size and $a = \max_{s \in \mathcal{S}} \sum_{d \in D} \overline{\mathbf{C}_{O}}_{d,s}$, i.e., the maximal sum of all coverage levels of a sensor.

Implementation Details. Since the coverage constraints are independent of each other, verifying their satisfaction can be done in parallel. We implemented both the exhaustive and the greedy algorithm in OpenCL [21], exploiting the computational power of the GPU. In lines 2-3 in Algorithm 2 and lines 8-9 in Algorithm 3, simple parallel computations are performed for each parametric location. In practice, these computations are executed by separate OpenCL work-items.

Algorithm 3 Greedy Approximation Algorithm for discretized CGC Problems

```
Input: $D, \mathcal{S}, \overline{\mathbf{C}_O} \in \mathbb{R}_{\geq 0}^{|D|\times|\mathcal{S}|}, \overline{\mathbf{R}} \in \mathbb{R}_{\geq 0}^{|D|}$
Output: $\hat{\mathcal{S}} \subseteq \mathcal{S}$
1: if $\overline{\mathbf{C}_O} \cdot \overline{\mathbf{1}}^{|\mathcal{S}|} \ngeq \overline{\mathbf{R}}$ then
2:     return "no solution";
3: $\hat{\mathcal{S}} \leftarrow \emptyset$;
4: while $\overline{\mathbf{R}} \neq \overline{\mathbf{0}}$ do
5:     $s \leftarrow \arg\min_{s\in\mathcal{S}} \left\{Remainder(s, \overline{\mathbf{C}_O}, \overline{\mathbf{R}})\right\}$;
6:     $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \cup \{s\}$;
7:     $\mathcal{S} \leftarrow \mathcal{S} \backslash \{s\}$;
8:     for each $d \in D$ do in parallel
9:         $\overline{\mathbf{R}}_d \leftarrow \max\{0, \overline{\mathbf{R}}_d - \overline{\mathbf{C}}_{Od,s}\}$;
```

The exhaustive algorithm computes every combination of sums of columns of $\overline{\mathbf{C}_O}$ and compares the resulting vector element-wise with $\overline{\mathbf{R}}$. For each combination, the number of satisfied coverage constraints is stored in GPU memory. Due to the vast amount of combinations and the consequently arising memory issues, the computation is performed in iterations, each treating thousands of combinations. Between the iterations, the stored data is passed to the CPU for further processing.

5. Applications

The presented definition of CGC problems subsumes many geometric covering problems arising in different scientific fields. In this work, we present solutions to two computer graphics-related GC problems:

The Multi-Visibility Problem (MVP). The objective of the *MVP* is to place a minimal number of observers in a scene such that every target location in the scene is visible to enough observers. The *MVP* is a special case of a DGC problem in the following way: the observers, e.g. cameras or guards, form the set of sensors $\mathcal{S}$. The occluders are opaque obstacles limiting the visibility of the sensors. A sensor can have a restricted field of view and it can have limits on the maximal distances it covers. These limitations are encoded in the sensor coverage matrix, $\overline{\mathbf{C}_O}$, computed in the rendering process (i.e., by Algorithm 1). Visibility is usually considered a dichotomic notion, assuming two states - 'visible' and 'invisible'. The entries of $\overline{\mathbf{C}_O}$ are consequently binary, indicating the visibility of the target locations by the sensors, while the entries of $\overline{\mathbf{R}}$ range above the natural numbers, representing for each target location the number of sensors it must be visible to.

The Illuminance Satisfaction Problem (ISP). In the *ISP*, light sources have to be placed such that every target location $\mathcal{T}(d)$ is lit at least with the illuminance value defined by the corresponding entry of the coverage requirement vector $\overline{\mathbf{R}}$. Like in the *MVP*, occluders are assumed to be opaque objects in the scene. The illuminance induced by the light sources is incorporated into $\overline{\mathbf{C}_O}$ and depends on several parameters: the visibility of $\mathcal{T}(d)$ by point light source s, the luminous flux of s, the distance between $\mathcal{T}(d)$ and s, and the angle of incidence. Although the *ISP* is a CGC problem, we discretize the range of possible levels for the illuminance values and consider an approximately equivalent DGC problem with 2^8 distinct illuminance levels.

We demonstrate our results based on two *MVP* examples and two *ISP* instances. The finite set of candidate sensors is predetermined manually in a way which attempts to adhere to the nature of each specific example. After computing the sensor coverage matrix, solutions of the GC problems are computed using the three described algorithms. Due to its complexity, the exhaustive algorithm is run with a limited maximum size of the solution set. Table 1 provides statistics on the sizes of the used data sets and the corresponding computation times.

MVP Example 1 (Pavilion). Our first *MVP* was introduced in Figure 1 and is based on a real pavilion[1] in London (see Figure 2). In the example, the coverage constraints require every target location in the two pavilion entrances to be visible to at least two cameras, while every target location on the pavilion's hull must be seen by at least one camera. Figure 1 presents a solution involving five cameras which has been computed by the exhaustive algorithm. The composition of the solution is shown in Figure 3. In each frame one camera is added, showing the progressively achieved coverage. Jointly, the five cameras satisfy the coverage constraints at 98.5% of the target locations. Satisfaction of all coverage constraints requires the employment of seven cameras.

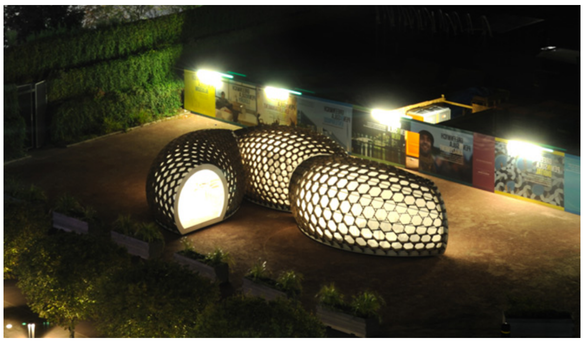
Figure 2: The real pavilion used in Figure 1. Used with permission.

MVP Example 2 (Car Chassis). Another scenario in which multi-visibility is required can be seen in Figure 4. The chassis of a manufactured car needs to be visually inspected at certain locations, and with certain redundancy. While 18 cameras (out of 170) are necessary for satisfying all coverage constraints, four cameras are enough to satisfy 99.9% of the constraints as is revealed by the exhaustive algorithm.

ISP Example 1 (Ruins). In Figure 5, we present an instance of an *ISP* involving a scene of ruins. Illumination coverage requirements are assigned to specific parts of the ruins and a set

[1]See also http://www.kreod.com

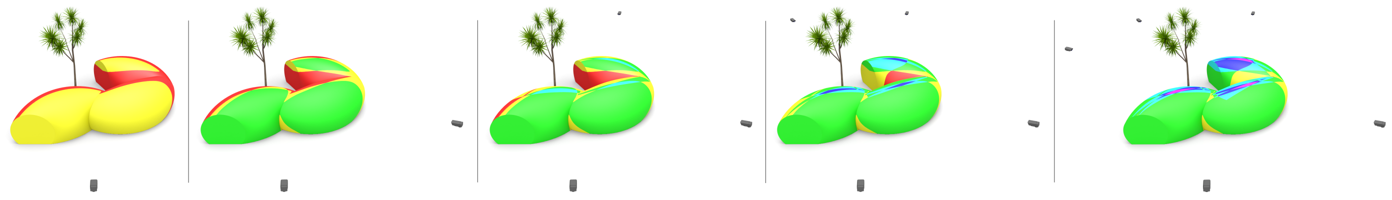

Figure 3: **From left to right:** The solution illustrated in Figure 1 and computed by the exhaustive algorithm is shown in parts, iteratively adding cameras. The same color coding for the coverage is used (0 - red; 1 - yellow; 2 - green; 3 - turquoise; 4 - blue; 5 - magenta).

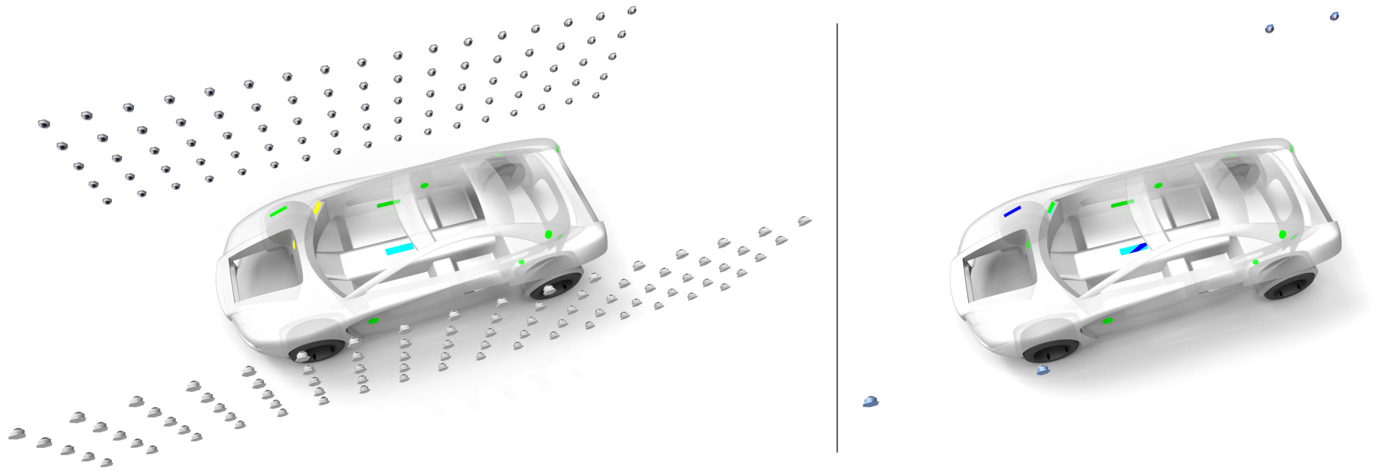

Figure 4: **Left:** A car chassis during an inspection process. Twelve patches of target locations have to be inspected with different redundancy (1 - yellow; 2 - green; 3 - turquoise; 4 - blue). 170 different positions for the cameras are allowed. The transparency is for better visualization only. **Right:** While 18 cameras are required for complete coverage, by means of the exhaustive algorithm we find that merely four cameras guarantee sufficient coverage for over 99.9% of the target locations.

of candidate light sources is provided. Due to the intricacy of the model, almost all sensors have to be used for achieving full coverage. However, the exhaustive algorithm finds a solution of only five light sources sufficiently illuminating over 70% of the selected area.

ISP Example 2 (Sculpture). Our last example consists of a sculpture for which we seek partial illumination (see Figure 6). The entries of the coverage requirement vector $\overline{\mathbf{R}}$, i.e. the illuminance values we attempt to reach, are provided by an approximate area light source illuminating the sculpture from one side. We sample sensors representing point light sources around the target and solve the ISP problem using the three different algorithms. In order to satisfy all coverage constraints, nine sensors are required. In contrast, by means of the exhaustive algorithm we find a solution satisfying over 87.7% of the constraints, while consisting of only four point light sources.

Observations. As can be seen in table 1, executing Algorithm 1 on the GPU made the construction of the sensor coverage matrix $\overline{\mathbf{C}_O}$ feasible within a few minutes despite the vast number of involved operations. The required coverage levels of some target locations cannot be reached even by the union of all sensors due to various reasons such as self-occlusion of the target, the rather sparse sampling of the candidate sensor set or — in case of the *ISP* — too low luminous fluxes.

In our examples, the greedy method performs not much worse than the optimal one. Also in comparison with the exhaustive procedure, the greedy algorithm performs well. In addition to the results displayed in table 1, we find that the greedy algorithm needs one sensor more than the exhaustive method in order to achieve at least the same percentage of coverage. These findings suggest that the greedy method is suitable for efficiently computing a solution of reasonable quality.

6. Discussion and Future Work

Geometric covering problems surface in many disciplines. Unfortunately, their relation to set cover renders them very difficult to solve, even beyond the complexity of processing the involved geometry in a numerically stable way. In this work, we perform a pixel level analysis in a different parametric space which reduces the geometric complexity, much like the z-buffer mechanism. By exploiting computer graphics tools, we get a dense sampling of the continuous problem and achieve a reasonable covering approximation of the geometry. The quality of this approximation can be adjusted conveniently by changing the sampling density and employing different solving methods.

Extensibility. As long as one can evaluate the contribution of sensor s to $\mathcal{T}$, our framework can be used for a diverse set of GC problems, e.g.,

- placement of cellular antennas, taking into account the decay in electromagnetic transmission due to concrete walls,

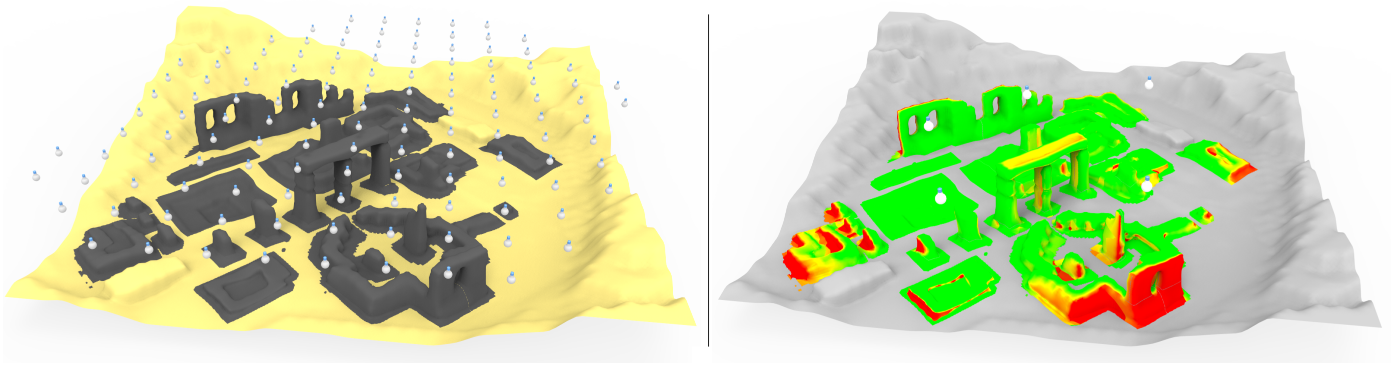

Figure 5: **Left:** A scene comprised of ruins which have to be sufficiently lit by a minimal number of point light sources. The required illumination on the target is provided in gray, while the yellow areas depict parts of the model for which no required coverage levels have been specified. **Right:** The exhaustive algorithm computes a solution consisting of five light sources (shown as white bulbs) that guarantee the required coverage levels at more than 70% of the relevant area. In contrast, almost all sensors are required for maximum satisfaction of the coverage constraints (see table 1). The deviation between the required coverage levels and the coverage levels induced by the light sources in the solution is shown. The colors linearly blend from red over yellow to green, indicating for each target location which percentage of its required coverage level has been reached (0% - pure red; 50% - pure yellow; 100% or more - pure green).

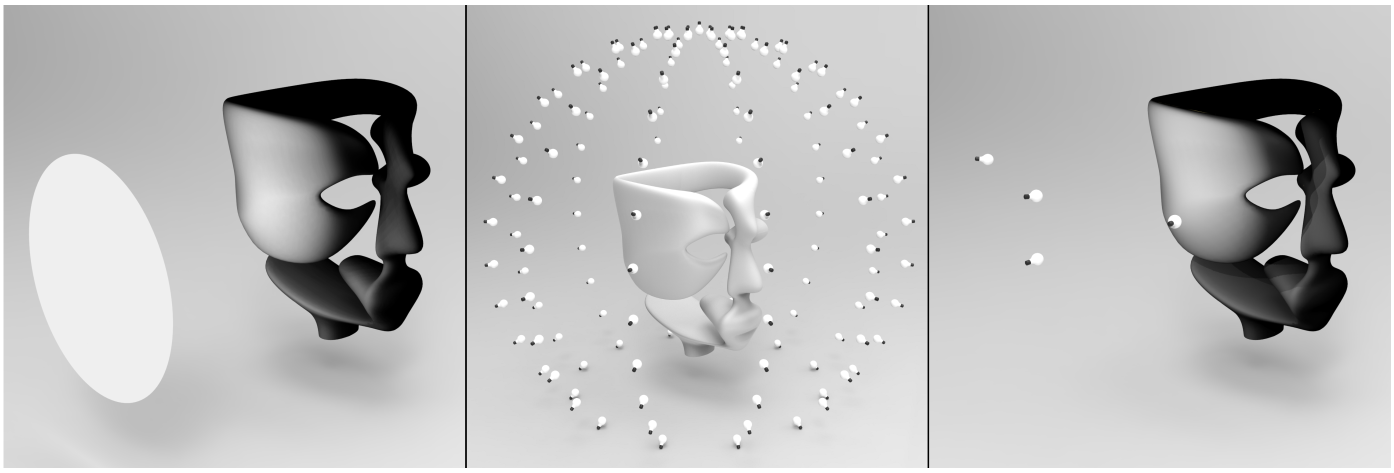

Figure 6: **Left:** A facial sculpture illuminated by an area light source. The resulting illumination distribution provides the required coverage levels. **Middle:** 145 point light source candidates are positioned on a sphere surrounding the target. **Right:** Four light sources — together reaching more than 87.7% of the required coverage levels — are found by the exhaustive algorithm. Satisfaction of all coverage constraints involves nine light sources.

Table 1: Runtime data employing Intel ® Core™i7-3770 CPU @ 3.40 GHz and AMD ® Radeon™HD 7970 (3 GB RAM)

Type	**Model**	**# sensors**	**# target locations**	**Computation of $\overline{C_O}$ (min:sec)**	**# removed unsatisfiable constraints**	**Algorithm**	**Maximum allowed solution size**	**Solution size**	**Solving time**	**Percentage of satisfied constraints**
MVP	Pavilion	100	3,383,528	01:30	4,476	Optimal	$\|S\|$	7	~ 1 min 20 s	100.000
						Greedy	$\|S\|$	10	~ 1.5 s	100.000
						Exhaustive	5	5	~ 10 h	98.500
						Greedy	5	5	< 1 s	97.400
	Car Chassis	170	907,872	01:36	117	Optimal	$\|S\|$	18	~ 25 s	100.000
						Greedy	$\|S\|$	20	~ 1 s	100.000
						Exhaustive	4	4	~ 48 min	99.976
						Greedy	4	4	< 1 s	99.125
ISP	Ruins	128	2,609,728	04:59	2,670	Optimal	$\|S\|$	121	~ 3 min 30 s	100.000
						Greedy	$\|S\|$	123	~ 6 s	100.000
						Exhaustive	5	5	~ 10 h	70.646
						Greedy	5	5	< 1 s	65.833
	Sculpture	145	3,086,624	02:02	0	Optimal	$\|S\|$	9	~ 15 min 40 s	100.000
						Greedy	$\|S\|$	10	~ 1 s	100.000
						Exhaustive	4	4	~ 49 min	87.734
						Greedy	4	4	< 1 s	64.652

- surveillance from the air,
- or, sprinkler placement for fire extinction or plant irrigation.

In addition, one can generalize the problem to allow for upper covering limits, e.g., in order not to flood the target with too much light in the case of the *ISP*.

Limitations. The main drawback of the framework is its discreteness. While it is highly desirable to directly handle GC problems where $\mathcal{T}$, $\mathcal{S}$ and the coverage requirement function, $\mathbf{R}$ are all continuous, such continuous solutions still constitute an open question. The discrete sampling of locations on $\mathcal{T}$ is acceptable, considering that the sampling is quite dense. In contrast, the set of sensor candidates is sampled less densely which is rather restricting and leads to an approximate answer that needs improvement. However, due to the complexity of the problem, neither the optimal solver nor the exhaustive algorithm scale well enough with the number of sensors in order to allow for a much denser sampling of sensors. Only the greedy algorithm can cope with a significantly larger candidate sensor set.

Future Work. The presented framework has room for improvement. A very desirable enhancement that will increase the accuracy of the results will consist in considering the actual area a pixel covers in the original, typically Euclidean, space. Based on the parametrization of $\mathcal{T}$, one can use the Jacobian of the mapping at every pixel in order to estimate the area contribution of this pixel in the original space. The proposed algorithms could then be modified accordingly by assigning weights to the corresponding coverage constraints.

Predetermining the set of sensor candidates, $\mathcal{S}$, is restricting. Possible directions for future research also include the improved and automated construction of $\mathcal{S}$. Simulated Annealing techniques [22] or other heuristics could be applied to alleviate this limitation.

In case of the *ISP*, it is desirable to compute the intensities of the light sources as part of the framework, rather than fixing them from the beginning. Last but not least, it is desirable to allow sensors which are not limited to a single point in space, but take up a volume. In this case, area light sources could be used in the *ISP*.

7. Conclusion

In this work, we presented a general framework to compute an approximate solution to geometric covering problems. Unlike most previous work that typically operates in Euclidean space, we operate in the parametric space of the target, which allows for a unified and highly parallelizable approach. In contrast to any previous work, our framework supports the specification of different required covering levels for different target locations. Several algorithmic methods solving the involved set cover queries are proposed and applied to two applications of interest in computer graphics and other fields, namely, multi-visibility analysis and illuminance satisfaction.

Acknowledgements

The research leading to the presented results has received partial funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement PIAP-GA-2011-286426 and was supported by the Israel Science Foundation (grant No.278/13). We thank Chun Qing Li for the permission to use the pavilion model. The photograph in Figure 2 is a courtesy of Jaap Oepkes and Pavilion Architecture.

References

[1] S. Fleishman, D. Cohen-Or, D. Lischinski, Automatic Camera Placement for Image-Based Modeling, Computer Graphics Forum 19 (2) (2000) 101–110. doi:10.1111/1467-8659.00447.

[2] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, D. Greenberg, Painting with light, in: 20th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH 1993, 1993, pp. 143–146. doi:10.1145/166117.166135.

[3] Y. Amit, J. S. B. Mitchell, E. Packer, Locating Guards for Visibility Coverage of Polygons, International Journal of Computational Geometry & Applications 20 (05) (2010) 601–630. doi:10.1142/S0218195910003451.

[4] T. Erlebach, E. J. V. Leeuwen, Approximating Geometric Coverage Problems, in: 19th annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2008, pp. 1267–1276.

[5] M. Liu, K. Ramani, On Minimal Orthographic View Covers for Polyhedra, in: IEEE International Conference on Shape Modeling and Applications, 2009, 2009, pp. 96–102. doi:10.1109/SMI.2009.5170169.

[6] N. Shragai, G. Elber, Geometric Covering, Computer-Aided Design 45 (2) (2013) 243–251. doi:10.1016/j.cad.2012.10.007.

[7] J. O'Rourke, Art Gallery Theorems and Algorithms, Oxford University Press, 1987.

[8] D. Roberts, A. Marshall, Viewpoint Selection for Complete Surface Coverage of Three Dimensional Objects, in: British Machine Vision Conference, 1998, pp. 740–750. doi:10.1.1.41.1437.

[9] J. Lu, L. Bao, T. Suda, Coverage-Aware Sensor Engagement in Dense Sensor Networks, in: International Conference on Embedded and Ubiquitous Computing, 2005, 2005, pp. 639–650. doi:10.1.1.144.7672.

[10] C.-F. Huang, Y.-C. Tseng, The Coverage Problem in a Wireless Sensor Network, in: 2nd ACM International Conference on Wireless Sensor Networks and Applications, 2003, 2003, pp. 115–121. doi:10.1145/941350.941367.

[11] G. Tarbox, S. Gottschlich, Planning for Complete Sensor Coverage in Inspection, Computer Vision and Image Understanding 61 (1) (1995) 84–111. doi:10.1006/cviu.1995.1007.

[12] E. Becker, G. Guerra-Filho, F. Makedon, Automatic Sensor Placement in a 3D Volume, in: 2nd International Conference on Pervasive Technologies Related to Assistive Environments, 2009, 2009, pp. 36:1–36:8. doi:10.1145/1579114.1579150.

[13] S. Marschner, D. Greenberg, Inverse Lighting for Photography, in: IST/SID Fifth Color Imaging Conference, 1997, pp. 262–265. doi:10.1.1.62.5025.

[14] M. Contensin, Inverse Lighting Problem in Radiosity, Inverse Problems in Engineering 10 (2) (2002) 131–152. doi:10.1080/10682760290004311.

[15] D. C. Pritchard, Lighting, Longman Scientific & Technical, 1995.

[16] R. M. Karp, Reducibility among Combinatorial Problems, in: Complexity of Computer Computations, Springer US, 1972, pp. 85–103. doi:10.1007/978-1-4684-2001-2_9.

[17] L. Liu, L. Zhang, Y. Xu, C. Gotsman, S. J. Gortler, A Local/Global Approach to Mesh Parameterization, in: Proceedings of the Symposium on Geometry Processing, Vol. 27, 2008, pp. 1495–1504. doi:10.1111/j.1467-8659.2008.01290.x.

[18] N. Pietroni, M. Tarini, P. Cignoni, Almost Isometric Mesh Parameterization through Abstract Domains., IEEE Transactions on Visualization and Computer Graphics 16 (4) (2010) 621–35. doi:10.1109/TVCG.2009.96.

[19] Gurobi Optimization Inc., Gurobi Optimizer Reference Manual (2013). URL `http://www.gurobi.com`

[20] G. Dobson, Worst-case Analysis of Greedy Heuristics for Integer Programming with Nonnegative Data, Mathematics of Operations Research 7 (4) (1982) 515–531. doi:10.1287/moor.7.4.515.

[21] A. Munshi, The OpenCL Specification - Version 1.2, 2011.

[22] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by Simulated Annealing, Science 220 (1983) 671–680. doi:10.1.1.123.7607.