Sensor Placement for Globally Optimal Coverage of 3D-Embedded Surfaces

Si Wei Feng Kai Gao Jie Gong Jingjin Yu

***Abstract*— We carry out a structural and algorithmic study of a mobile sensor coverage optimization problem targeting 2D surfaces embedded in a 3D workspace. The investigated settings model multiple important applications including camera network deployment for surveillance, geological monitoring/survey of 3D terrains, and UVC-based surface disinfection for the prevention of the spread of disease agents (e.g., SARS-CoV-2). Under a unified general "sensor coverage" problem, three concrete formulations are examined, focusing on optimizing visibility, single-best coverage quality, and cumulative quality, respectively. After demonstrating the computational intractability of all these formulations, we describe approximation schemes and mathematical programming models for near-optimally solving them. The effectiveness of our methods is thoroughly evaluated under realistic and practical scenarios.**

I. Introduction

We perform a systematic study of a class of mobile sensor[1] deployment optimization problems targeting the coverage of 2D surfaces embedded in 3D domains, applicable to a broad set of practical settings, for example: (1) the selection of surveillance camera locations for maximizing the joint coverage of an art museum, (2) the deployment of mobile robots with range-based sensing apparatus for the monitoring of complex 3D terrains with quality assurances, and (3) the optimization of UVC light source locations for disinfecting interior surfaces of public indoor spaces, e.g., airplanes, buses, hospital rooms, and schools (Fig. 1), which is of key relevance to the ongoing COVID-19 pandemic. Despite the problems' apparent differences, the above-mentioned tasks fall under the general problem of placing mobile sensors for optimizing some form of coverage. This work is devoted to providing such a unifying problem formulation, understanding its rich structural properties, and delivering effective computational methods for readily solving the multiple variations, which all have high application potentials.

As a summary of the research and its contributions, under a general formulation, three coverage optimization problems are examined that are based on *visibility*, *best quality*, and *cumulative quality*, respectively. The visibility model only takes into account line-of-sight sensing. In the best quality model, the coverage quality of a point in the environment is determined by the closest visible sensor, i.e., the quality is determined by distance. A max-min optimization over this quantity is performed. In the cumulative quality model, the coverage of a point is the sum of coverage by all visible

All authors are at Rutgers, the State University of New Jersey, Piscataway, NJ, USA. S. W. Feng, K. Gao, and J. Yu are with the Department of Computer Science; J. Gong is with the Department of Civil and Environmental Engineering. E-Mails: {siwei.feng, kai.gao, jiegong.cee, jingjin.yu} @ rutgers.edu. The work is supported in part by NSF awards IIS-1734419 and IIS-1845888.

[1]As will be explained, "mobile sensor" is used here in a broad sense.

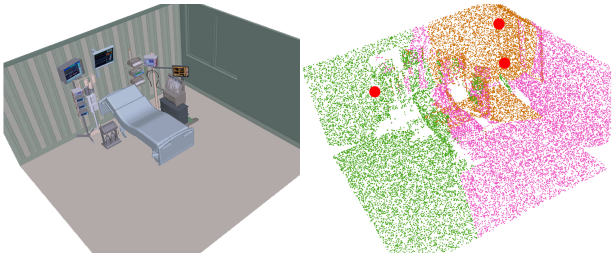


Fig. 1: [left] The 3D model of an intensive care unit (ICU) in a hospital. [right] A near-optimal coverage of the ICU using three UVC light sources deployed on the ceiling of the room (shown as red discs) where a minimum level of exposure dose can be guaranteed. Each color (orange, green, pink) marks the covered surface locations of a given UVC source. Areas with insufficient exposure are also readily shown as scattered white regions, which can be eliminated through adding more UVC sources.

"sensors". For this model, the area over which a minimum quality can be guaranteed is maximized. Even in simpler 2D settings, these formulations, are known to induce significant computational challenges. They are frequently NP-hard and sometimes hard to approximate, which we briefly discuss. On the algorithmic side, we show how some of these problems can be approximately solved in polynomial time and then develop general integer programming methods, assisted with local improvements, for quickly computing high quality (i.e., $(1+\varepsilon)$-optimal) solutions. Extensive evaluations are performed over multiple realistic application scenarios, confirming the effectiveness of our algorithmic solutions.

The general formulation and specific models investigated in this work have their origins in two main lines of research: Art Gallery [1]–[3] and studies on mobile sensor networks, e.g., [4]–[9]. Our visibility-based model has deep roots in Art Gallery problems [1]–[3], which commonly assume a sensor model based on line-of-sight visibility [10]; a main task is to guard every point in the interior of a bounded 2D regions (a point is guarded when it is visible to at least one of the guards). Depending on the exact formulation, guards may be placed on boundaries, corners, or the interior of the region. Not surprisingly, Art Gallery problems are typically NP-hard [11]. Other than Art Gallery, 2D coverage problems with other sensing models, e.g., disc-based, have also been considered [5], [12]–[16]. Some formulations prevent the overlapping of individual sensing ranges [12], [13] while others seek to ensure a full coverage which often require overlapping sensor coverage.

In an influential body of work [5], [6], a gradient-based iterative method was devised that drives multiple mobile sensors to a locally optimal coverage configuration, with convergence guarantees. Whereas [5], [6] assume availability of gradients *a priori*, such information can also be learned

[8]. Subsequently, the method is further extended to allow the coverage of non-convex and disjoint 2D domains [17] and to work for mobile robots with heterogeneous capabilities [16]. In contrast to these iterative local interaction-based methods, this work emphasizes the direct computation of globally optimal solutions under challenging 3D settings.

Distributed sensor coverage [5], [8] builds on the study of facility location problems [14], [18] examining the selection of facility (e.g., warehouses) locations that minimize the cost of delivery of goods to spatially distributed customers. These are known (e.g., in operations research and computer science) as clustering problems [19], with many variations depending on the cost structure. Our investigation benefits from the vast literature on clustering and related problems, e.g., [20]–[24]. Clustering problems are in turn related to packing [13], tiling [12], and Art Gallery problems [1], [2].

This work is a continuation of our systematic effort [25]–[27] at tackling sensor coverage problems. Our earlier studies focus on 1D/2D sensors models covering 1D/2D domains, which are significantly less complicated than the 3D settings examined in the current study.

The rest of the paper is organized as follows. In Section II, we provide an umbrella problem statement and detail the three coverage models. Computational complexity of the formulations is briefly discussed. In Section III, we describe effective algorithmic solutions for solving these sensor coverage problems. Extensive evaluation results are provided in Section IV containing multiple realistic application settings. We conclude the work in Section V.

II. Preliminaries

A. Sensor Placement for Optimal Coverage: Formulations

Let $\mathcal{E} \subset \mathbb{R}^3$ be a bounded three-dimensional workspace that is path-connected, e.g., a hilly terrain or an intensive care unit (ICU) in a hospital. We consider the problem of deploying k "mobile sensors", $c_1, \ldots, c_k$, to guard a *critical subset* S embedded in the surface of $\mathcal{E}$, i.e., $S \subset \partial\mathcal{E}$ where ∂ is the boundary operator. For example, if $\mathcal{E}$ is a hospital ICU, S may be a part of its interior surface. The sensors are to be deployed to achieve a globally optimal coverage of S satisfying some prescribed objective, to be made more precise as the problems are further grounded for specific sensor models. We denote this broad class of problems as *Sensor Placement for Optimal Coverage* (SPOC).

The terms *mobile sensor* and *coverage* are used in a broad sense. Beyond traditional sensors that only collect information, we are interested in mobile robots with means to effect the environment as well. For example, a mobile robot may be equipped with a disinfecting light source (e.g., UVC) for eradicating harmful microbes (e.g., viruses and bacteria). Nevertheless, such settings can be nicely captured under a general sensor coverage formulation.

In this study, we explore two common types of coverage models: *visibility*-based and *quality*-based. In a *visibility*-based sensor coverage model, as the term suggests, a point $p \in S$ is considered covered by a sensor c if p is visible from c. When there are more than one sensor, a point p is covered if it is visible from any sensor $c_i \in \{c_1, \ldots, c_k\}$. In a *quality*-based model, the coverage quality of a point $p \in S$ is captured by some function that potentially depends on the sensors, the point p, and its neighborhood in S. For example, one type of quality measurement can be based on the inverse of the distance between a point p and its closest sensor c.

Formally, we capture the different sensor models under a unified function $f(c_1, \ldots, c_k, p, \mathcal{E})$ whose co-domain is non-negative reals, i.e., $f : \mathbb{R}^{3k+3} \times \mathcal{E} \to \mathbb{R}_+ \cup \{0\}$. In what follows, $\mathcal{E}$ is omitted but understood to be part of the input to f. In this paper, the following instantiations are considered:

- Let $vis(p, c) = 1$ if the point p is visible to the sensor c. Otherwise, $vis(p, c) = 0$. For example, an omni-directional visibility model would have $vis(p, c) = 1$ if the open line segment between p and c does not intersect $\mathcal{E}$. In a model based purely on **visibility**,
$$f(c_1, \ldots, c_k, p) := \max_{1 \le i \le k} vis(p, c_i). \tag{1}$$
- Let the *coverage quality* of a point p by a sensor c_i be represented as a function $\phi(p, c_i) \in \mathbb{R}_+ \cup \{0\}$. In a **quality maximization** sensor model, the coverage quality for a point p is determined by a single best sensor:
$$f(c_1, \ldots, c_k, p) := \max_{1 \le i \le k} \phi(p, c_i) vis(p, c_i). \tag{2}$$
- In a **cumulative quality** sensor model, the overall quality of coverage at a point p is the sum of the effects of all visible sensors:
$$f(c_1, \ldots, c_k, p) := \sum_{1 \le i \le k} \phi(p, c_i) vis(p, c_i). \tag{3}$$

All above models have direct and practical applications. A visibility model is applicable to the deployment of a network of 360° cameras for monitoring. Letting $\phi(p, c_i) = \|pc_i\|^{-1}$ ($\|pc_i\|$ denotes the distance between p and c_i), the quality maximization model becomes the k-center problem [18] if we optimize $\min_p f(c_1, \ldots, c_k, p)$, a broadly applicable problem. For the cumulative quality model, when the "sensors" are UVC lights, one may ask the question of how to optimally place these lights to ensure the highest percentage of S can be exposed to sufficient UVC light for eradicating SARS-CoV-2 and other microbes. Here, a cumulative quality model clearly makes sense.

To fully ground the discussion that follows, we formulate three concrete optimization problems, one for each of the above-mentioned sensor models.

Problem II.1 (Visibility Maximization)**.** *Given $S \subset \partial\mathcal{E}$ with $\mathcal{E} \subset \mathbb{R}^3$, $c_i \in \mathbb{R}^3$, $1 \le i \le k$, and $f(c_1, \ldots, c_k, p)$ from* (1)*, determine a placement of $c_1, \ldots, c_k$ that maximizes the support of f, i.e.,*
$$supp(f) = \{p \in S \mid \max_{1 \le i \le k} vis(p, c_i) > 0\}.$$

Problem II.2 (Quality Maximization)**.** *Given $S \subset \partial\mathcal{E}$ with $\mathcal{E} \subset \mathbb{R}^3$, $c_i \in \mathbb{R}^3$, $1 \le i \le k$, and $f(c_1, \ldots, c_k, p)$ from* (2) *with $\phi(p, c_i) = \|pc_i\|^{-1}$, determine a placement of $c_1, \ldots, c_k$ that maximizes the minimum coverage quality, i.e.,*
$$\min_{p \in S} \max_{1 \le i \le k} \frac{vis(p, c_i)}{\|pc_i\|}.$$

Problem II.3 (Cumulative Quality). *Given $S \subset \partial\mathcal{E}$ with $\mathcal{E} \subset \mathbb{R}^3$, $c_i \in \mathbb{R}^3$, $1 \leq i \leq k$, and $f(c_1, \ldots, c_k, p)$ from* (3) *with $\phi(p, c_i) = \|pc_i\|^{-2}\langle \hat{n}_p, \hat{n}_{pc_i}\rangle$ where $\hat{n}_p$ is the unit normal of S at p and $\hat{n}_{pc_i}$ is the unit vector in the direction from p to c_i, determine a placement of $c_1, \ldots, c_k$ that maximizes the coverage area where f is above a given threshold $\Phi > 0$,*

$$supp(f - \Phi) = \{p \in S \mid \sum_{1 \leq i \leq k} \frac{vis(p, c_i)\langle \hat{n}_p, \hat{n}_{pc_i}\rangle}{\|pc_i\|^2} > \Phi\}.$$

An implicit assumption for Problem II.2 to be meaningful is that an arbitrary $p \in S$ is visible to the closest sensor, which limits the choice of S. Problem II.2 is a suitable model for, e.g., surveillance applications that cannot tolerate any blind spots. A generalization with less limitation can require a certain percentage of S, e.g., 80%, to have optimized coverage. Problem II.3, which computes coverage quality using the formula $\langle \hat{n}_p, \hat{n}_{pc_i}\rangle \|pc_i\|^{-2}$, takes after a standard light exposure model that depends on the inverse of the squared distance and incoming light angle with respect to a local surface region (see Fig. 2).

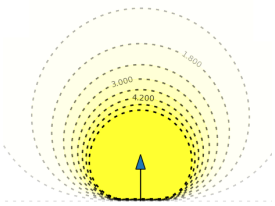


Fig. 2: For a point with the normal shown as the arrow, light sources on the same dotted curve provide same level of exposure.

A 2D illustration of the three models is given in Fig. 3.

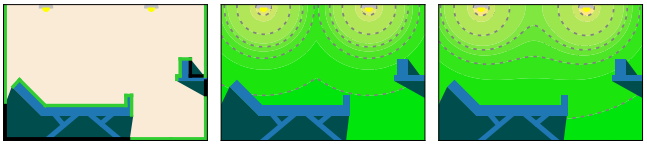


Fig. 3: Illustration of the effects of two sensors (or light sources) under three different models. Only a 2D slice of a simple ICU with a bed and a counter is shown for clarity. [left] A visibility-based sensing model where the green segments show the visible surfaces. [middle] A quality maximization model where the dotted lines show the equal-quality level sets. $\phi(p, c_i) = \|pc_i\|^{-1}$ [right] A cumulative quality model with $\phi(p, c_i) = \|pc_i\|^{-2}\langle \hat{n}_p, \hat{n}_{pc_i}\rangle$. Again, the dotted lines show the additive quality. The impact of the surface normal is not displayed in the figure.

Problems II.1-II.3 allow sensor locations to be anywhere in $\mathcal{E}$. In practice, sensor locations are often limited to a 2D surface. For example, in a museum or a bus, cameras are mounted on walls and ceilings. For drones surveying an area, there is often a preferred height to fly at. With this in mind, whereas algorithms we develop are general, the evaluation is mainly focused on practical settings where sensors locations are confined to some 2D surface.

B. Computational Complexity

As the computation of 3D visibility is a well-known hard problem [28] Problems II.1-II.3 are all computationally intractable because they all involve, as part of the solution, computation of 3D visibility sets. The involvement of multiple robots/sensors introduces additional sources of computational complexity, which we briefly discuss.

Problem II.1 may be viewed as an Art Gallery [1] problem in 3D. The basic 2D Art Gallery problem, which asks the question that how many guards with omnidirectional visibility are needed to ensure that every point in a simply-connected (2D) polygon is visible to at least one guard, is shown to be NP-hard [11]. Problem II.1 is then also NP-hard through reducing the 2D Art Gallery problem to a 3D one by creating a third dimension that is very "thin".

Our recent work [27] shows that a 2D version of Problem II.2, called Optimal Set Guarding (OSG), is NP-hard to approximate within a factor of 1.152. Similarly, we may reduce OSG to Problem II.2 by adding a thin third dimension. Therefore, through this route, we know that optimal solutions to Problem II.2 is hard to approximate within a factor of 1.152, even when the surface S is a simple polygon.

From an instance of Problem II.2, reduced from an instance of OSG, we can add further 3D structures to obtain an instance of Problem II.3 such that cumulative effect from multiple sensors are limited. That is, when sensors are forced to have no interactions, a version of Problem II.3 that is similar to Problem II.2 is obtained, which is again NP-hard.

We summarize the discussion in Theorem II.1. Full proofs of these complexity results, which are too lengthy to be included here and are not as essential in comparison to the problem formulations and the algorithmic results, will be detailed in an extended version of this work.

Theorem II.1. *Problems II.1-II.3 are NP-hard.*

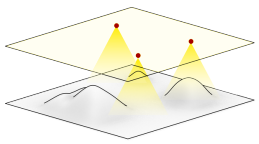
III. Fast Computation of High-Quality Solutions

In this section, we first describe a polynomial time approximation algorithm for a restricted version of Problem II.2. Then, we describe a general integer linear programming framework for solving Problems II.1-II.3, and local improvement techniques for enhancing solution quality.

A. Polynomial Time Approximation Algorithm

In their general forms, Problems II.1-II.3 require the computation of 3D visibility, a hard task on its own. Due to this reason, a polynomial time algorithm with guaranteed good approximation ratio for these problems appear difficult to come by. It is an interesting question to ask whether some form of approximation scheme can be derived. Here, we show that for Problem II.2, if one relax the visibility requirement, i.e. letting $vis(\cdot, \cdot) \equiv 1$, then a polynomial time $(2 + \varepsilon)$-approximation algorithm can be obtained.

Taking Problem II.2, we examine a setup assuming that each point $p \in S$ has good visibility, i.e., p is always visible to the nearest sensor. Such scenarios happen when the 3D domain does not have large curvatures that would easily block sensors' view, e.g., covering the earth with GPS satellites or using drones to survey a vineyard. To drive a specific approximation bound, we further assume that sensors have spherical range sensing and are in a plane of some fixed height h from the ground, which may be relaxed. Denote this surface as H_C. The figure on the right provides an illustration of the target environment setting.



The main idea is to first obtain a dense sample of S and then adapt 2-approximation algorithms for the corresponding 2D setting, which requires some non-trivial reasoning.

Two well-known approximation algorithms for k-center like problem in 2D are based on *farthest point clustering* [22] and *dominating set* [21], [29]. Both of these approaches work for our purpose; we show how to work with the former.

Let a uniformly sampled set of point of S be $S_N = \{o_1, \ldots, o_N\}$. We apply farthest point clustering [22] on S_N as follows. As the name suggest, it picks farthest sensor locations until the number of sensors are exhausted. In the original approach, the points to be clustered are also sensor locations, which is not true here. Instead, we perform clustering in the set S_N and project the selected samples to H_C gradually. The relatively straightforward process is given in Algorithm 1 (($d(\cdot, \cdot)$ denotes the distance between the inputs, one or both of which may be sets).

Algorithm 1: Farthest Point Clustering

Input : S_N=$\{o_1, \ldots, o_N\}$: N sampled points on the surface $S \subset \partial\mathcal{E}$; k: number of sensors; H_C: a plane with a fixed height
Output: $\mathcal{C}$: sensor location set

$\mathcal{C} \leftarrow \{o_1$'s vertical projection onto $H_C\}$
for $i \leftarrow 1$ **to** k **do**
  for $o \in S_N$ **do**
    compute the distance $d(o, \mathcal{C})$ between o and $\mathcal{C}$
  $o \leftarrow$ point $v \in S_N$ with the largest $d(v, \mathcal{C})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{o$'s projection onto $H_C\}$
return $\mathcal{C}$

To prove the claimed $(2+\varepsilon)$-approximation bound, denote the optimal sensor location set and the sensor location set derived by Algorithm 1 as $\mathcal{C}_{OPT}$ and $\mathcal{C}$, respectively. Since these are centers of spherical sensing ranges, we call them center set for short. Denote the minimum coverage radius in the spherical sensing model as r_{OPT}. Let h be the minimum distance between surface S and sensor space H_C, i.e. $h := d(S, H_C)$. r_{OPT} and $r_{\mathcal{C}}$ are defined as follows:

$$r_{OPT} := \max_{o \in S_N} d(o, \mathcal{C}_{OPT}) \tag{4}$$

$$r_{\mathcal{C}} := \max_{o \in S_N} d(o, \mathcal{C}) \tag{5}$$

Proposition III.1. *The center set obtained by Algorithm 1 achieves coverage radius of at most* $\sqrt{4r_{OPT}^2 - 3h^2}$.

Proof. Denote the center set generated at the ith round as $\mathcal{C}_i$, and r_i as the cluster radius $r_i := \max_{o_\tau} \min_{c_j \in \mathcal{C}_i} d(o_\tau, c_j)$. It is straightforward to observe that $r_k \leq r_{k-1} \leq \cdots \leq r_1$. Consider the relationship between the optimal center set $\mathcal{C}_{OPT}$ and the center set obtained by Algorithm 1, we have the following 2 cases.

Case 1: For each sphere $\mathcal{B}_c$ centered at a point $c \in \mathcal{C}_{OPT}$ with radius of r_{OPT}, the projection of $\mathcal{B}_c \bigcap S$ onto the sensor space H_C contains exactly one point of $\mathcal{C}_k$.

In this case, let v be an arbitrary point in S. Let c_α be the nearest center to v in $\mathcal{C}_{OPT}$ and c_β be the point in $\mathcal{C}_k$ whose projection on S is inside $\mathcal{B}_{c_\alpha}$. Therefore, we have:

$$d(v, c_\beta) \leq \sqrt{4r_{OPT}^2 - 3h^2} \tag{6}$$

Case 2: There exists a sphere $\mathcal{B}_c$ centered at a point $c \in \mathcal{C}_{OPT}$ with radius of r_{OPT}, the projection of $\mathcal{B}_c \bigcap S$ onto the sensor space H_C contains at least 2 points of $\mathcal{C}_k$. In this case, denote the two centers by c_i and c_j $(i < j)$, and their projections on S are in the same sphere $\mathcal{B}_c$. As c_j is added after c_i, then,

$$\begin{aligned} r_{\mathcal{C}} = r_k \leq r_j &\leq d(c_i, c_j\text{'s projection on } S) \\ &\leq \sqrt{4r_{OPT}^2 - 3h^2} \end{aligned} \tag{7}$$

Summarizing the two cases proves Proposition III.1. □

B. Integer Programming-Based Algorithmic Framework

With Problems II.1-II.3 being computationally intractable, a natural algorithmic alternative is mathematical programming. In [27], an integer linear programming (ILP) model was shown to be effective for a 2D setting. For our 3D problems, visibility constraints must be effectively handled. We pre-compute pairwise visibility at a given sample granularity. The information is then fed to an ILP model. As the discretization granularity gets smaller, we can then realize *globally optimal* $(1 \pm \varepsilon)$-approximations (depending whether it is a maximization or a minimization).

As a first step to building the ILP model, visibility information must be computed. We work with two discretizations, the surface S to be covered and the space where the sensors may be deployed (as discussed in Section II, this is a 3D space though in practice it is frequently a 2D subset). For each pair of samples, we use a collision checker [30] to determine whether the line segments between the two samples intersects $\mathcal{E}$. During the process, we also compute for each sample $p \in S$ its normal $\hat{n}_p$.

With the visibility pre-computation performed, we are ready to construct the fully ILP models. For all three problems, recall that we have $S_N = \{o_1, \ldots, o_N\}$ for discretizing the surface S through grid-based sampling. We use boolean variable y_i to indicate whether sample o_i is covered. Candidate sensor locations are also discretized to obtain a sample set $\{c_1, \ldots, c_M\}$, from which k locations would be selected with z_i indicating whether c_i is selected. The ILP model for For Problem II.1 is

$$y_i \leq \sum_{j \ s.t. \ vis(o_i, c_j)=1} z_j \quad \text{for each } o_i \tag{8}$$

$$\sum_j z_j \leq k \tag{9}$$

$$\max \quad y_1 + \cdots + y_N \tag{10}$$

The cumulative quality case (Problem II.3) is similar. Denoting the sensing quality between sensor location c_j and surface point p as $\phi(p, c_j) = vis(p, c_j) \cdot (\hat{n}_p, \hat{n}_{pc_j})/||pc_j||^2$, the ILP model may be constructed as

$$y_i \cdot \Phi \leq \sum_j \phi(o_i, c_j) \cdot z_j \quad \text{for each } o_i \tag{11}$$

$$\sum_j z_j \leq k \tag{12}$$

$$\max \quad y_1 + \cdots + y_N \tag{13}$$

For quality maximization (Problem II.2), the objective is to maximize the minimum distance of a sampled point on the surface to its nearest sensor location. For a required coverage ratio ρ and radius r, we can verify whether it is possible to

put k sensors and cover $N\rho$ discretized points by checking the feasibility of the following model:

$$y_i \leq \sum_{j \ s.t. \ ||c_j - o_i|| \leq r} z_j \quad \text{for each } o_i \tag{14}$$

$$\sum_j z_j \leq k \tag{15}$$

$$N\rho \leq \sum_i y_i \tag{16}$$

A subsequent binary search can be applied to find the smallest feasible r.

C. Local Enhancement of Coverage Quality

Whereas the ILP models for Problems II.1-II.3 support arbitrary precision, given that these problems are all computationally intractable, it can be expected that a pure ILP-based solution will only be scalable up to a certain point before an exorbitant amount computation time is needed. Inspired by the iterative update approach form [5], we propose a two-phase optimization pipeline of using ILP (or the approximation algorithm for Problem II.2) as the first phase with a good level of *global* optimality guarantee and follow that with *local* improvements that can be quickly computed to enhance the initial solution. We note that, as the local improvement is enhancing a solution with a level of global optimality guarantee, the enhancement is also global in effect. For example, in Problem II.2, if we start with a 2-approximation solution and obtain an initial coverage quality r and subsequent local improvement reduce that to $0.75r$, then the final solution is a globally 1.5-optimal solution.

We develop two such routines. The first is generally applicable and straightforward to implement: as the discretization level increases, we move the set of initial sensor locations (computed by Algorithm 1 or ILP) locally, one at a time. More formally, given an initial solution $\mathcal{C} = \{c_1, \ldots, c_k\}$, denote $S_j \subset S$ as the region covered (possibly partially when working with Problem II.3) by the sensor deployed at c_j. For each S_j, we try improving the quality of the solution by finding a better location for c_j to cover S_j at a finer resolution. Subsequently, S_j can be updated based on the new c_j. The process may be repeated until convergence.

The second local improvement routine is via solving a "1-cener" like problem and is applicable to Problem II.2 and Problem II.3. Due to limited space, we omit the lengthy algorithmic details and give a high-level description. For Problem II.2, a sensor located at c_j is "responsible" for visible points of S that falls within a ball $B(c_j, r)$. Our improvement routine examines $S \cap B(c_j, r)$ and attempts to compute a new ball with a smaller radius that covers all of $S \cap B(c_j, r)$. The routine uses the ideas from Welzl's algorithm for computing minimum enclosing discs [31], [32] and take time that is expected linear with respect to the number of samples that falls within $B(c_j, r)$, which is fairly fast. This method can be readily extended to Problem II.3 where the spheres become "distorted" (Fig. 2).

IV. Experimental Evaluation

For each of Problems II.1-II.3, extensive experimental evaluations were carried out to evaluate our proposed algorithmic solutions. Here, we present representative evaluation demonstrating the effectiveness of our methods, with a focus on three realistic settings (the ICU model from Fig. 1, bus and subway car models shown in Fig. 4). For all environments, the surface S is selected to be all visible surfaces not facing downward. Due to limited space, result on the (2+ε)-approximation algorithm (Algorithm 1) is omitted (as shown in [27], such methods are fast but are quite sub-optimal). The experiments were carried out on a median-end quad-core Intel i7 processor with 16GiB RAM. Algorithms were implemented in C++. Gurobi [33] was used as the Integer Programming solver. Source code is available at `https://github.com/rutgers-arc-lab/3d_coverage`.

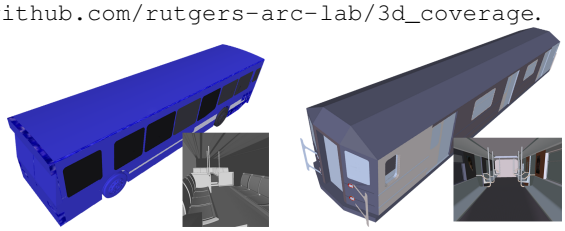


Fig. 4: Realistic 3D environments used in our evaluation in addition to the ICU model from Fig. 1. [left] A 40-foot large bus model and its interior. [right] A subway car and its interior.

Results on Problem II.1, using ILP, is given in Fig. 5. For each model, 600 candidate sensor locations and $20,000$ coverage surface points are sampled using grids. As expected, the surface coverage ratios increase as the number of sensors increase, approaching full coverage. We note that certain surface region is not visible, e.g., ground underneath seats in subway cars, leading to plateaus below 100% coverage. The computation time is very reasonable for offline computations. The spikes in the middle of the computation time plot correspond to hard cases when the visibility coverage is about to plateau. We also observe that subway $<$ ICU $<$ bus in terms of computation time, which may be explained by the interior complexity of these environments. This aspect is different across the three problems.

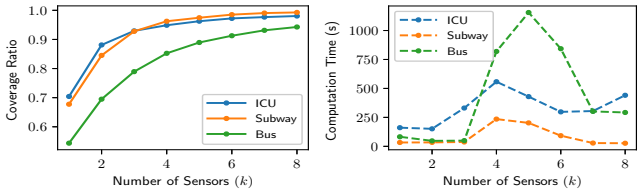
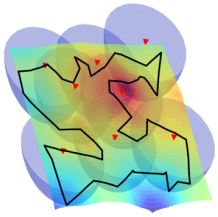


Fig. 5: Coverage quality and computation time for Problem II.1 for the three environments as the number of sensors change.



In evaluating Problem II.2, we first examine a case where mobile sensors (e.g., camera drones) are deployed to cover a synthetic terrain with relatively small curvature, i.e., $vis(\cdot, \cdot) \equiv 1$. An illustration of the setting is given in the figure on the right, where the color indicates the height of the terrain. The sensors (8 red triangles in the figure) are placed at a fixed height above

the terrain and must guard the region enclosed in the black curve. Spherical range sensing is assumed. For the setup (600 sensor locations and $20,000$ surface points), computation time and solution quality as the number of sensors changes are listed in the table. Computation time decreases as the number of sensors increases, indicating the problem is harder when sensors are too few to provide a good coverage. It also shows that the ILP running time does not depend positively on sensor quantity. The quality (smaller is better) increase becomes minimal as sensor quantity reaches 10.

#Sensors	2	4	6	8	10	12	14	16
Time (s)	42.9	25.4	18.5	12.7	13.1	11.3	7.64	6.70
Radius	5.10	3.31	2.76	2.43	2.27	2.16	2.07	1.99

In a second evaluation of Problem II.2, visibility is considered with the optimization coverage ratio set to 80%. That is, at least 80% of the maximum visible target surface (for a given k) will be guaranteed the achieved coverage quality. The result, summarized in Fig. 6, again demonstrates a negative correlation between the computational time and the number of sensors. Here, however, the computational time is $20+$ times more than when having full visibility.

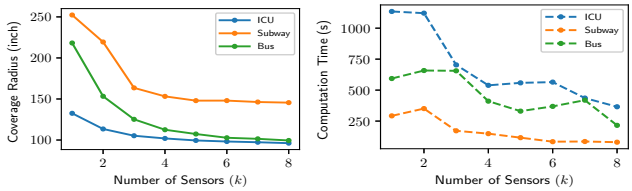


Fig. 6: Coverage quality (lower is better) and computation time for Problem II.2 for the three environments, as sensors increase.

Our last benchmark on Problem II.2 tests the effectiveness of the local improvement following the resolution of a coarsely generated ILP, at 60 candidate sensor locations and 1000 surface samples (Fig. 7). The right figure shows much faster computation time. The left figure shows that the faster method does a decent job for the bus environment (other environments have similar outcomes). The result suggests which method to use would depend on whether computational time or solution optimality is more important to the task at hand. We note that the local improvement method does not help improve the ILP result at the higher resolution.

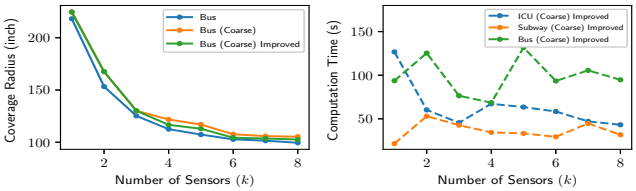


Fig. 7: [left] Solution quality (lower is better) for the bus environment. The first curve (Bus) is the same as that from Fig. 6. [right] Computation time using coarse ILP + local improvement.

For Problem II.3, computation becomes more demanding. At the specified discretization level, most ILP models did not complete the optimization process in 10 minutes. The intermediate quality result is given in Fig. 8, on the left (the same threshold, selected to make the computation challenging, is used for all three environments), where the lines corresponds to the coverage ratio returned by the ILP model and the attached vertical bars show the reported optimality gap. The crosses show the updated ratio after local improvement is carried out (the triangles will be explained shortly). The subway data was shifted to the left to improve readability. For the bus, we see that the local improvement does help improve solution optimality, suggesting it is the most difficult problem. For the other two, it appears that the solution by the ILP model is already quite optimal, but the ILP solver still needs time to close the gap from the above. The subway case has worse coverage by the same number of sensors because it is larger. If we run a coarse ILP model (60 sensor candidates, 1000 surface sample) for one minute and then do local improvement, we get coverage ratios shown as the triangles in Fig. 8, left. Fig. 8, right shows the total computation time used. We observe that except for the challenging bus model, the faster method achieves essentially identical optimality as running ILP at higher resolutions. Subway costs most time here because it is the largest.

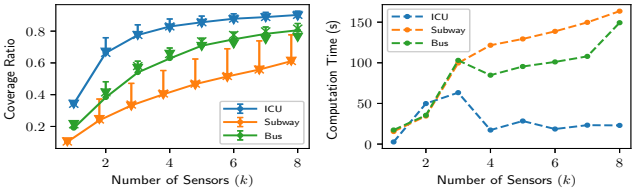


Fig. 8: [left] Coverage ratio (lines) for Problem II.3 returned by multiple methods. [right] Computation time used by running a coarse ILP plus local improvements.

Lastly, we provide some additional visualization to help further demonstrate the structure of the problems. Fig. 9 shows that Problems II.2 and II.3 induce different optimal distribution of sensors. Generally, Problems II.2 tends to cause the sensors to be evenly spaced out. On the other hand, Problem II.3 tends to balance between spacing out sensors and provide good cumulative coverage, which may require sensors to aggregate, which can be observed in Fig. 10.

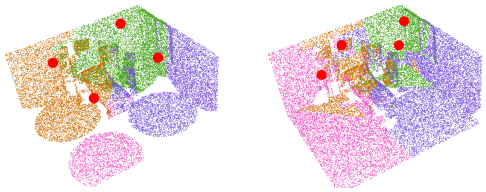


Fig. 9: 4 sensor ICU result for Problems II.2 (left) and II.3(right).

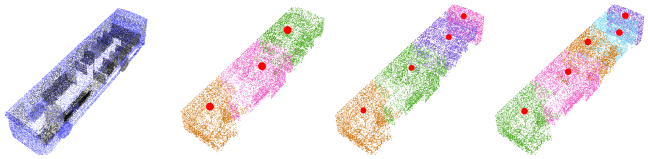


Fig. 10: The coverage of bus (see through model on the left) using 3 to 5 sensors under Problem II.3. Aggregation of sensors at the front of the bus, which is structurally more complex, can be observed.

V. Conclusion

We have formulated a general Sensor Placement for Optimal Coverage (SPOC) problem with three concrete instantiations, each with distinct practical applications. We provide near-optimal methods for solving these challenging optimization problems and demonstrated their effectiveness with extensive evaluations. We are currently exploring real-world applications of our methods.

REFERENCES

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

[2] T. C. Shermer, "Recent results in art galleries (geometry)," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1384–1399, 1992.

[3] J. O'Rourke, "Visibility," in *Handbook of discrete and computational geometry, 3rd Ed.*, 2017, pp. 875–896.

[4] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[5] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics & Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[6] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[7] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[8] M. Schwager, D. Rus, and J.-J. Slotine, "Decentralized, adaptive coverage control for networked robots," *The International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, 2009.

[9] G. A. Hollinger and G. S. Sukhatme, "Sampling-based motion planning for robotic information gathering." in *Robotics: Science and Systems*, vol. 3, no. 5. Citeseer, 2013.

[10] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[11] D. Lee and A. Lin, "Computational complexity of art gallery problems," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 276–282, 1986.

[12] A. Thue, *Über die dichteste Zusammenstellung von kongruenten Kreisen in einer Ebene.* Christiania : Dybwad in Komm., 1910.

[13] T. C. Hales, "A proof of the kepler conjecture," *Annals of Mathematics*, vol. 162, no. 3, pp. 1065–1185, 2005.

[14] Z. Drezner, *Facility Location: A Survey of Applications and Methods*. Springer Verlag, 1995.

[15] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Equitable partitioning policies for robotic networks," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2356–2361.

[16] A. Pierson, L. C. Figueiredo, L. C. Pimenta, and M. Schwager, "Adapting to sensing and actuation variations in multi-robot coverage," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 337–354, 2017.

[17] M. Schwager, B. J. Julian, and D. Rus, "Optimal coverage for multiple hovering robots with downward facing cameras," in *Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3515–3522.

[18] A. Weber, *Theory of the Location of Industries*. University of Chicago Press, 1929.

[19] S. Har-Peled, *Geometric Approximation Algorithms*. American Mathematical Soc., 2011, no. 173.

[20] T. Feder and D. Greene, "Optimal algorithms for approximate clustering," in *Proceedings ACM Symposium on Theory of Computing*. ACM, 1988, pp. 434–444.

[21] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, 1985.

[22] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[23] M. S. Daskin, "A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results," *Communications of the Operations Research Society of Japan*, vol. 45, no. 9, pp. 428–436, 2000.

[24] M. I. Shamos and D. Hoey, "Closest-point problems," in *16th Annual Symposium on Foundations of Computer Science (FOCS 1975)*. IEEE, 1975, pp. 151–162.

[25] S. W. Feng, S. D. Han, K. Gao, and J. Yu, "Efficient algorithms for optimal perimeter guarding," in *Robotics: Sciences and Systems*, 2019.

[26] S. W. Feng and J. Yu, "Optimal perimeter guarding with heterogeneous robot teams: complexity analysis and effective algorithms," *IEEE Robotics and Automation Letters*, 2020.

[27] ——, "Optimally guarding perimeters and regions with mobile range sensors," in *Robotics: Sciences and Systems*, 2020.

[28] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *28th Annual Symposium on Foundations of Computer Science (FOCS 1987)*. IEEE, 1987, pp. 49–60.

[29] V. V. Vazirani, *Approximation Algorithms*. Springer, 2013, ch. 5.

[30] P. Alliez, S. Tayeb, and C. Wormser, "3d fast intersection and distance computation (aabb tree)," in *CGAL User and Reference Manual*, 5.1 ed. CGAL Editorial Board, 2020.

[31] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *New Results and New Trends in Computer Science*. Springer, 1991, pp. 359–370.

[32] M. Berg, de, O. Cheong, M. Kreveld, van, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer, 2008, ch. 4.7.

[33] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com