Directional Sensor Planning for Occlusion Avoidance

Jake Gemerek, *Member, IEEE*, Bo Fu, *Member, IEEE*, Yucheng Chen, Zeyu Liu, Min Zheng, David van Wijk, and Silvia Ferrari, *Senior Member, IEEE*

***Abstract*—Directional sensors, such as video cameras, have become ubiquitous to many autonomous robots applications, such as monitoring and surveillance. The performance of these sensors and processing algorithms, however, may be hindered by the presence of objects that block visibility. This article presents a novel approach for planning the path of a mobile directional sensor deployed to observe multiple targets distributed in an environment populated with multiple obstacles and occlusions. Unlike existing art gallery or watchman's route methods, the visibility theory and motion planners developed in this article account for both line-of-sight visibility and bounded field-of-view constraints, and can provide obstacle avoidance based on robot geometry and kinodynamic constraints. The computational complexity analysis and experiments on a camera-equipped drone demonstrate that, despite the challenging geometric characteristics of directional C-targets, the approach scales to real-world problems. Furthermore, when compared to algorithms inspired by traveling salesman and target coverage approaches, the directional visibility planners presented in this article are significantly more effective both at guaranteeing complete target visibility and at minimizing distance traveled.**

***Index Terms*—Avoidance, camera, coverage, directional, line of sight (LOS), obstacle, occlusion, path planning, sensor, visibility, target.**

I. Introduction

DIRECTIONAL sensors are characterized by a preferred sensing direction that may be interrupted by the presence of occlusions [1]. Examples include but are not limited to monocular cameras [2], [3], stereo cameras [4], [5], radar [6], active sonar [7], [8], neuromorphic vision sensors [9], [10], and many other imaging sensors. When the sensor position and orientation is known, the visibility of a target at a known location and in the presence of potential occlusions can be established by checking the line-of-sight (LOS) visibility requirement. Directional sensors are typically characterized by a bounded field of view (FOV) that limits the range and opening angle of its visibility region. Therefore, when the sensor is installed on a mobile robot, its position and orientation along the robot path can greatly influence the quality of the images. Planning an effective path is especially important when targets must be observed repeatedly over time, e.g., in demining [7], [11] and monitoring of gas pipelines and marine structures [8], [12], [13], or when collisions and occlusions present a safety hazard, e.g., in autonomous vehicles [14], digital agriculture [15], and monitoring of indoor or urban environments [2], [16].

Methods that account for LOS visibility include solutions to the art gallery problem (AGP) [17] and the watchman's route problem (WRP) [18]. AGP solutions seek to determine the minimum number of stationary omnidirectional cameras required to simultaneously view a set of targets in a polygonal workspace characterized by occlusions [17]. Because AGP solution approaches do not obtain visibility representations in closed form, they are not applicable to mobile sensors and do not take into account bounded FOVs or obstacles to be avoided by the robot on which the sensor is installed. WRP methods check the LOS requirement while searching for the shortest path that maps the entire workspace by following the gradient of visibility-level set functions [18]. Although relevant to the LOS requirement used here, WRP and AGP methods are not applicable, nor scale up to mobile sensors observing a set of point targets, and cannot be modified to account for the bounded geometry of the FOV.

The new visibility theory and sensor path planning algorithms developed in this article allow us to take into account both LOS and bounded FOV requirements while optimizing the distance traveled by the mobile sensors and avoiding collisions with obstacles. The approach is demonstrated on a benchmark problem that consists of observing a set of targets located at known and fixed positions in the sensor workspace. The construction of visibility regions presented in this article is obtained by extending the geometric concepts known as C-targets [19]–[21] and coverage cones [16], [22] to directional sensors. Using the coverage cone and convex hull of the occlusion, a directional visibility region can be obtained in closed form as a function of the target and sensor positions (see Section III), excluding a so-called "shadow region" that precludes LOS visibility inside the FOV.

The resulting visibility regions are shown to amount to subsets of the free configuration space that are possibly intersecting, concave, and may exhibit holes. As a result, they cannot be utilized by existing planning algorithms (e.g., [3], [19], [20], [23]–[26]). A novel approach for constructing a directional visibility graph

Manuscript received 10 November 2021; revised 29 April 2022; accepted 11 May 2022. Date of publication 15 September 2022; date of current version 6 December 2022. This work was supported by the Office of Naval Research under Grants N00014-17-1-2175 and N00014-19-1-2144. This article was recommended for publication by Associate Editor J. O'Kane and Editor F. Chaumette upon evaluation of the reviewers comment. *(Corresponding author: Silvia Ferrari.)*

Jake Gemerek is with the Moog Inc., Elma, NY 14059 USA (e-mail: jrg362@cornell.edu).

Bo Fu, Yucheng Chen, Zeyu Liu, Min Zheng, and Silvia Ferrari are with the Laboratory for Intelligent Systems and Controls, Cornell University, Ithaca, NY 14850 USA (e-mail: bf284@cornell.edu; yc2383@cornell.edu; zl542@cornell.edu; minzheng486@gmail.com; ferrari@cornell.edu).

David van Wijk is with the Land, Air and Space Robotics Laboratory, Texas A&M University, College Station, TX 77840 USA (e-mail: dev37@cornell.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TRO.2022.3180628.

Digital Object Identifier 10.1109/TRO.2022.3180628

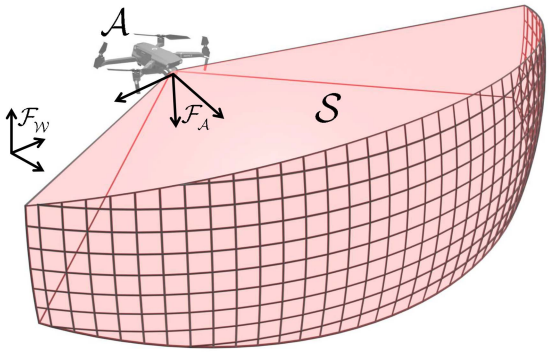


Fig. 1. Robot geometry, $\mathcal{A}$, and sensor FOV, $\mathcal{S}$, described relative to body frame $\mathcal{F}_{\mathcal{A}}$ and operating in a workspace with inertial frame $\mathcal{F}_{\mathcal{W}}$.

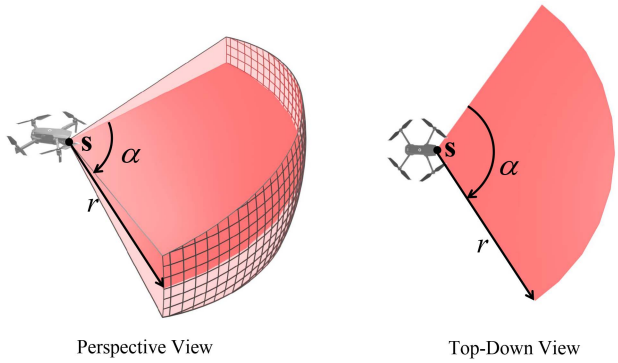


Fig. 2. Sensor FOV 2-D parameterization with apex $\mathbf{s} \in \mathcal{W}$, opening angle $\alpha \in \mathbb{S}^1$, and (maximum) range $r > 0$.

from visible target labels is presented in Section III-C. Four new directional visibility planners are developed in Section IV and analyzed in Section VI. These new methods are demonstrated and implemented on an autonomous quadrotor equipped with a monocular camera and computer vision algorithms using numerical simulations in Unreal Engine [27] and, then, in physical experiments (see Section VII-B). The results in Section VII show that the directional sensor planning algorithms developed in this article find efficient paths that avoid obstacles and view all targets, both in outdoor and indoor complex environments. Furthermore, the new directional connectivity and pruned visibility (PV) graph planners presented in Sections IV-A and IV-B are highly effective at minimizing the distance traveled, allowing the drone to find the shortest path that yields correct identification of all targets via region-based convolutional neural network (R-CNN) [28].

II. Problem Formulation and Assumptions

This article considers the problem of planning the path of a mobile directional sensor, such as a camera-equipped drone, deployed to observe multiple targets for purposes such as classification or surveillance. For illustration purposes, the robot-mounted sensor is assumed to be operating in a workspace that is closed, bounded, and 2-D, denoted by $\mathcal{W} \subset \mathbb{R}^2$. This latter assumption holds for ground robots, as well as aerial robots that can operate in steady-level flight, at an altitude commensurate to the sensor FOV (see Fig. 1). In this article, it is assumed that the 2-D workspace is populated with M opaque and solid objects, $\mathcal{B}_j \subset \mathcal{W}$, indexed by $j \in \mathcal{J}$, $\mathcal{J} = \{1, \ldots, M\}$, that constitute occlusions for the sensor and obstacles for the robot.

For simplicity, it is assumed that all M opaque objects are known, convex, polygonal, and fixed with respect to $\mathcal{W}$. For nonconvex and nonpolygonal objects, the approach can be applied by first approximating their geometry as a bounding polygonal approximation and, then, by decomposing it into convex polygonal objects, e.g., obtaining a so-called bounding rectangloid decomposition (see [20] for more details). The sensor must explore the workspace in order to observe N stationary targets (objects) of interest that are each located at a known position $\mathbf{x}_i \in \mathcal{W}$, where $i \in \mathcal{I}$, and $\mathcal{I} = \{1, \ldots, N\}$. The present formulation, which can be viewed as an extension of the AGP [17] to mobile sensors, is applicable to a number of modern surveillance systems in which environmental maps are first obtained through stationary or remote sensors and, then, a mobile sensor is deployed on site to obtain additional target information, such as action/object classification (see Section VII).

The sensor is mounted on a mobile robot, such as a ground, aerial, or underwater robot, with a rigid geometry $\mathcal{A} \subset \mathcal{W}$. Then, every point in $\mathcal{A}$ and every point in the sensor FOV, $\mathcal{S} \subset \mathcal{W}$, can be represented by the position and orientation of the moving body frame $\mathcal{F}_{\mathcal{A}}$ relative to an inertial frame $\mathcal{F}_{\mathcal{W}}$ embedded in $\mathcal{W}$, with origin $\mathcal{O}_{\mathcal{W}}$. Assume that $\mathcal{F}_{\mathcal{A}}$ is embedded in $\mathcal{A}$ with origin $\mathcal{O}_{\mathcal{A}}$ at the sensor location (e.g., camera pinhole), as illustrated in Fig. 1. The sensor is assumed to be fixed with respect to $\mathcal{A}$, such that the sensor position $\mathbf{s} \in \mathcal{W}$ and orientation $\theta \in \mathbb{S}^1$ are represented by the robot configuration, defined as $\mathbf{q} \triangleq [\mathbf{s}^T \quad \theta]^T$.

Directional sensors, such as monocular or stereo cameras, or active sonar and radar, are characterized by a preferred sensing direction and, thus, are only able to observe a target if it satisfies LOS visibility and is inside the sensor FOV defined as follows.

Definition II.1 (FOV): For a sensor characterized by configuration $\mathbf{q} = [\mathbf{s}^T \quad \theta]^T$, the sensor FOV is a rigid object that can be described by a closed and bounded subset $\mathcal{S}(\mathbf{q}) \subset \mathcal{W}$ in which the sensor may obtain target measurements.

In this article, the sensor FOV $\mathcal{S}$ is modeled by a sector with opening angle $\alpha \in [0, 2\pi)$ and radius $r > 0$, representing the sensor aperture and maximum range, respectively. The sector apex coincides with the sensor position (e.g., camera pinhole), $\mathbf{s} \in \mathcal{W}$, as shown in Fig. 2. Target LOS visibility, defined in the following, requires that, in addition to being inside the FOV, the target is free of occlusions caused by opaque obstacles ($\mathcal{B}_j$), as schematized in Fig. 3.

Definition II.2 (LOS): Given an opaque object $\mathcal{B}_j \subset \mathcal{W}$, a target at $\mathbf{x} \in \mathcal{W}$ is in the LOS of a directional sensor with apex at $\mathbf{s} \in \mathcal{W}$ if and only if

$$L(\mathbf{s}, \mathbf{x}) \cap \mathcal{B}_j = \emptyset \tag{1}$$

where $L(\mathbf{s}, \mathbf{x}) \triangleq \{(1 - \gamma)\mathbf{s} + \gamma\mathbf{x} | \gamma \in [0, 1]\}$ is a line segment connecting the sensor position, $\mathbf{s}$, to the target at $\mathbf{x}$.

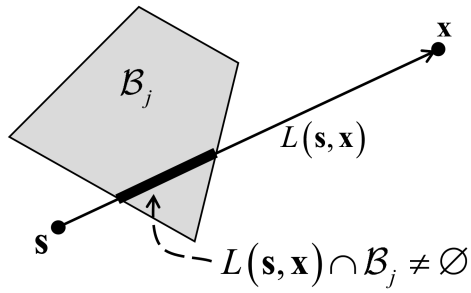


Fig. 3. Opaque object $\mathcal{B}_j$ occludes a target at $\mathbf{x}_i$ for a given sensor apex, $\mathbf{s}$.

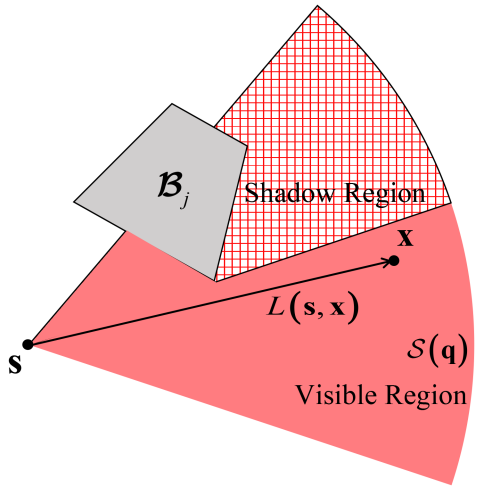


Fig. 4. Example of visible target, $\mathbf{x}$, in the presence of one opaque obstacle, $\mathcal{B}_j$, occluding a portion of the sensor FOV, $\mathcal{S}$.

Therefore, given a sensor with apex $\mathbf{s}$ and FOV $\mathcal{S}(\mathbf{q})$, a target positioned at $\mathbf{x}$ is said to be *visible* if and only if $\mathbf{x}\in\mathcal{S}(\mathbf{q})$ and $L(\mathbf{s},\mathbf{x})\cap\mathcal{B}_j=\emptyset$ for all $j\in\mathcal{J}$ (see Fig. 4). For a mobile sensor, the configuration space is the space $\mathcal{C}\cong SE(2)$ of all configurations $\mathbf{q}\in\mathcal{C}$, defined with respect to $\mathcal{F}_{\mathcal{W}}$. As a result, sensor planning consists of deciding the sequence of robot configurations that avoids obstacle collisions for robot $\mathcal{A}(\mathbf{q})$, and enables target visibility for sensor $\mathcal{S}(\mathbf{q})$. The sensor's configuration space $\mathcal{C}$ is homeomorphic to the Special Euclidean group, SE(2), meaning that the configuration space can be represented as any space with the same topological properties as SE(2) [30]. Thus, the configuration space is equivalently represented as $\mathcal{C}\cong\mathcal{W}\times\mathbb{S}^1$, and, because $\mathbb{S}^1$ is 2π-periodic and multiply connected, the configuration space is also 2π-periodic and multiply connected in the direction of the sensor rotation angle.

Let the topology of $\mathcal{C}$ be induced by the distance metric

$$D(\mathbf{q},\mathbf{q}')\triangleq\left[(\mathbf{q}-\mathbf{q}')^T\mathbf{W}(\mathbf{q}-\mathbf{q}')\right]^{1/2} \tag{2}$$

where

$$\mathbf{W}\triangleq\begin{bmatrix}w_t & 0 & 0\\ 0 & w_t & 0\\ 0 & 0 & w_r\end{bmatrix}$$

and $w_t>0$ and $w_r\geq 0$ are translational and rotational weights, respectively. Note that when $w_t=w_r=1$, the distance metric in (2) reduces to the Euclidian metric in $\mathbb{R}^3$. Having defined the topology of $\mathcal{C}$, the collision-free sensor path from an initial to a final configuration, denoted by $\mathbf{q}_0$ and $\mathbf{q}_f$, respectively, is obtained from the classic definition of a path in a topological space [31], as follows.

Definition II.3 (Path): A path from $\mathbf{q}_0$ to $\mathbf{q}_f$ is a continuous map

$$\tau:[0,1]\rightarrow\mathcal{C}_{\text{free}}$$

with $\tau(0)=\mathbf{q}_0$ and $\tau(1)=\mathbf{q}_f$.

The sensor motion is governed by an ordinary differential equation that models the robot kinematic and dynamic constraints as follows:

$$\dot{\mathbf{q}}(t)=\mathbf{f}[\mathbf{q}(t),\mathbf{u}(t),t],\quad \mathbf{q}(t_0)=\mathbf{q}_0 \tag{3}$$

where $\mathbf{u}(t)\in\mathcal{U}$ is the control vector, and $\mathcal{U}\subset\mathbb{R}^m$ is the m-dimensional space of admissible control inputs. The vector function $\mathbf{f}(\cdot)$ is obtained from a model of the robot that, for simplicity, in this article is assumed to obey single-integrator dynamics, such that $\mathbf{f}[\cdot]=\mathbf{u}(\cdot)$. Then, letting the sensor path (τ) be described by a continuous piecewise-linear path, known as a *polygonal chain*, with vertices $(\mathbf{q}_0,\mathbf{q}_1,\ldots,\mathbf{q}_n)$, the path length can be computed using the distance metric in (2) as follows:

$$J(\tau)\triangleq\sum_{k=0}^{n-1}D(\mathbf{q}_k,\mathbf{q}_{k+1}) \tag{4}$$

The problem considered in this article consists of finding the minimum-distance obstacle-free path such that every target in $\mathcal{W}$ is visible, at minimum, at one sensor configuration, i.e.,

$$\min_{\tau}\ J(\tau) \tag{5}$$

$$\text{sbj to } \mathbf{x}_i\in\mathcal{S}(\tau(\gamma))\wedge L(\tau(\gamma),\mathbf{x}_i)\cap\mathcal{B}_j=\{\emptyset\} \tag{6}$$

$$\forall i\in\mathcal{I},\forall j\in\mathcal{J},\forall\gamma\in[0,1],\tau(\gamma)\in\mathcal{C}_{\text{free}} \tag{7}$$

where $\wedge$ is the conjunction operator. When the goal is to minimize time or energy consumption, the above problem can be modified to incorporate new objective functions along with robot dynamic constraints that properly reflect speed, acceleration, and power consumption, using the optimal control approach reviewed in [1].

III. Visibility Theory

This article presents a novel theoretical framework for directional sensor planning that accounts for both LOS and bounded FOV visibility constraints in closed form. Existing robot planning methods for obstacle avoidance consider the set of all configurations that cause collisions between the robot $\mathcal{A}$ and an obstacle $\mathcal{B}_j$, by mapping $\mathcal{B}_j$ into the robot configuration space, obtaining a C-obstacle defined as $\mathcal{CB}_j\triangleq\{\mathbf{q}\in\mathcal{C}|\mathcal{B}_j\cap\mathcal{A}(\mathbf{q})\neq\emptyset\}$ [30], [32]. As a result, the robot must travel in the free configuration space, $\mathcal{C}_{\text{free}}\triangleq\mathcal{C}\setminus\bigcup_{j=1}^{M}\mathcal{CB}_j$, which represents the subset of $\mathcal{C}$ comprised of collision-free configurations. Similarly,

existing sensor planning methods account for the bounded sensor FOV $\mathcal{S}$ by mapping targets into the robot configuration space obtaining C-targets, which, for point targets, can be defined as $\mathcal{CT}_i = \{\mathbf{q} \in \mathcal{C} | \mathbf{x}_i \in \mathcal{S}(\mathbf{q})\}$ [20], [21].

This article presents a new approach that takes into account both the bounded sensor FOV and the LOS visibility constraints by constructing new directional visibility regions using convexity theory. These visibility regions represent all sensor configurations that enable target visibility in the presence of occlusions comprised of opaque objects inside the sensor FOV. Because these regions are concave and, often, enable observations from multiple targets simultaneously, existing planners (reviewed in [33]) are not applicable. Instead, new constructs referred to as "set visibility regions" are obtained and, then, used to develop four new path planning methods (see Section IV) that provide approximate solutions to the directional sensor planning problem in (5)–(7).

A. Target Visibility

As a first step, visibility regions that guarantee both FOV and LOS target visibility are obtained from the C-targets and the coverage cone method first introduced in [22].

Definition III.1 (Target Visibility Region): Given a directional sensor with apex at $\mathbf{s} \in \mathcal{W}$ and FOV geometry $\mathcal{S}(\mathbf{q})$, the visibility region of a target at $\mathbf{x}_i \in \mathcal{W}$ in the presence of M opaque objects $\mathcal{B}_j$ $(j = 1, \ldots, M)$ is defined as the subset of $\mathcal{C}_{\text{free}}$ that simultaneously satisfies the FOV and LOS target visibility conditions

$$\mathcal{TV}_i \triangleq \{\mathbf{q} \in \mathcal{C}_{\text{free}} | \mathbf{x}_i \in \mathcal{S}(\mathbf{q}), \quad L(\mathbf{s}, \mathbf{x}_i) \cap \mathcal{B}_j = \emptyset \quad \forall j \in \mathcal{J}\} \tag{8}$$

where $L(\mathbf{s}, \mathbf{x}_i) \triangleq \{(1-\gamma)\mathbf{s} + \gamma \mathbf{x}_i | \gamma \in [0, 1]\}$.

The closed form expression for the target visibility region is obtained in terms of the convex hull and coverage cone, using a series of set operations described in this section. From convex analysis [34], [35], given a nonempty subset A of a Euclidian space, the "cone generated by A," denoted by cone(A), is the set of all nonnegative combinations of the elements of A. In the 2-D Euclidian space, the cone generated by the opaque object $\mathcal{B}_j$ and with origin at $\mathbf{s} \in \mathcal{W} \backslash \mathcal{B}_j$, is defined as

$$\mathcal{K}(\mathcal{B}_j, \mathbf{s}) \triangleq \{\alpha \mathbf{z} + (1-\alpha)\mathbf{s} | \alpha \in \mathbb{R}^+, \mathbf{z} \in \mathcal{B}_j\} \tag{9}$$

and referred hereon as coverage cone of $\mathcal{B}_j$ [22].

Proposition III.2: If $\mathcal{B}_j$ is a convex polyhedral object, the polyhedral coverage cone $\mathcal{K}(\mathcal{B}_j, \mathbf{s})$ generated by $\mathcal{B}_j$ with origin at $\mathbf{s}$ is also convex.

See [36] for a proof.

Now, let the convex polygon $\mathcal{B}_j$ be represented by a finite set of vertices in $\mathcal{W}$ denoted by $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$. Then, the cone

$$\text{cone}(\mathcal{B}_j, \mathbf{s}) = \left\{ \mathbf{s} + \sum_{k=1}^{K} a_k (\mathbf{z}_k - \mathbf{s}) \,\middle|\, \mathbf{z}_k \in \mathcal{B}_j, a_k \geq 0 \right\} \tag{10}$$

is a polyhedral cone, where $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ are the generators of the cone [36]. The coverage cone in (10) is also known as *tangent cone* or *cone of feasible directions* of $\mathcal{B}_j$ at $\mathbf{s}$, and represents all line transversals of $\mathcal{B}_j$ through $\mathbf{s}$. The convex hull of $\mathcal{B}_j$, defined as

$$\text{conv}(\mathcal{B}_j) \triangleq \left\{ \sum_{k=1}^{K} a_k \mathbf{z}_k \,\middle|\, \mathbf{z}_k \in \mathcal{B}_j, a_k \geq 0, \sum_{k=1}^{K} a_k = 1 \right\} \tag{11}$$

represents the intersection of all convex sets containing $\mathcal{B}_j$.

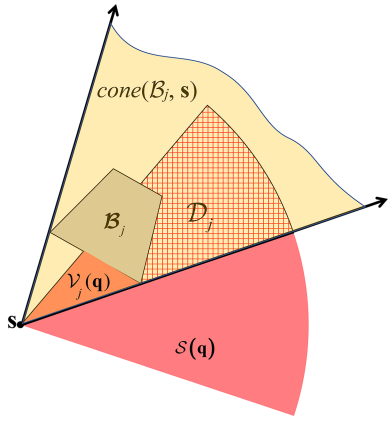


Fig. 5. Sensor shadow region (checkered color) and visibility region (orange) obtained from the sensor FOV (pink) and the obstacle coverage cone (yellow).

Now, the coverage cone and convex hull of the opaque object can be used to obtain the so-called shadow region (e.g., checkered region in Fig. 5), which represents the region of the sensor FOV occluded by the object and is defined as follows.

Definition III.3 (Sensor Shadow Region): Given a sensor FOV, $\mathbf{S}(\mathbf{q})$, modeled by a sector with apex at $\mathbf{s}$, the sensor shadow region with respect to an obstacle $\mathcal{B}_j \subset \mathcal{W}$ is defined as

$$\mathcal{D}_j(\mathbf{q}) \triangleq \{\mathbf{x} \in \mathcal{S}(\mathbf{q}) | L(\mathbf{s}, \mathbf{x}) \cap \mathcal{B}_j \neq \emptyset\} \tag{12}$$

where $L(\mathbf{s}, \mathbf{x}) \triangleq \{(1-\gamma)\mathbf{s} + \gamma \mathbf{x} | \gamma \in [0, 1]\}$.

From (12), it follows that the visibility region is the complement of $\mathcal{D}_j$ in $\mathcal{S}$

$$\mathcal{V}_j(\mathbf{q}) = \mathcal{S}(\mathbf{q}) \setminus \mathcal{D}_j(\mathbf{q}) \tag{13}$$

and represents the set of target positions that are visible to the sensor at configuration $\mathbf{q}$, as illustrated in Fig. 5.

From (10) and (11), it can be easily shown that the sensor shadow region can be obtained by the following set operations:

$$\begin{aligned} \mathcal{D}_j(\mathbf{q}) &= \{\text{cone}(\mathcal{B}_j, \mathbf{s}) \setminus \text{conv}(\mathcal{B}_j)\} \cap \mathcal{S}(\mathbf{q}) \\ &= \{\text{cone}(\mathcal{B}_j, \mathbf{s}) \cap \mathcal{S}(\mathbf{q})\} \setminus \{\text{conv}(\mathcal{B}_j \cap \mathcal{S}(\mathbf{q}))\}. \end{aligned} \tag{14}$$

Because the coverage cone and sensor FOV always have the same origin (apex), their intersection is a sector. Furthermore, the intersection of a polygonal obstacle with the sensor FOV is a convex polygon. Therefore, the above set operations can be carried out efficiently for all obstacles in the workspace, provided $\mathbf{q}$ is known.

When planning the sensor path, however, the configuration $\mathbf{q}$ is not known *a priori* and the presence of multiple targets and multiple obstacles must be taken into account simultaneously. Therefore, consider an alternate representation of the shadow region that is based on the known target position, $\mathbf{x}_i$, and

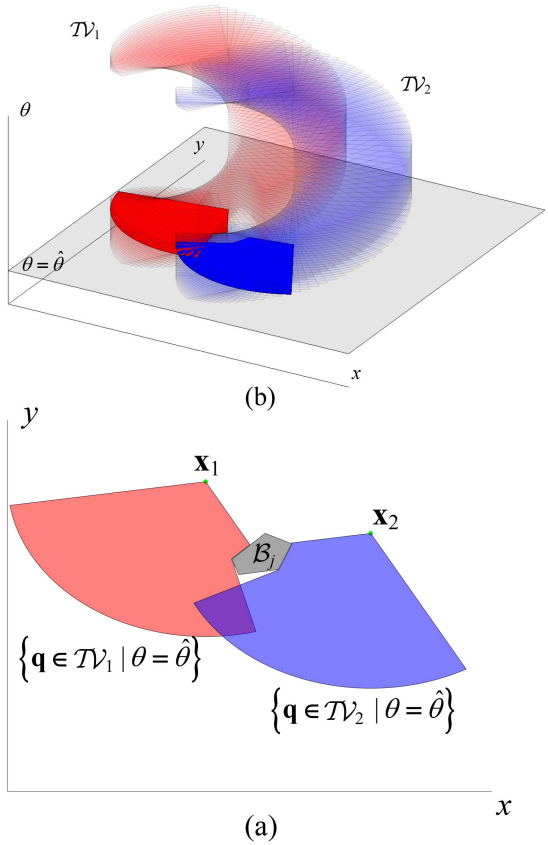


Fig. 6. (a) Visibility regions of two targets $\mathbf{x}_1$ and $\mathbf{x}_2$ at a fixed orientation $\hat{\theta}$. (b) Top-down view of target visibility regions in configuration space.

represents all sensor configurations at which target i is occluded by obstacle $\mathcal{B}_j$ despite being located inside the sensor FOV $\mathcal{S}(\mathbf{q})$, i.e.,

$$\mathcal{D}(\mathbf{x}_i) = \bigcup_{j \in \mathcal{J}} (\text{cone}(\mathcal{B}_j, \mathbf{x}_i) \setminus \text{conv}(\mathcal{B}_j \cup \{\mathbf{x}_i\})) \triangleq \mathcal{D}_i \quad \forall i \in \mathcal{I}. \tag{15}$$

The above *target shadow region* is independent of the sensor rotation and, thus, can be mapped from $\mathcal{W}$ to $\mathcal{C}$ as follows:

$$\hat{\mathcal{D}}_i = \{\mathbf{q} = [\mathbf{s}^T \quad \theta]^T \in \mathcal{C} | \mathbf{s} \in \mathcal{D}_i, \theta \in \mathbb{S}^1\} \quad \forall i \in \mathcal{I}. \tag{16}$$

Then, the visibility region of a target located at $\mathbf{x}_i$ can be computed from its shadow region and C-target

$$\mathcal{TV}_i = \mathcal{CT}_i \setminus \hat{\mathcal{D}}_i \quad \forall i \in \mathcal{I} \tag{17}$$

reintroducing the region's dependency on the sensor orientation (θ).

In practice, the rotational component of the configuration space $\theta \in [0, 2\pi)$ is discretized into $\kappa > 0$ linearly spaced intervals and the C-target and shadow region are computed as planar geometries. Examples of target visibility regions are shown in Fig. 6, where Fig. 6(a) shows two planar visibility regions at a sample fixed rotation $\theta = \hat{\theta}$ and Fig. 6(b) shows the two visibility regions in configuration space for the two target positions and occlusion plotted in Fig. 6(a). By discretizing the rotational component of the configuration space into a collection of κ 2-D manifolds, existing computational geometry tools (such as [37]) can be applied to determine the visibility regions of all N targets in the presence of M obstacles.

B. Set Visibility

Unlike C-targets [20], target visibility regions are typically concave even for point targets and convex sensor FOVs. Furthermore, for dense target environments and/or long-range sensors, multiple visibility regions may intersect giving rise to valuable subsets of the free configuration space that enable simultaneous observations from multiple targets. As a result, by visiting these regions, the robot is able to avoid occlusions and obtain measurements (e.g., images) from multiple targets that appear simultaneously inside its FOV. For this reason, after computing the visibility regions $\mathcal{TV}_i$ $(i = 1, \ldots, N)$ for all known targets in $\mathcal{W}$, it is convenient to determine their intersections as explained in this section.

For N target visibility regions that are not mutually disjoint, every region of intersection can be associated with an index set that represents the indices of all targets visible from the corresponding set of sensor configurations. Then, a one-to-one correspondence can be established between target sets and visibility by introducing the following definition.

Definition III.4 (Set Visibility Region): Given a set of N target visibility regions $\{\mathcal{TV}_i | i \in \mathcal{I}\}$, let $P \subseteq \mathcal{I}$ denote the set of indices of two or more intersecting regions, $\mathcal{TV}_{i_1} \cap \mathcal{TV}_{i_2} \cap \cdots \cap \mathcal{TV}_{i_n} \neq \emptyset$, such that $P = (i_1, \ldots, i_n)$. Then, the set visibility region is defined as

$$\mathcal{V}_P \triangleq \{\mathcal{TV}_{i_1} \cap \cdots \cap \mathcal{TV}_{i_n} | P = (i_1, \ldots, i_n), i_1, \ldots, i_n \in \mathcal{J}\} \tag{18}$$

where P is an ordered tuple of visible targets' indices.

A closed-form expression for the set visibility region is obtained as follows:

$$\mathcal{V}_P = \left\{ \bigcap_{i \in P} \mathcal{TV}_i \right\} \setminus \left\{ \bigcup_{i \notin P} \mathcal{TV}_i \right\} \tag{19}$$

and, under the assumptions in Section II, consists of a 3-D subset of the free configuration space $\mathcal{C}_{\text{free}}$. The 2-D cross sections of set visibility regions can be obtained by considering the intersections of target visibility regions at a constant sensor orientation [as can be seen in Fig. 6(a)].

An example of the closed-form set visibility region in (19) is illustrated in Fig. 7, where six set visibility regions are obtained in a workspace with three targets at positions $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$, and two opaque objects $\mathcal{B}_1$ and $\mathcal{B}_2$. The three target visibility regions are obtained at a fixed sensor orientation ($\hat{\theta}$), and their intersections lead to six set visibility regions labeled by the index set of the visible targets. In this example, the sensor configurations in $\mathcal{V}_{\{1,2,3\}}$ are the most valuable because they enable simultaneous observation of all three targets. Also, the example illustrates why visibility regions are typically concave and, possibly, disconnected (e.g., see $\mathcal{V}_{\{3\}}$ in Fig. 7). As a result, existing sensor planning methods [1] are not directly applicable.

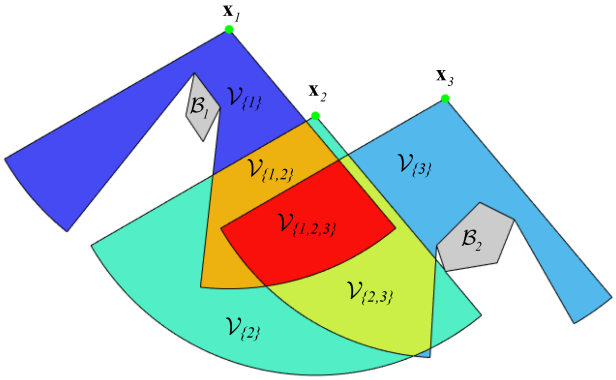


Fig. 7. Set visibility regions of three targets $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ (green dots), in the presence of two occlusions $\mathcal{B}_1$ and $\mathcal{B}_2$ (grey polygons) at a fixed sensor orientation $\hat{\theta}$.

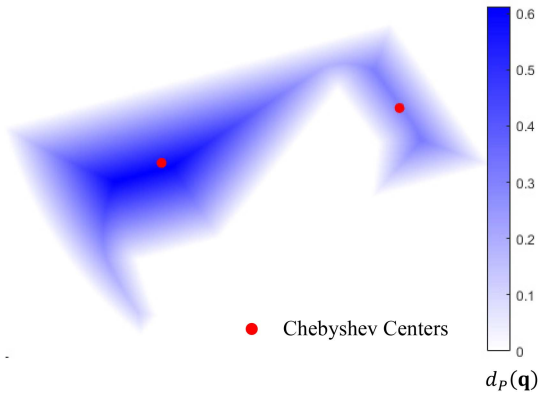


Fig. 8. Boundary-distance function (color bar) and Chebyshev center of a concave set visibility region, $\mathcal{V}_{\{1\}}$ obtained for the example in Fig. 7.

As shown in the next section, the target and set visibility regions can be used to produce a new type of connectivity graph that captures parsimoniously both target visibility and the connectivity of the free space. Four new planning algorithms are then presented in Section IV that offer computationally tractable solutions to the directional sensor planning problem in (5)–(7).

C. Directional Connectivity Graph (DCG)

Inspired by traditional connectivity graphs for robot planning [20], [33], [38], the new DCG presented in this section seeks to capture the free-space connectivity between sensor configurations that are characterized by visible target information. In order to guarantee the inclusion of all targets while maintaining the representation tractable, the graph is grown incrementally from samples obtained from the set visibility regions. The nodes of the graph are then labeled with the index set associated with the generating region so that the search for the optimal path can continue until it is guaranteed to cover all N targets.

The DCG, $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, is comprised of a set of nodes, $\mathcal{N}$, and a set of undirected edges or arcs, $\mathcal{E}$. Similarly to traditional connectivity graphs, a node $n_l \in \mathcal{N}$ represents an obstacle-free robot configuration $\mathbf{q}_l \in \mathcal{C}_{\text{free}}$, and an arc $(n_l, n_p) \in \mathcal{E}$ is placed between two nodes, $n_l, n_p \in \mathcal{N}$, if there exists a (straight) obstacle-free path between them. The arc (n_l, n_p) is then labeled by the distance $D(\mathbf{q}_l, \mathbf{q}_p)$, where the metric D is defined in (2).

Unlike traditional connectivity graphs [1], in the DCG, every node, say n_l, is labeled by the index set P of the set visibility region that contains $\mathbf{q}_l$, i.e., $\mathbf{q}_l \in \mathcal{V}_P$. Furthermore, multiple configurations are sampled from each set visibility region in order to improve connectivity as well as path efficiency. The set of nodes, $\mathcal{N}$, is grown incrementally starting from an initial set of configurations that are representative of the geometries of these regions. Because the set visibility region often is concave, its centroid may lie outside the region and, thus, may not allow any of the targets to be visible to the sensor.

Therefore, the set of configurations representative of the set visibility region $\mathcal{V}_P$ is obtained from the set of all local maxima of the boundary-distance function corresponding to the Chebyshev centers, namely,

$$\mathcal{Q}_P = \left\{ \mathbf{q}_l^* \in \mathcal{V}_P | \mathbf{q}_l^* = \arg\max_{\mathbf{q} \in \mathcal{V}_P} d_P(\mathbf{q}), \|\mathbf{q} - \mathbf{q}_l^*\| \leq c \right\} \tag{20}$$

where

$$d_p(\mathbf{q}) \triangleq \min_{\boldsymbol{\xi} \in \partial \mathcal{V}_P} \|\boldsymbol{\xi} - \mathbf{q}\| \tag{21}$$

$c > 0$ is a user-defined margin, and l is a finite positive integer. A Chebyshev center or "incenter" is defined as the center of the largest inscribed circle and, for a concave polygon, requires solving a nonconvex optimization problem [35]. As an example, the boundary-distance function and the two Chebyshev centers of a concave set visibility region $\mathcal{V}_{\{1\}}$ are plotted in Fig. 8, for the example at a fixed sensor orientation shown in Fig. 7. It can be seen that the Chebyshev centers are the local maxima of the boundary distance function.

In building the DCG, all of the Chebyshev centers of every set visibility region, found from (20), are added to set of nodes $\mathcal{N}$, and each node (center) is labeled by the accompanying index set P. Then, the sensor's initial and final configurations, $\mathbf{q}_0$ and $\mathbf{q}_f$, labeled by their two index sets of visible targets are added to $\mathcal{N}$. Finally, in order to guarantee connectivity, some nodes are sampled from the rest of the free configuration space $\mathcal{C}_{\text{free}}$ using a roadmap method [24] and labeling each with an empty index set to indicate the absence of visible targets.

IV. Directional Visibility Planners

The problem of finding the shortest directional sensor path visiting all target visibility regions can now be viewed as a generalized traveling salesman problem (GTSP) [39], [40], also known as group TSP [41] or one-of-a-set TSP [42]. In particular, the GTSP that arises from the directional sensor planning problem (5)–(7) is characterized by continuous regions, also known as *neighborhoods*, that are typically concave, intersecting, and disconnected. Therefore, existing GTSP algorithms, including

the growing self-organizing array (GSOA) algorithm [43], the hybrid random-key genetic algorithm [44], or its later modifications which rely on the regions' centroids [45], are not effective at finding the optimal path (see Section VII).

The new planners developed in this section overcome the difficulties associated with GTSPs characterized by disconnected and intersecting neighborhoods by exploiting the set visibility regions and DCG presented in Section IV-A. Many graph-search algorithms were investigated for this purpose. Search tree pruning, a variant of label-correcting algorithms, was first adopted (see Section V) because of its guarantee that the shortest path will be found, if one exists [46]. However, because it requires storing every unpruned branch, this algorithm runs out of memory for real-world-size applications (see Section V). The second class of graph algorithms investigated is known as mixed-integer linear programming with constraints [47], including constraint generation [48], branch-and-bound [47], and branch-and-cut [49]. However, it was found that, as the workspace complexity and number of targets increase, the tree generated by the branching procedures becomes too large, causing the algorithm to run out of memory before terminating.

The metaheuristic method known as Monte Carlo Tree Search (MCTS), presented in [50], was modified to search the DCG, as explained in Appendix A, and found to provide the best performance thanks to its tradeoff of local versus stochastic search, which enabled efficient explorations of the most promising regions of the graph, $\mathcal{G}$. The advantages of MCTS over other algorithms are that it overcomes the curse of dimensionality (see Section VI-A) and is characterized by a bounded expected cumulative regret. In particular, it can be shown that the expected deviation of the approximate MCTS solution from the optimal path in $\mathcal{G}$ is bounded by $O(\log(n))$, where n is the number of graph nodes along the path.

A. DCG Planner

The DCG planner seeks to search the directional connectivity graph derived in Section III-C for the optimal sensor path that covers all targets by imposing constraints on the union of the nodes' labels. The set visibility regions represent useful intersections that allow us to shorten the sensor path by considering a type of geometric hitting set problem in which one seeks to find a path hitting the geometric set cover of the target visibility regions [51]–[54]. Hence, the DCG search seeks to produce a piecewise continuous path, τ, as a sequence of adjacent nodes or *channel*

$$C = \{n_0, n_1, \ldots, n_n\}, \quad (n_l, n_{l+1}) \in \mathcal{E} \quad \forall l \tag{22}$$

starting at the initial configuration $\mathbf{q}_0$ and covering all target labels.

The cost associated with the channel is defined in (4) as the sum of the labels attached to the arcs in the channel, namely the distance metric defined in (2). The set of targets visited by the channel is given by the union of the node labels, and therefore, the solution of the directional sensor planning problem (5)–(7)

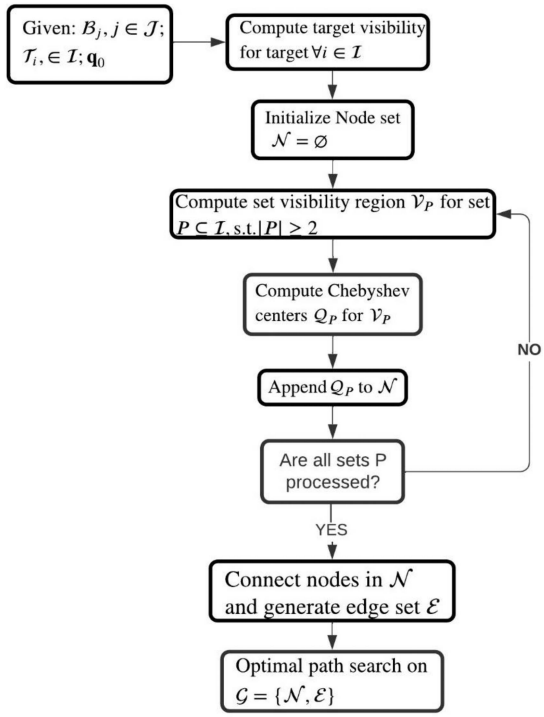


Fig. 9. Flowchart of DCG planner (see Algorithm 1).

can be found by solving the following graph optimization

$$\min_{C \in \mathcal{G}} \quad J(C) = \sum_{k=0}^{n-1} D(\mathbf{q}_n, \mathbf{q}_{n+1}) \tag{23}$$

$$\text{sbj to} \quad \bigcup_{k=0}^{n} P_k = \mathcal{I} \tag{24}$$

for the optimal channel C^*.

The pseudocode of the DCG method is shown in Algorithm 1, where the user-defined parameters are the number of rotational discretizations $\kappa > 0$ and the number of samples $\eta > 0$ used in the roadmap step. Fig. 9 shows the flowchart representation of the algorithm.

B. PV Planner

Although feasible for many real-world problems (see Section VII), the DCG approach can become computationally too burdensome for problems with a large number of targets (N). A PV method is presented in this section in an effort to reduce the computation required while taking advantage of the visibility theory presented in Section IV-A. As a first step, the approach computes the N target visibility regions in free configuration space, $\mathcal{TV}_i \subset \mathcal{C}_{\text{free}}$, as shown in (17), and projects them back onto $\mathcal{W}$ to obtain 2-D orthographic projection of the target visibility region

$$\tilde{\mathcal{T}}_i \triangleq \bigcup_{\theta \in \mathbb{S}^1} \left\{ \mathbf{x} \in \mathcal{W} | [\mathbf{x} \quad \theta]^T \in \mathcal{TV}_i \right\} \subseteq \mathcal{W} \quad \forall i \in \mathcal{I} \tag{25}$$

Algorithm 1: Directional Connectivity Graph (DCG) Planner.

Require: $\mathcal{B}_j \subset \mathcal{W}$, $\forall j \in \mathcal{J}$, $\mathbf{x}_i \forall i \in \mathcal{I}$, $\mathbf{q}_0 \in \mathcal{C}_{\text{free}}$
Parameters: $\kappa > 0, \eta > 0, \mu > 0$
Initialize: $\mathcal{N} \leftarrow \emptyset, \mathcal{E} \leftarrow \emptyset$
for $i \in \mathcal{I}$ **do**
  $\mathcal{TV}_i \leftarrow$ getTargetVisibility$(\cup_j \mathcal{B}_j, \mathbf{x}_i; \kappa)$
end for
for $P \subseteq \mathcal{I}$ **do**
  $\mathcal{V}_P \leftarrow$ getSetVisibility$(\{\mathcal{TV}_i\}_{i\in\mathcal{I}}, P)$
  $\mathcal{Q}_P \leftarrow$ getChebyshevCenters$(\mathcal{V}_P)$
  for $\mathbf{q} \in \mathcal{Q}_P$ **do**
    $\mathcal{N} \leftarrow \mathcal{N} \cup (\mathbf{q}, P)$
  end for
end for
$\mathcal{E} \leftarrow$ connectNodes$(\mathcal{N}, \cup_j \mathcal{B}_j; \eta)$
$\mathcal{G} = (\mathcal{N}, \mathcal{E})$
$C^* \leftarrow$ MonteCarloTreeSearch$(\mathcal{G}; \mu)$
return C^*

which can be viewed as a new "target" set of sensor positions in the workspace from which $\mathcal{T}_i$ is visible provided the sensor assumes the correct orientation.

Similarly to Definition III.4, the intersections of the 2-D visibility regions in (25) can be computed using (19). Then, a 2-D set visibility region, defined as

$$\tilde{\mathcal{V}}_P \triangleq \{\tilde{\mathcal{T}}_{i_1} \cap \cdots \cap \tilde{\mathcal{T}}_{i_n} | P = (i_1, \ldots, i_n), i_1, \ldots, i_n \in \mathcal{J}\} \tag{26}$$

represents the set of sensor positions from which a subset of targets, labeled by $P \subseteq \mathcal{I}$, is visible provided the sensor assumes the correct orientation.

In an effort to reduce computational complexity, redundant set visibility regions are pruned, i.e., for any pair of (nonempty) regions $\tilde{\mathcal{V}}_{P_1}$ and $\tilde{\mathcal{V}}_{P_2}$ characterized by target index sets that obey $P_1 \subset P_2$, region $\tilde{\mathcal{V}}_{P_1}$ is eliminated because all targets in P_1 are also visible from sensor positions in $\tilde{\mathcal{V}}_{P_2}$. Then, the method in Section III-C can be used to construct a *pruned connectivity graph*, denoted by $\mathcal{G}_{\text{PV}} = (\mathcal{N}_{\text{PV}}, \mathcal{E}_{\text{PV}})$, by sampling only the pruned set visibility regions, where now the set of nodes $\mathcal{N}_{\text{PV}} = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots\}$ represents a set of sensor positions, and $\mathbf{x}_0 \in \mathcal{W}$ is the initial sensor position.

Although this approach may result in longer sensor paths, it significantly reduces computational complexity while guaranteeing that all targets are represented in the connectivity graph. The results in Section VII confirm that pruning dramatically reduces the total number of set visibility regions, thereby reducing in turn the size of the connectivity graph. Because it initially ignores sensor orientation, this planner is particularly advantageous when performing robot rotations is significantly less costly than performing translations. For example, PV solutions are particularly well suited to unicycle robots or quadrotors that are able to turn "on a dime," while DCG may be more advantageous for car-like robots with a large (minimum) turning radius or for fixed-wing aircraft.

C. Cell Decomposition (CD) Planner

Sampling methods, such as DCG and PV, can be shown at best to be probabilistically complete [32]. CD, on the other hand, can guarantee completeness by obtaining a convex decomposition of the free configuration space, under proper assumptions [32]. For comparison, an approximate CD planner for directional sensors is presented in this section and analyzed in Section VI-C. Because the set visibility regions are not only concave but also possibly disconnected, an approximate CD algorithm is obtained by considering the target visibility regions computed by the approach developed in Section III-A. The configuration space is discretized by considering $\kappa > 0$ orientations, $\hat{\theta}_1, \ldots, \hat{\theta}_\kappa$. Then, holding each orientation $\hat{\theta}_u$ fixed ($u = 1, \ldots, \kappa$), a rectangloid approximation $\mathcal{RT}_i$ of the cross section of the ith target visibility region at the sensor orientation $\hat{\theta}_u$ is obtained such that it satisfies the following conditions:

$$\mathcal{RT}_i(\hat{\theta}_u) \subset \mathcal{TV}_i(\hat{\theta}_u), \quad \mathcal{TV}_i(\hat{\theta}_u) \triangleq \{\mathbf{q} \in \mathcal{TV}_i | \theta = \hat{\theta}_u\} \tag{27}$$

for all $i \in \mathcal{I}$. A rectangloid decomposition

$$\mathcal{K}(\hat{\theta}_u) = \bigcup_{i\in\mathcal{I}} \mathcal{RT}_i(\hat{\theta}_u) \tag{28}$$

can then be used to represent the N target visibility regions for every discrete orientation $\hat{\theta}_u$.

Rectangloid approximations of every C-obstacle ($\mathcal{CB}_j$), denoted by $\mathcal{RB}_j$, are similarly obtained such that $\mathcal{CB}_j(\hat{\theta}_u) \subset \mathcal{RB}_j(\hat{\theta}_u)$ for all $j \in \mathcal{J}$ and $u = 1, \ldots, \kappa$. Then, the rest of the free configuration space can be represented by decomposing the space that is free of both obstacles and target visibility regions, i.e.,

$$\mathcal{C}_{\text{void}}(\hat{\theta}_u) = C_{\text{free}}(\hat{\theta}_u) \setminus \left\{ \left(\bigcup_{j\in\mathcal{J}} \mathcal{CB}_j(\hat{\theta}_u) \right) \cup \left(\bigcup_{i\in\mathcal{I}} \mathcal{TV}_i(\hat{\theta}_u) \right) \right\}, u = 1, \ldots, \kappa \tag{29}$$

obtaining a rectangloid decomposition $\mathcal{K}_{\text{void}}(\hat{\theta}_u)$ for every discrete orientation $\hat{\theta}_u$.

Finally, the centroid of each rectangloid in the full decomposition, $\mathcal{K}(\hat{\theta}_u) \cup \mathcal{K}_{\text{void}}(\hat{\theta}_u)$, along with the corresponding orientation, $\hat{\theta}_u$, provides a sensor configuration that is represented by a node in the CD connectivity graph $\mathcal{G}_{\text{CD}} = (\mathcal{N}_{\text{CD}}, \mathcal{E}_{\text{CD}})$. Two nodes in $\mathcal{N}_{\text{CD}}$ are connected by an arc in $\mathcal{E}_{\text{CD}}$ if they are characterized by adjacent rectangloid cells with the same sensor orientation, or if the corresponding rectangloids have a nonzero intersection and the corresponding orientation angles, say $\hat{\theta}_i$ and $\hat{\theta}_j$, satisfy $|\hat{\theta}_i - \hat{\theta}_j| \leq \epsilon$, where $\epsilon > 0$ is a user-defined threshold. A schematized representation of the key differences between the novel DCG, PV, and CD planners developed in this article is illustrated in Fig. 10. A detailed performance comparison, including a study of the influence of the user-defined parameters, is provided in Section VII.

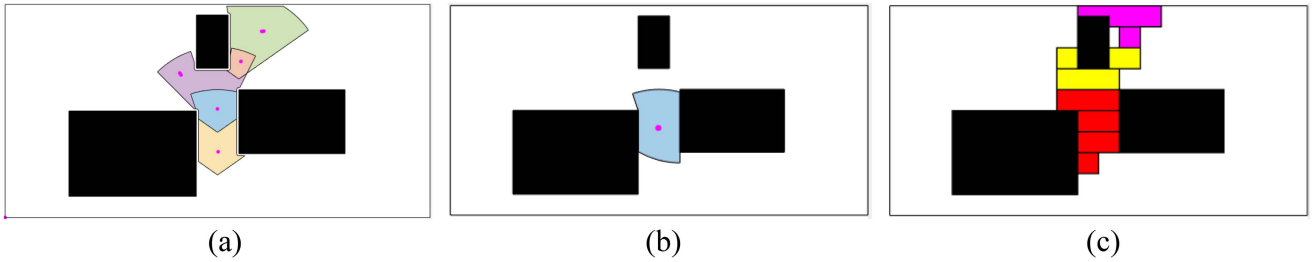


Fig. 10. Conceptual representation of (a) DCG, (b) PV, and (c) CD planners' key differences.

D. GTSP Planner

A fourth planner is developed and implemented by treating the target visibility regions as the neighborhoods of a GTSP in configuration space [45], [55]. Because even Euclidian TSPN is known to be APX-hard [56], GTSP solution algorithms all rely on heuristic approaches to reduce computational complexity [45], [57], [58]. These and other recent GTSP solution methods, reviewed and compared in [59], either reduce GTSP to a classic traveling salesman problem (TSP), inevitably increasing the problem size, or exploit restricted types of neighborhoods, such as disks or polygons. Because the target visibility regions do not obey any of these restricted geometries and typically consist of concave, disconnected, and intersecting regions, the GTSP planner developed in this section adopts a heuristic TSP solution approach.

A TSP graph, $\mathcal{G}_{\rm TSP} = (\mathcal{N}_{\rm TSP}, \mathcal{E}_{\rm TSP})$, is obtained by assigning a node to each target location in $\mathcal{W}$, and then augmenting the node set using the same roadmap sampling method used by the DCG planner (see Section IV-A). The nodes in the TSP graph are connected by first constructing a classic visibility diagram [60], [61], such that any two nodes characterized by target labels are connected by an edge if and only if they satisfy LOS visibility (see Definition II.2). As a final step, the remaining nodes are connected using a roadmap local planner that samples a uniform distribution over the workspace in order to avoid collisions with obstacles and guarantee that $\mathcal{G}_{\rm TSP}$ is fully connected. By this approach, the directional sensor planning problem (5)–(7) is transformed into a classic TSP problem [25] that can be solved using the MCTS algorithm in Appendix A to find a path that visits all N targets.

This heuristic TSP solution significantly increases the problem size even under the stated simplifying assumptions. Therefore, a tractable solution to real-world problems (see Section VII) is obtained by neglecting the sensor orientation and bounded FOV at the expense of target coverage.

V. Optimal Directional Sensor Planning Solution

When the size of the directional visibility graph allows for search-tree pruning or label-correcting algorithms to be implemented, the optimal path may be found by imposing additional search constraints on the target labels attached to the target visibility nodes. Also, in some special cases, the optimal sensor path maybe found analytically. This section shows that, by implementing the (scalable) MCTS algorithm presented in Appendix A, the DCG planner returns solutions that are approximately optimal and, yet, can be applied to the real-world problems presented in Section VII-B.

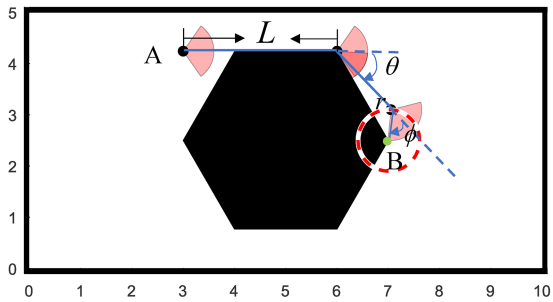


Fig. 11. Hexagonal benchmark example and optimal solution.

A. Hexagonal Benchmark Example

Consider the simple directional sensor planning problem illustrated in Fig. 11, in which the robot is at an initial configuration labeled by A, with zero heading angle, and a single target of interest is located at a position ($\mathbf{x}$) labeled by B. A single large (regular) hexagonal occlusion is present in this workspace, with edge length equal to l. The optimal (shortest) path is given by an analytical solution comprised of a straight line of length L from A to the second vertex in the direction of the target and, followed by a second straight line with a turn angle θ. Given the user-defined translational and rotational weights, w_t and w_r, in the robot distance metric (2), the optimal value of the turn angle depends on the sensor FOV and can be found by minimizing

$$J(\tau) = w_t \left\{ L + l \sin\left(\frac{\pi}{6} + \theta\right) - r\cos\phi \right\} + w_r \theta \qquad (30)$$

with respect to θ, subject to the constraint imposed by the law of sines:

$$\phi = \arcsin\left\{ \frac{l}{r} \sin\left(\frac{\pi}{3} - \theta\right) \right\} \qquad (31)$$

By substituting (31) in (30), the optimal turn radius is found by solving a simple minimization problem, such that, for $w_t = 1$ $[\mathrm{m}^{-1}]$ and $w_r = 0.1\,[\mathrm{rad}^{-1}]$, $\theta^* = 1.026\,[\mathrm{rad}]$. Then, the optimal path has a total cost $J^* = 4.5037$, while the approximately optimal path found by the DCG planner, using the MCTS algorithm, shown in Fig. 12, has a cost $J = 5.2392$.

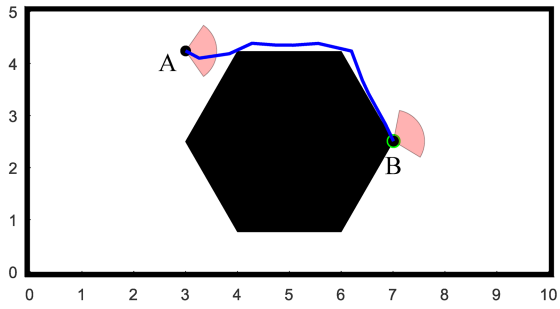


Fig. 12. Approximately optimal path obtained by the DCG planner applied to the hexagonal example in Fig. 11.

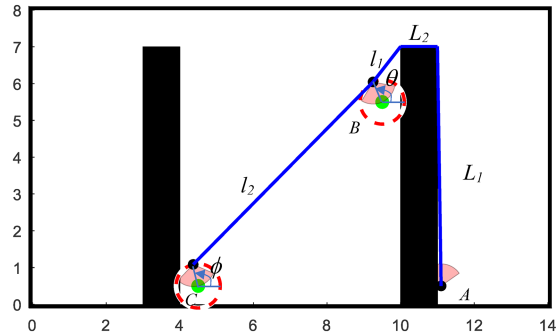


Fig. 13. Wall benchmark example and optimal solution.

B. Wall Benchmark Example

Consider the directional sensor planning problem illustrated in Fig. 13, in which the robot must observe two targets of interest, with positions ($\mathbf{x}_1$ and $\mathbf{x}_2$) labeled by B and C (green dots). The targets are both located between two walls of known dimensions and, because of the robot initial position (A), they are both occluded by the right wall. Additionally, the sensor FOV (pink sector) is bounded and small relative to the size of the workspace. Thus, the robot must arrive at a suitable position and orientation in order to observe each target. When the rotational weight w_r equals zero, an optimal path can be found analytically by minimizing the cost

$$J(\tau) = L_1 + L_2 + l_1(\theta) + l_2(\theta, \phi) \tag{32}$$

with respect to θ and ϕ.

In particular, when $w_t = 1$ [m^{-1}] (and $w_r = 0$), the optimal solution is given by $\theta^* = 4.817$ [rad] and $\phi^* = 0.500$ [rad], and is characterized by the total cost $J^* = 14.572$. In comparison, the approximately optimal path found by the DCG planner, shown in Fig. 14, has a cost $J = 16.142$ and is remarkably close to the optimal solution.

C. Search-Tree Benchmark Example

When an analytical solution cannot be found but the problem size is sufficiently small, a near-optimal solution can be found by implementing the DCG planner with an optimal graph search algorithm, such as search-tree pruning or label-correcting. In this case, under the assumptions stated in Section II, the problem

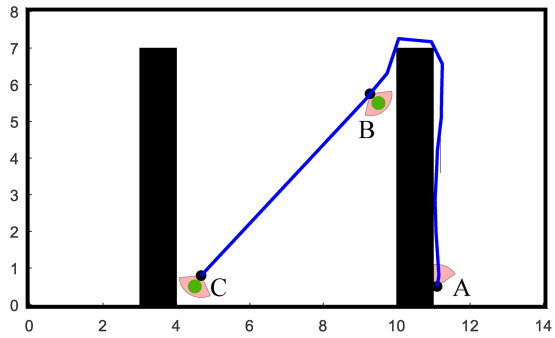


Fig. 14. Approximately optimal path obtained by the DCG planner applied to the wall example in Fig. 13.

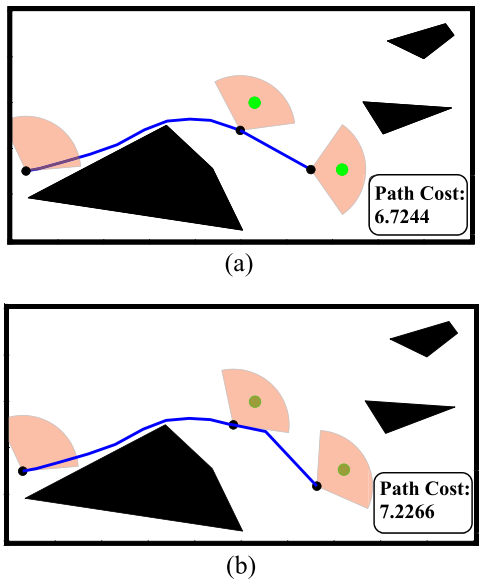


Fig. 15. Performance comparison between the solution obtained via (a) search-tree pruning and (b) MCTS.

can be solved numerically by searching the DCG, and the only approximation arises from the (rotational) discretization of the target visibility regions, defined in Section III-A. In this case, the DCG solution is nearly optimal provided the total number of discretizations, κ, is sufficiently large. For a small workspace with few targets of interest, such as the example in Fig. 15, a large κ affords a nearly optimal solution, because the search-tree algorithm is guaranteed to find the shortest path. In comparison, the DCG solution obtained by the MCTS is not guaranteed to be the optimal path in the graph, as shown by the results in Fig. 15(b), but is significantly more efficient and, therefore, is applicable to larger real-world problems.

In fact, when the search-tree pruning algorithm is applied to the solution of a problem with four obstacles/occlusions (not shown for brevity) and an increasing number of targets, the algorithm runs out of memory at $N = 5$ targets (see Fig. 16). On the other hand, the MCTS solution continues to be tractable for $N \geq 11$ (see Fig. 16), when the simulations are performed on a PC with Intel Xeon processor and 32 GB RAM. Additional

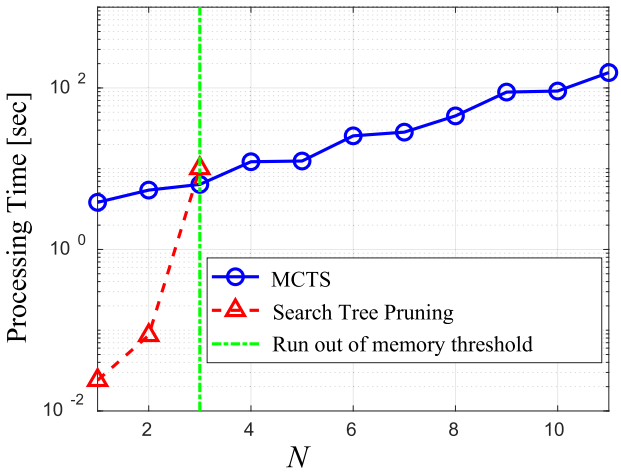


Fig. 16. Processing time required by search-tree pruning and MCTS algorithms for a benchmark example of dimensions analogous to Fig. 15, but with four occlusions and an increasing number of targets (N).

results regarding the planners' computational complexity and performance are provided in the following two sections.

VI. Complexity Analysis

This section presents computational complexity results for the three novel planners developed in this article, namely, the DCG, PV, and CD planners. All three planners require the implementation of a graph optimization algorithm. In this article, graph optimization was carried out using the MCTS method presented in Appendix A, which was found to significantly outperform all other graph-search algorithms (results omitted for brevity). The computational complexity required by MCTS to find the optimal channel in a graph, $\mathcal{G}$, is $O(\mu bd)$, where μ is the number of Monte Carlo simulations, b is the maximum breadth of the (spanning) tree, and h is the maximum depth of the tree. Because the width and depth of the spanning tree must never exceed the number of targets in the workspace (N), the computational complexity required by MCTS algorithm to find the optimal directional sensor path is $O(\mu N^2)$. The computational complexity required by DCG, PV, and CD to obtain the graphs $\mathcal{G}$, $\mathcal{G}_{\mathrm{PV}}$, and $\mathcal{G}_{\mathrm{CD}}$, respectively, is derived in the next three sections.

A. Computational Complexity Analysis of DCG Algorithm

The computational complexity of the target visibility regions (17) is found to be $O(\kappa NMm\log m+\kappa NMm)$, where κ is the number of discretizations for the sensor orientation θ, M is the number of obstacles, m_j is the number of vertices of the polygonal (opaque) object $\mathcal{B}_j$, and

$$m \triangleq \frac{1}{M}\sum_{j=1}^{M} m_j \tag{33}$$

is the average number of vertices of the objects in $\mathcal{W}$. The first term (contribution) to the algorithm's complexity derives from the computation of the convex hull in (14), which requires $O(m\log m)$ time [62] for an opaque object with m vertices and is repeated κNM times to determine the shadow regions of N targets in the presence of M occlusions. The second term of the target-visibility regions' computational complexity derives from the calculation of the coverage cone, which requires $O(m)$ time because finite generated, and is also repeated κNM times.

Finding the set visibility regions in (19) requires determining the power set of the targets, with complexity $O(2^N)$, for discrete sensor orientations (i.e., κ times). Thus, the set visibility regions require time $O(\kappa 2^N)$ and may be prohibitive for very large N. Finally, ensuring that the DCG is connected requires the use of probabilistic roadmaps and Dijkstra's algorithm [30], which results in a computational complexity $O(\eta^2)$, where η is the number of samples used to construct the roadmap. Then, connecting the DCG graph, $\mathcal{G}$, requires $O(\kappa N\eta^2)$ time, and searching $\mathcal{G}$ for the optimal channel via MCTS requires $O(\mu N^2)$ time. Because the stated operations are conducted in series, the computational complexity of the DCG algorithm is

$$O(\kappa NMm\log m+\kappa NMm+\kappa 2^N+\kappa N\eta^2+\mu N^2) \tag{34}$$

It can be seen that the term $\kappa 2^N$ dominates the runtime of the DCG algorithm. The computation required can be reduced by exploiting the fact that set visibility regions do not exist for targets that are more than twice the maximum sensing radius from each other, that is

$$\|\mathbf{x}_i-\mathbf{x}_j\|>2r \Rightarrow \mathcal{TV}_i\cap\mathcal{TV}_j=\emptyset \tag{35}$$

By considering only the indices of targets that are within a distance $2r$ of each other, the computation required can be greatly reduced. Let R denote the maximum number of targets inside any ball of radius $2r$ in $\mathcal{W}$, also referred to as neighboring targets. Then, the set visibility region requires $O(2^R)$ time and, if the targets are approximately uniformly distributed in a workspace of area $A(\mathcal{W})$, it follows that

$$R\approx\frac{\pi r^2}{A(\mathcal{W})}N. \tag{36}$$

Therefore, as long as $r^2\ll A(\mathcal{W})$, the computation required by the DCG algorithm remains tractable. The numerical results plotted in Fig. 17 illustrate that, for small values of the ratio $a=R/N$ (e.g., $a\ll 0.2$), the computational complexity of the set visibility regions is far less than its upper bound (black line) for any number of targets (including $N>15$, not shown for brevity).

B. Computational Complexity Analysis of PV Algorithm

The PV algorithm requires computing all target visibility regions, with complexity $O(\kappa NMm\log m+\kappa NMm)$, but not all of the set visibility regions. By using the projections of set visibility regions with large cardinality, the computation required can be significantly reduced. Assume $\mathcal{V}_P\neq\emptyset$ for $i\in P$. Then, any other set visibility region $\mathcal{V}_Q$, with $Q\ni i$ and $|Q|<|P|$, is ignored by the PV algorithm, and the graph $\mathcal{G}_{\mathrm{PV}}$ is obtained from samples of $\mathcal{V}_P$. By this approach, the maximum number of set visibility regions required is N. Also, because the PV algorithm projects these regions onto $\mathcal{W}$, their construction is not repeated for every (discrete) sensor orientation. As a result, the

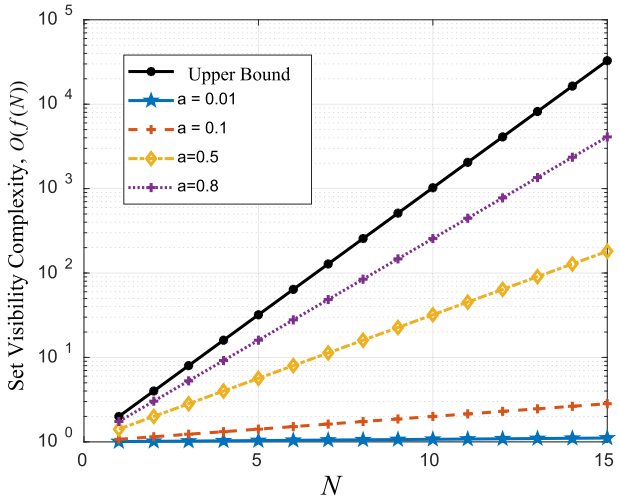


Fig. 17. Computational complexity of set visibility regions as a function of the number of neighboring targets over the total number of targets.

computational complexity is reduced from $O(\kappa 2^N)$ to $O(N)$, in every case (regardless of relative target locations).

The time required by the other steps of the PV algorithm is the same as that of the DCG algorithm. Therefore, the computational complexity of the PV algorithm is

$$O(\kappa NMm \log m + \kappa NMm + N + N\eta^2 + \mu N^2) \quad (37)$$

Clearly, as N increases the PV algorithm provides significant savings when compared to the DCG method, reducing the leading complexity from exponential to linear. Furthermore, the PV graph size is inevitably much smaller than that of the DCG graph, and, thus, the time required by the MCTS optimization is also greatly reduced (see Section VII).

C. Computational Complexity Analysis of CD Algorithm

The computational complexity of rectangloid decompositions [20] is applied here by considering the decomposition of C-obstacles and target visibility regions that are possibly concave and characterized by a total of b and v edges, respectively. The discretization of the sensor orientation in configuration space is incorporated linearly, such that obtaining the rectangloid decomposition $\mathcal{K} \cup \mathcal{K}_{\text{void}}$ requires $O(\kappa(b+v)^2)$ time. Then, the construction of the CD graph, $\mathcal{G}_{\text{CD}}$, requires $O(\kappa(b+v))$ time, as shown in [20].

The search for the optimal path in $\mathcal{G}_{\text{CD}}$ is conducted via MCTS. However, in this case, the search can lead to much wider and deeper trees, with approximate dimensions $\kappa(b+v)$, and, therefore, the MCTS optimization requires $O(\mu\kappa^2(b+v)^2)$ time. Thus, the computational complexity of the CD algorithm is

$$O(\kappa(b+v)^2 + \kappa(b+v) + \mu\kappa^2(b+v)^2) \quad (38)$$

Because v can be relatively large for complex workspaces, the CD algorithm can become prohibitive for complex and large workspaces populated by many targets.

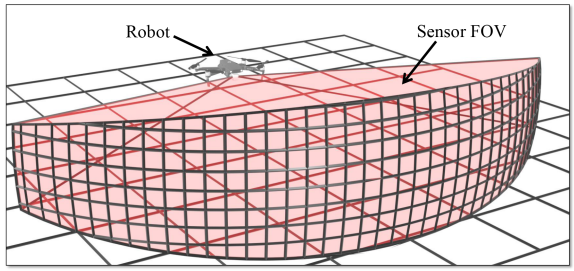


Fig. 18. Robot and sensor FOV used in Unreal Engine simulation environment.

In summary, the computation required by the DCG method scales exponentially in the number of targets, that of the PV algorithm scales quadratically in the number of targets, and that of the CD algorithms scales quadratically in the total number of edges of occlusions and visibility regions.

VII. Directional Sensor Planning Results

The performance and computational complexity of the four sensor planning algorithms developed in this article are tested and compared both in simulations and in physical experiments. Although the DCG algorithm is expected to provide the best path performance by leveraging a complete representation of the set visibility regions, the other planners also present several advantages and can offer useful solutions depending on the problem characteristics. For instance, the PV algorithm is shown to provide a good tradeoff between computational efficiency and path performance by discarding set visibility regions with small cardinality and projecting them onto the 2-D sensor workspace. The CD algorithm, on the other hand, is potentially useful for guaranteeing completeness, but may be computationally prohibitive for complex occlusions' and visibility regions' geometries.

The four novel planners presented in Section IV are also compared here to a coverage algorithm inspired by the WRP solution in [18]. The coverage algorithm combines the notion of lawnmower path in the sensor workspace with a probabilistic roadmap in order to cover all targets while avoiding collisions with obstacles. Because WRP solutions do not account for the bounded sensor FOV, target coverage is not guaranteed. The performance of the algorithms is evaluated by means of the path length $J(\tau)$ in (4), the number of targets observed along a path, denoted by $n_T(\tau) \in [0, N]$, and the overall sensor performance $n_T(\tau)/J(\tau)$. The normalized target coverage metric, $n_T(\tau)/N$, is used to compare different workspaces, where the maximum value of one represents complete coverage, i.e., all N targets are observed at least once by the sensor.

A. DCG Simulation Results

In this section, the effectiveness of the visibility theory and DCG algorithm presented in this article is demonstrated using the simulation of a camera equipped drone (see Fig. 18) operating in a complex and photorealistic workspace (see Fig. 19), simulated using Unreal Engine [27]. All simulations and algorithms

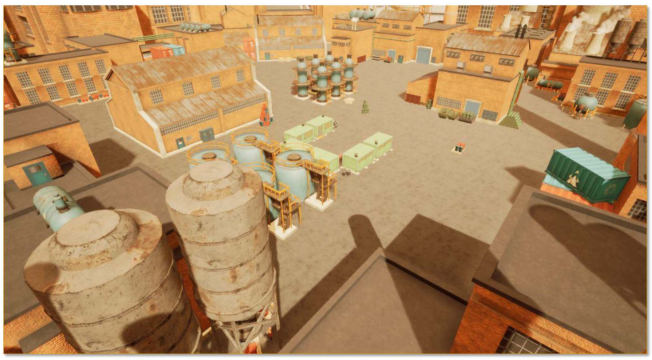


Fig. 19. Unreal Engine simulation environment.

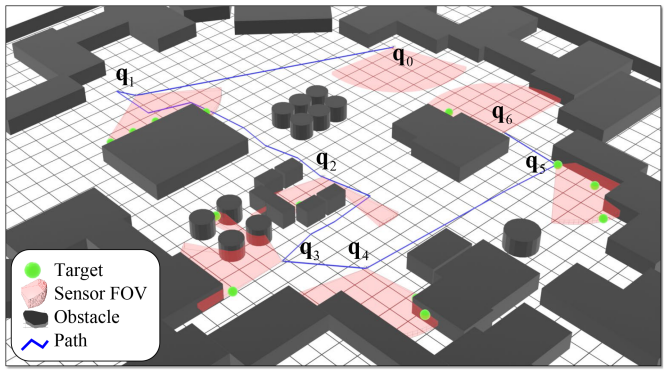


Fig. 20. Geometric rendering of simulation workspace in Fig. 19 used to obtain the set visibility regions and the optimal sensor path, illustrated by a solid line and six sample sensor configurations ($\mathbf{q}_1, \ldots, \mathbf{q}_6$).

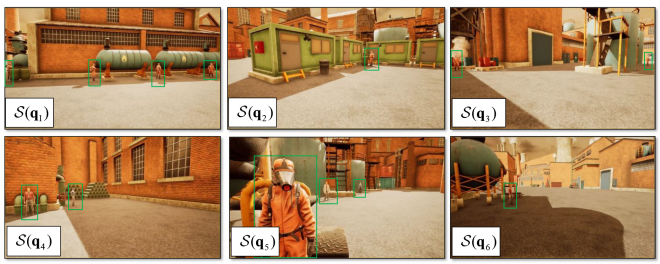


Fig. 21. Examples of camera frames obtained by the DCG algorithm enabling human target detection (green bounding boxes) by the onboard R-CNN.

are implemented on a Dell Precision Tower 7910 equipped with two Intel 2.40 GHz Xeon CPU processors. The Unreal Engine camera is simulated to be at high-resolution (1080p image resolution) and to have a maximum sensing range $r = 25$ m. The Unreal Engine environment is populated with human targets (see Fig. 21) and opaque objects comprised of elaborate buildings and machineries (see Fig. 19). The DCG algorithm is implemented in MATLAB 2019b and interfaced with Unreal Engine, as well as with an off-the-shelf faster R-CNN algorithm [28] for target detection. These Unreal Engine simulations also allow for variable luminosity and shadow conditions, as shown in Fig. 19.

The workspace, $\mathcal{W}$, in this simulation consists of $M = 59$ opaque objects and $N = 13$ human targets. The opaque objects also comprise obstacles to be avoided by the drone, making up several narrow passages. The initial configuration ($\mathbf{q}_0$ in Fig. 20) is given with no visible targets, and the polygonal representation of the objects (see Fig. 20) is obtained from the Unreal Engine mesh data. The convex hull of each opaque object is obtained from its vertices, extruding the geometries to be uniform in the vertical direction. After constructing the set visibility regions (as shown in Section III) the DCG planner (see Section IV-A) is used to obtain the optimal sensor path. The sensor position is plotted as a solid blue line in Fig. 20 along with six sample sensor configurations, in order to illustrate the sensor orientation (and corresponding FOV in pink) at six moments in time. The DCG sensor path provides both position and orientation at every time step and is executed by interfacing it with the drone feedback controller. As shown in Fig. 20, the optimal DCG sensor path successfully observes all 13 targets (green circles). As a result, the onboard R-CNN object recognition algorithm successfully recognizes the 13 human targets, as shown by the R-CNN bounding boxes (green rectangles) plotted in Fig. 21 for six sample frames obtained by the camera-equipped drone.

As shown in Fig. 22, the DCG algorithm finds configurations capable of viewing multiple targets simultaneously, thanks to the use of set visibility regions. It can be seen that, by using knowledge of the occlusions, the drone configuration allows the onboard camera to observe a human inside a narrow passage using a short and efficient path (see Fig. 20). Finally, a small workspace in Fig. 23 is used to illustrate that the visibility theory and the DCG algorithm presented in this article allow us to simultaneously optimize the sensor performance and avoid drone collisions with the obstacles. In this example, one of the targets is only visible through a narrow passage between two walls that the drone may reach only by flying through another narrow passage in its workspace.

1) Performance Comparison: The four novel planners presented in this article (see Section IV) and the WRP-inspired coverage algorithm are compared in this section using multiple workspaces and sensor conditions, generated as follows. A variable number of targets (N) and opaque objects (M) are chosen by the user, while the initial sensor configuration ($\mathbf{q}_0$), and the target and object positions and geometries ($\mathbf{x}_1, \ldots, \mathbf{x}_N$ and $\mathcal{B}_1, \ldots, \mathcal{B}_M$) are randomly generated. Positions are sampled uniformly over $\mathcal{W}$, and polygonal objects are randomly obtained with a number of vertices between 3 and 8. An example workspace, obtained by this approach, is shown in Fig. 24 for $M = 25$ obstacles and $N = 8$ targets, along with the optimal DCG sensor path. The same MCTS parameters are used for all four novel planners. Then, three case studies are used to investigate and compare the algorithm performance as a function of number of targets, number of occlusions, and maximum sensor range.

In the *first case study*, the number of targets varied from $N = 1$ to $N = 15$, while the number of opaque objects and the sensor range were held fixed at $M = 25$ and $r = 5$ m, respectively. The results, averaged over multiple workspaces and conditions and plotted in Fig. 25, show that, for small N, the algorithms perform similarly. But, as the number of target increases, the DCG and PV algorithms provide the shortest

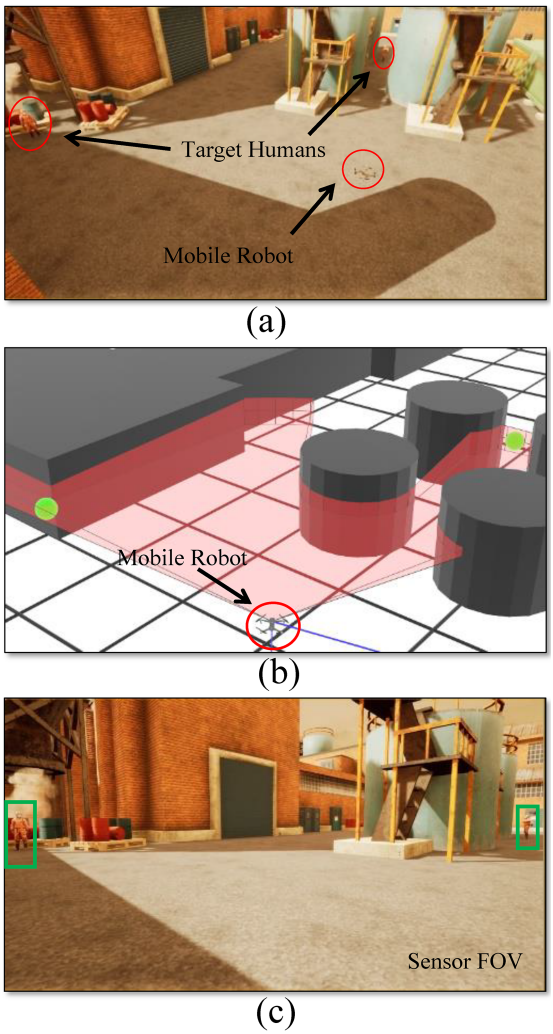


Fig. 22. (a) Bird-eye view of a sensor configuration along the DCG optimal path, enabling the view of two targets including a human inside a narrow passage, with (b) geometric rendering, and (c) corresponding camera frame with R-CNN human bounding boxes plotted in green.

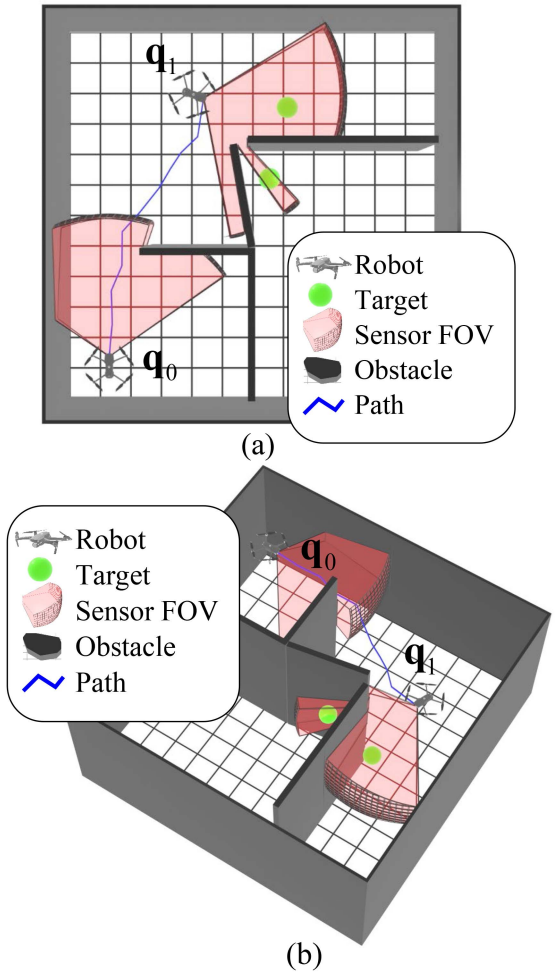


Fig. 23. Example of DCG optimal sensor path enabling simultaneous obstacle avoidance by the drone and efficient camera viewing of two targets through the only opportunity presented by a narrow passage.

distance while covering all targets. The TSP algorithm in this case is the only one that fails to observe all targets because the sensor misses targets that are close to the occlusions for its inability to prevent collisions while guaranteeing visibility. Because the number of MCTS simulations (μ) is held constant for all N, the PV algorithm eventually outperforms DCG as the size of its graph does not grow as rapidly as that of DCG. However, the DCG algorithm could be made to outperform PV by increasing the value of μ at the expense of computation.

In the *second case study*, the number of opaque objects is varied from $M = 1$ to $M = 30$, while $N = 10$ and $r = 5$ m are held constant. The results, averaged over multiple workspaces and conditions and plotted in Fig. 26, show that PV and DCG provide the best sensor performance, and that all of the three target-visibility-based planners (PV, DCG, and CD) guarantee coverage of all targets even as the number of occlusions increases. Also, it can be seen from Fig. 26(c) that the density of

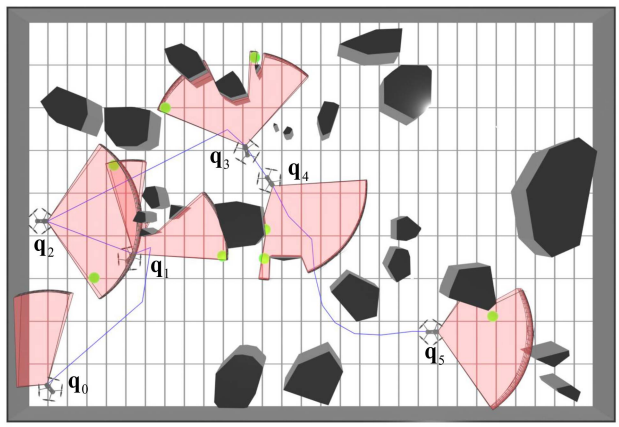


Fig. 24. Example of sensor workspace, randomly generated to include $M = 25$ opaque objects and $N = 8$ targets, and optimal DCG sensor path.

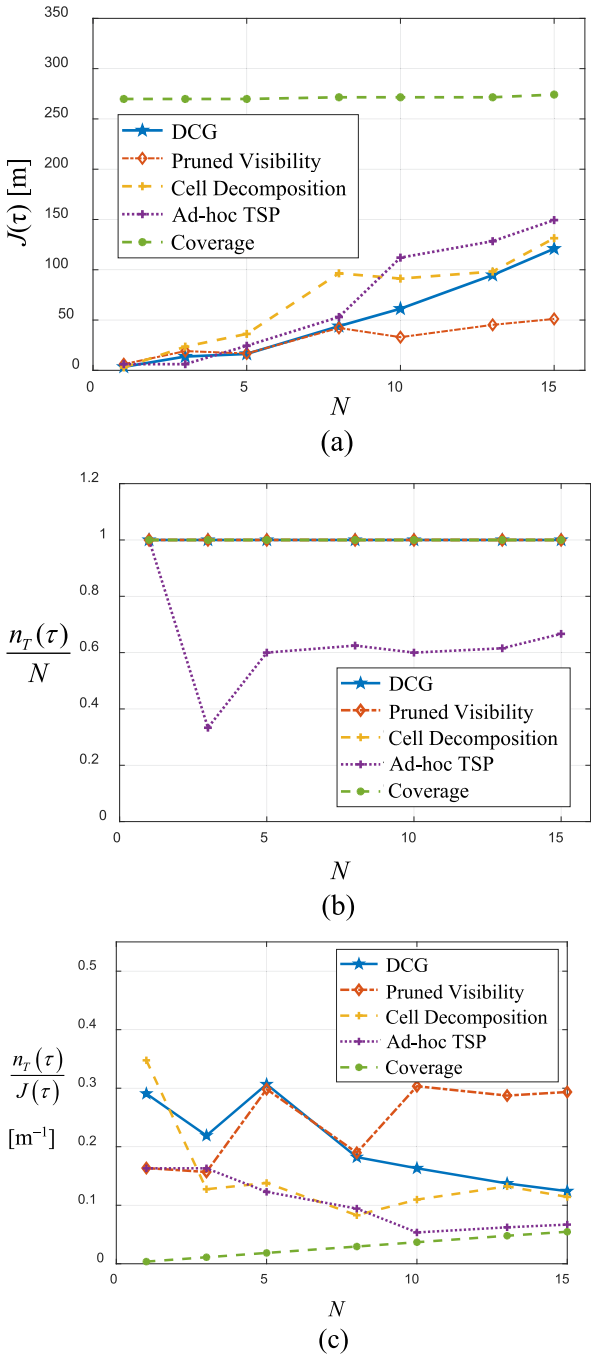


Fig. 25. (a) Sensor path length, (b) normalized target coverage, and (c) sensor performance as a function of the number of targets.

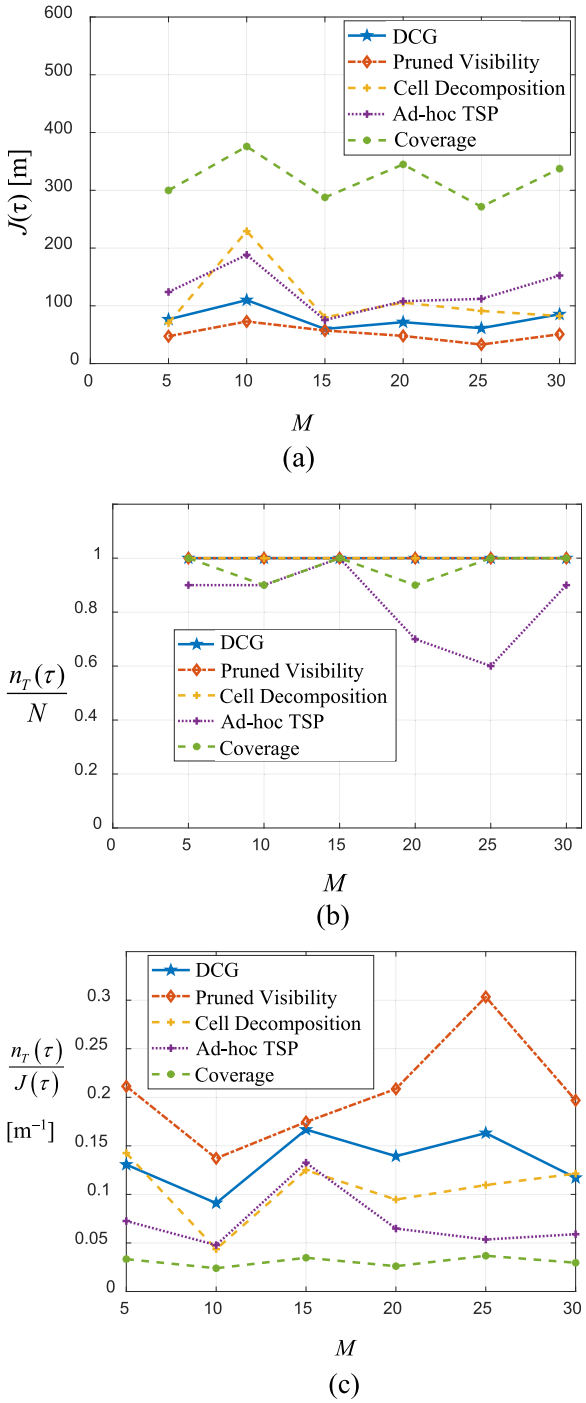


Fig. 26. (a) Sensor path length, (b) normalized target coverage, and (c) sensor performance as a function of the number of opaque objects.

opaque objects does not decrease the performance of these three planners, while it does prevent complete coverage by the TSP and coverage algorithms [see Fig. 26(b)].

In the *third case study*, the maximum sensing range is increased from $r = 0$ (point sensor) to $r = 15$ m, while $N = 3$ and $M = 25$ are held constant. The average results, plotted in Fig. 27, show that the performance of all four novel planners presented in Section IV does not vary significantly with the sensor range, with DCG displaying the best performance overall. As before, the TSP algorithm cannot provide full coverage. Although in this case the coverage algorithm is successful at observing all targets, it also requires a very long path, such that even for large r its performance is far less than that of the planners developed in this article. In the limit of $r = 0$, the sensor must visit every target, and, thus, the coverage algorithm requires the sensor to travel a very large distance compared to other methods.

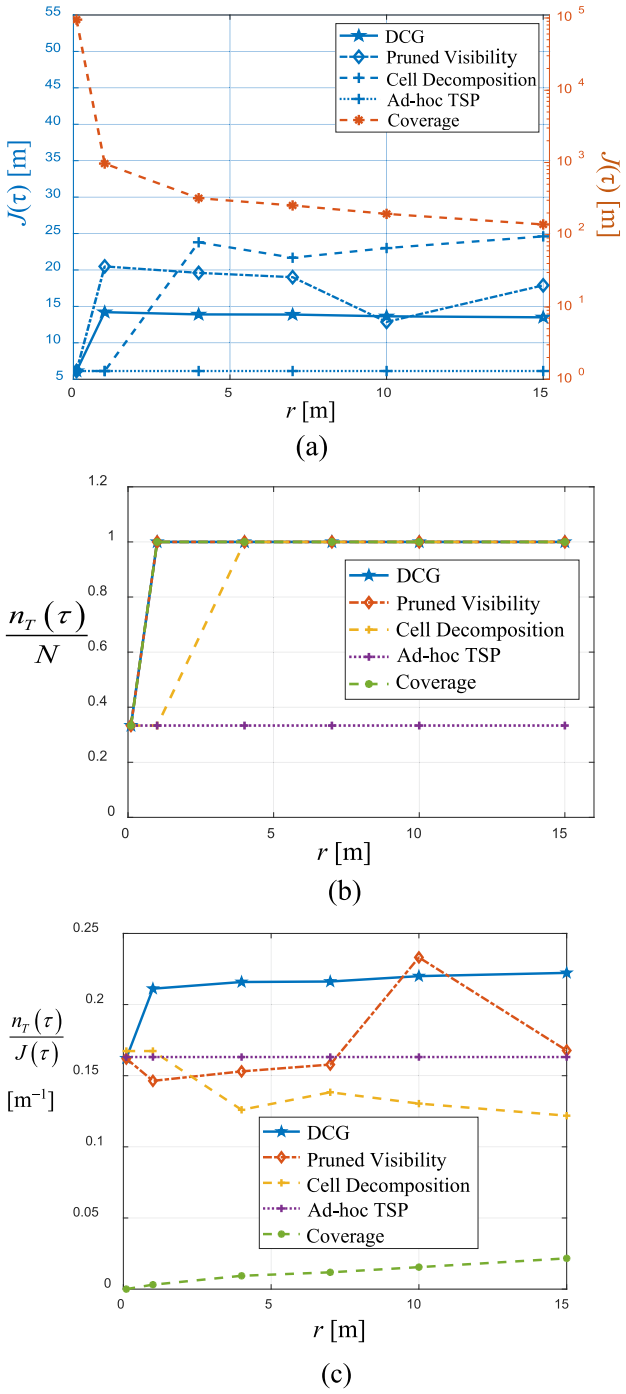


Fig. 27. (a) Sensor path length, (b) normalized target coverage, and (c) sensor performance as a function of maximum sensor range.

TABLE I
NUMBER OF GRAPH NODES

	$N=1$	3	5	8	10	13	15
DCG	9	34	76	110	127	175	238
PV	2	3	3	7	9	13	14
CD	442	500	560	675	709	854	906
TSP	2	4	6	9	11	14	16

TABLE II
NUMBER OF GRAPH EDGES

	$N=1$	3	5	8	10	13	15
DCG	36	392	2267	4838	6505	12610	23380
PV	1	5	3	38	66	139	163
CD	2871	3991	5182	7217	7924	10400	11640
TSP	1	2	8	26	38	67	101

TABLE III
GRAPH CONSTRUCTION TIME (SECONDS)

	$N=1$	3	5	8	10	13	15
DCG	1.6	347.3	4173.6	12388	16741	32141	51753
PV	0.4	16.4	1.4	137.6	244.9	410.6	474.5
CD	378.1	606.9	850.9	1085.2	1240.1	1436.1	1506.6
TSP	0.2	31.0	70.9	235.8	389.3	540.2	683.1

TABLE IV
GRAPH OPTIMIZATION TIME (SECONDS)

	$N=1$	3	5	8	10	13	15
DCG	1.8	9.2	16.1	74.7	173.4	196.8	365.5
PV	0.3	0.8	0.9	5.4	7.1	12.0	15.3
CD	95.7	635.9	4837.2	21589	21101	62054	78421
TSP	0.3	0.3	1.7	4.0	6.1	11.0	15.1

TABLE V
TOTAL RUN TIME (SECONDS)

	$N=1$	3	5	8	10	13	15
DCG	3.4	356.5	4189.7	12463	16914	32338	52119
PV	0.7	17.2	2.3	143.0	252.0	422.6	489.8
CD	473.8	1242.8	5688.1	22674	22341	63490	79928
TSP	0.5	31.3	72.6	239.8	395.4	551.2	698.2
Cov	46.1	52.5	52.8	52.6	52.7	44.6	43.3

From the computational complexity analysis in Section VI, it can be seen that the number of targets N is the parameter with the greatest influence on running time required by the target-visibility-based algorithms (DCG, PV, and CD) presented in this article. Therefore, a comparison of the computational requirements as a function of N is provided in Tables I–V. The graph size, shown in Tables I and II, highlights the reason for the high CD and DCG computational complexity. The PV method, on the other, requires a significantly smaller graph while achieving competitive sensor performance. In fact, the number of nodes required by PV is bounded as follows: $|\mathcal{N}_{\mathrm{PV}}| \leq N+1$. Therefore, PV is always at least as efficient as TSP in terms of graph size. Tables III and IV show the time required to respectively generate and optimize each of the graphs, confirming the analysis in Section VI. The total runtime, obtained by summing the entries of Tables III and IV, is shown in Table V. Overall, CD is shown to be the most expensive of the methods, followed by the DCG method. Interestingly, PV is more computationally efficient than the TSP algorithm, because it takes advantage

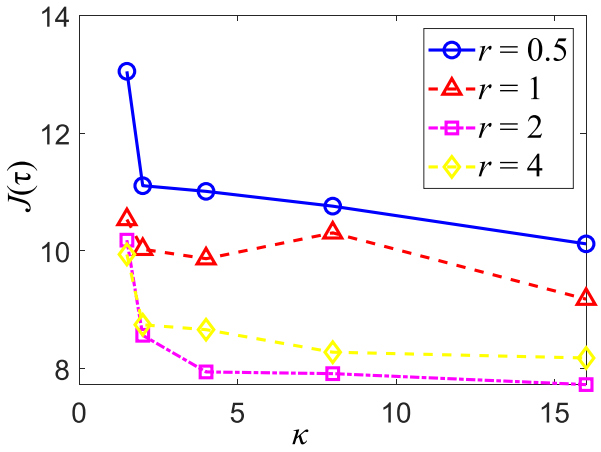


Fig. 28. DCG planner performance decreases with the number of discretizations (κ) based on the size of the sensor FOV, r (results shown for a workspace with $N = M = 4$, not shown for brevity).

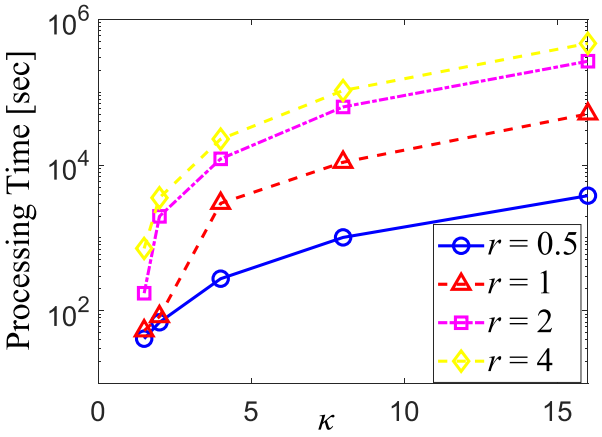


Fig. 29. Processing time required for DCG graph construction and search increases with the number of discretizations (κ) based on the size of the sensor FOV, r (results shown for a workspace with $N = M = 4$, not shown for brevity).

of intersecting target visibility regions. Although the coverage algorithm is by far the most computationally efficient, it is ultimately unable to observe all targets, as shown in Figs. 25–27.

2) Impact of Discretization on DCG Planner Performance: Because the DCG planner seeks an optimal path in configuration space, the target visibility regions derived in Section III-A typically must be discretized with respect to the robot orientation, using κ intervals. From an extensive set of simulation studies, the most significant results shown in Figs. 28 and 29 demonstrate that the best choice of κ depends primarily on the size of the sensor FOV, r. For a given value of r, a good compromise for the choice of κ can be found such that the path performance is close to optimal (see Fig. 28) with minimal processing time (see Fig. 29). After this ideal tradeoff (elbow of the curves in Fig. 28), increasing the value of κ continues to increase the processing time (as anticipated by the computational complexity analysis in Section VI) without significantly improving the path performance.

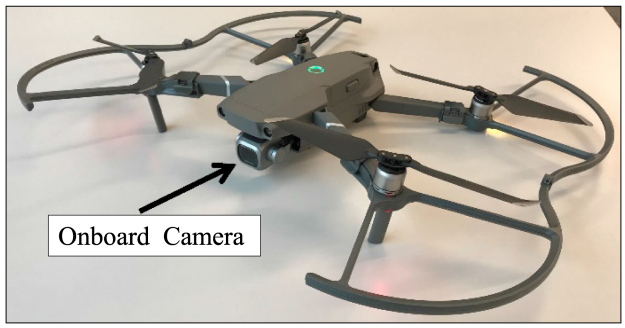


Fig. 30. DJI Quadrotor used in physical experiments equipped with an onboard CMOS monocular camera.

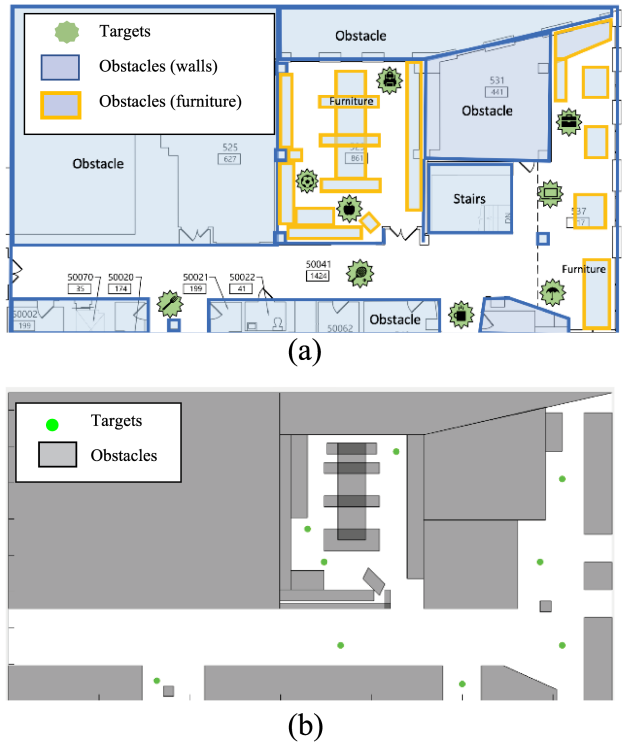


Fig. 31. (a) Indoor workspace for physical experiments conducted near and inside the LISC, located on the fifth floor of Upson Hall at Cornell University, and (b) polygonal map representation used by the DCG planner.

B. Experimental Results

The DCG method, shown to have the best overall sensor path performance of all the algorithms developed in this article, was also tested on a DJI Mavic 2.0 quadrotor equipped with a 1-inch CMOS monocular camera (see Fig. 30). The quadrotor was also equipped with multiple vision and time-of-flight depth sensors for highly accurate state estimation via the built-in DJI functionality, and with a GPS operable in most outdoor environments. The experiments were conducted both indoor and outdoor on the Cornell University campus in Ithaca, NY, USA. As shown in Figs. 31– 34, a variety of targets including backpacks, laptop computers, umbrellas, and bottles, to name a few, were placed

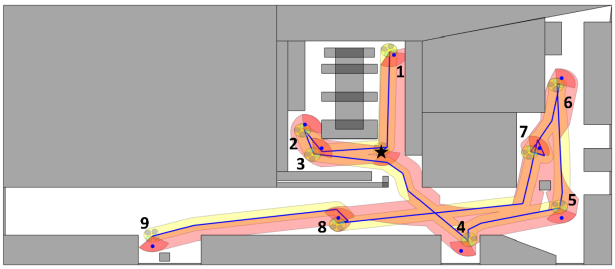


Fig. 32. Optimal DCG sensor path executed by quadrotor to observe all nine targets in the indoor workspace in Fig. 31.

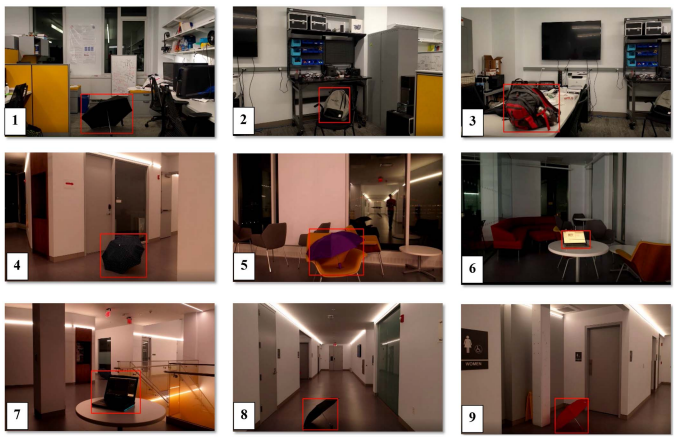
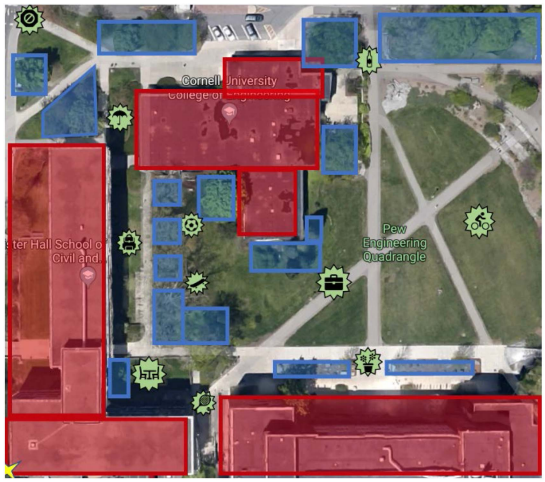
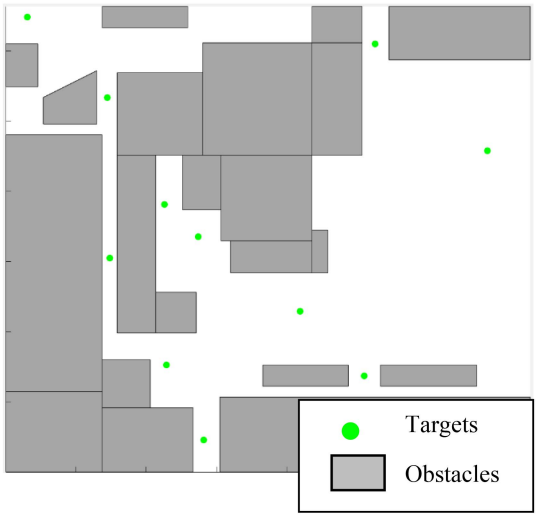


Fig. 33. Quadrotor camera frames, obtained indoor by executing the optimal DCG sensor path in Fig. 32, show that all nine targets are recognized by the R-CNN (red bounding boxes).



(a)



(b)

Fig. 34. Outdoor workspace for physical experiments conducted in (a) the Hollister parking lot at Cornell University and (b) polygonal map representation used by the DCG planner (see [63] for a video of the experiments).

in the workspace, and their location was used along with a map of the workspace to develop the target and set visibility regions.

Using the initial drone configuration (position and orientation), the optimal sensor path was obtained by the DCG algorithm prior to takeoff. A closed-loop trajectory-following controller was implemented in Java on a Dell Alienware mobile workstation with an Intel CPU. The drone state estimate was computed onboard and communicated to the workstation over a Wi-Fi connection, such that the corresponding motor control inputs could be computed and communicated back to the quadrotor in real time. The R-CNN object recognition software [28] was then used to process the camera frames and detect the targets in the sensor FOV by producing a bounding box (red box in Fig. 33). Once detected via R-CNN, a target was considered "covered" by the sensor. Two experimental case studies were considered: 1) an indoor workspace located in Cornell Upson Hall (see Fig. 31), and 2) an outdoor workspace located in Cornell Hollister parking lot, adjacent to Hollister Dr. (see Fig. 34).

In the *indoor case study*, the GPS is not available to help localize the quadrotor. The velocity of the quadrotor is estimated by fusing the data from an IMU and the optical flows computed from a down-facing camera. Then, the position is obtained by numerically integrating the estimated velocity, and the heading is estimated by fusing the information from an IMU and a magnetic compass. The workspace included ample occlusions and narrow passages because of the presence of a hallway, sitting area, and cluttered LISC lab space. Nine targets comprised of backpacks, computers, and other objects were placed in this workspace modeled by 26 opaque objects that also functioned as obstacles for the quadrotor. The optimal DCG sensor path was executed successfully by the quadrotor also covering all nine targets, as shown by the results in Fig. 32. The quadrotor was able to avoid collisions and navigate successfully through narrow passages, such as the doorway connecting the LISC to the hallway, in order to find and observe all nine targets. The resulting camera frames, processed via R-CNN for object recognition, are shown in Fig. 33 and show that by this approach the onboard camera was able to obtain occlusion-free high-quality frames from all nine targets.

In the *outdoor case study*, the quadrotor operated in a larger scale outdoor environment, shown in Fig. 34, that was populated

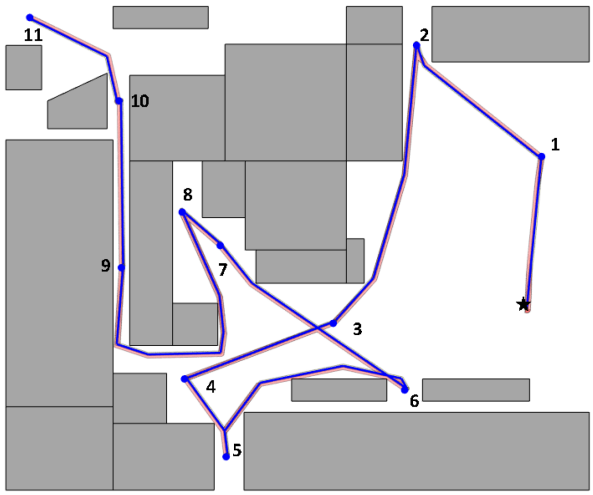


Fig. 35. Optimal DCG sensor path executed by quadrotor to observe all 11 targets in the outdoor workspace in Fig. 34 (see [63] for a video of the experiments).

by 21 opaque objects comprised of large buildings (red regions) and trees (blue regions) that constitute obstacles for the robot and occlusions for the onboard camera. Eleven stationary targets comprised of umbrellas, athletic gear, other inanimate objects, as well as a human (see Fig. 34), were placed in this workspace, resulting in the successful execution of optimal DCG sensor paths, as shown by the example in Fig. 35. These experiments, filmed in [63], demonstrate that the DCG approach developed in this article can be successfully applied to real-world problems of useful dimensions. In this case, the drone was able to access GPS coordinates and use them to follow the optimal DCG path, thereby recognizing all 11 targets using the R-CNN algorithm.

VIII. Conclusion

This article presents novel visibility theory for mobile directional sensors, such as cameras, characterized by both a bounded FOV and LOS visibility. Using the coverage cone and convex hull of polygonal occlusions, relative to the sensor position (FOV apex), the shadow region of each occlusion can be determined. Similarly, the target and set visibility regions can be obtained in closed form, as a function of the target positions, and used to construct a connectivity graph representation of directional visibility in free configuration space. Three novel planning approaches, DCG, PV, and CD, are developed using directional visibility theory and, then, compared to two TSP and lawnmower coverage algorithms developed based on existing paradigms. The computational complexity analysis and comparative studies show that, although the three directional visibility planners present different advantages, DCG and PV significantly outperform DC. Their effectiveness is demonstrated both through extensive simulations with varying densities of targets and occlusions, as well as indoor and outdoor physical experiments on camera-equipped autonomous drones. In all cases, DCG and PV outperform other methods, including coverage and TSP, and achieve both target coverage and collision avoidance with minimum distance traveled. Future work will extend the theory and methods presented in this article to problems involving moving targets, moving occlusions, and online planning.

Appendix A
MCTS Optimization

An MCTS metaheuristic optimization approach inspired by the work in [64] is developed in order to adaptively search a connectivity graph $\mathcal{G}$ for a path connecting the root node n_0 to the shortest path achieving the goals of the chosen directional sensor planning algorithm (see Section IV). Consider the DCG planner as an example. For every node n_p in a fully connected graph $\mathcal{G}$, with $(n_0, n_p) \in \mathcal{E}$, μ Monte Carlo (MC) simulations are performed, where $\mu > 0$ is chosen by the user. One MC simulation consists of iteratively and randomly adding a node to the path until all targets labels have been covered or there are no unvisited nodes in $\mathcal{G}$. At the end of the MC simulation, the total length of the path is evaluated, such that, after μ simulations, the node n_l with minimum path length is selected, where $(n_p, n_l) \in \mathcal{E}$. Then, the process is repeated with n_l as the root node, considering the set of unvisited nodes connected to n_l. This process continues until the shortest path has visited all targets. The value function used for the MC simulation is defined as

$$Q(n_p) \triangleq \frac{1}{\bar{J}} + \gamma \sqrt{\frac{\log N_l}{N_p}} \tag{39}$$

where N_l and N_p are the number of times nodes n_l and n_p have been visited, respectively, and γ is a predefined parameter that represents the exploration–exploitation tradeoff. $\bar{J}$ is the minimum-length path found over all MC simulations that have traversed edge $(n_l, n_p) \in \mathcal{E}$.

References

[1] S. Ferrari and T. Wettergren, *Information-driven Planning and Control* (Cyber physical systems series). Cambridge, MA, USA: MIT Press, 2021. [Online]. Available: https://books.google.com/books?id=ijCXzQEACAAJ

[2] J. Gemerek, S. Ferrari, B. H. Wang, and M. E. Campbell, "Video-guided camera control for target tracking and following," in *Proc. 2nd Annu. Conf. Cyber- Phys. Hum. Syst.*, 2018, pp. 176–183.

[3] H. Wei *et al.*, "Camera control for learning nonlinear target dynamics via Bayesian nonparametric Dirichlet-process Gaussian-process (DP-GP) models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 95–102.

[4] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware path planning," 2016, arXiv:1605.04151.

[5] V. E. Gai, I. V. Polyakov, and O. V. Andreeva, "Depth mapping method based on stereo pairs," in *Proc. Int. Conf. Neuroinform.*, 2019, pp. 303–308.

[6] J. Steinbaeck, C. Steger, G. Holweg, and N. Druml, "Next generation radar sensors in automotive sensor fusion systems," in *Proc. IEEE Sensor Data Fusion: Trends, Solutions, Appl.*, 2017, pp. 1–6.

[7] P. Zhu, J. Isaacs, B. Fu, and S. Ferrari, "Deep learning feature extraction for target recognition and classification in underwater sonar images," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 2724–2731.

[8] B. Englot and F. Hover, "Inspection planning for sensor coverage of 3D marine structures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 4412–4417.

[9] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1656–1665.

[10] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2032–2039, Jul. 2018.

[11] M. J. Bays, A. Shende, D. J. Stilwell, and S. A. Redfield, "A solution to the multiple aspect coverage problem," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1531–1537.

[12] J. R. Gemerek, S. Ferrari, and J. D. Albertson, "Fugitive gas emission rate estimation using multiple heterogeneous mobile sensors," in *Proc. ISOCS/IEEE Int. Symp. Olfaction Electron. Nose*, 2017, pp. 1–3.

[13] J. D. Albertson *et al.*, "A mobile sensing approach for regional surveillance of fugitive methane emissions in oil and gas production," *Environ. Sci. Technol.*, vol. 50, no. 5, pp. 2487–2497, 2016.

[14] A. M. Nascimento *et al.*, "A systematic literature review about the impact of artificial intelligence on autonomous vehicle safety," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 12, pp. 4928–4946, Dec. 2020.

[15] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *J. Intell. Robot. Syst.*, vol. 74, no. 3–4, pp. 965–983, 2014.

[16] H. Wei and S. Ferrari, "A geometric transversals approach to sensor motion planning for tracking maneuvering targets," *IEEE Trans. Autom. Control*, vol. 60, no. 10, pp. 2773–2778, Oct. 2015.

[17] J. Urrutia, "Art gallery and illumination problems," in *Handbook of Computational Geometry*. Amsterdam, The Netherlands: Elsevier, 2000, pp. 973–1027.

[18] R. Goroshin, Q. Huynh, and H.-M. Zhou, "Approximate solutions to several visibility optimization problems," *Commun. Math. Sci.*, vol. 9, no. 2, pp. 535–550, 2011.

[19] W. Lu, G. Zhang, and S. Ferrari, "An information potential approach to integrated sensor path planning and control," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 919–934, Aug. 2014.

[20] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition," *IEEE Trans. Syst., Man, Cybernet., Part B., Cybernet.*, vol. 39, no. 3, pp. 672–689, Jun. 2009.

[21] A. Swingler and S. Ferrari, "On the duality of robot and sensor path planning," in *Proc. 52nd IEEE Conf. Decis. Control*, 2013, pp. 984–989.

[22] K. Baumgartner and S. Ferrari, "A geometric transversal approach to analyzing track coverage in sensor networks," *IEEE Trans. Comput.*, vol. 57, no. 8, pp. 1113–1128, Aug. 2008.

[23] H. Wei, W. Lu, P. Zhu, G. Huang, J. Leonard, and S. Ferrari, "Optimized visibility motion planning for target tracking and localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 76–82.

[24] G. Zhang, S. Ferrari, and M. Qian, "An information roadmap method for robotic sensor path planning," *J. Intell. Robot. Syst.*, vol. 56, no. 1–2, pp. 69–98, 2009.

[25] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ, USA:: Princeton Univ. Press, 2006.

[26] H. Choset, "Coverage for robotics—A survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1–4, pp. 113–126, 2001.

[27] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and Service Robotics*, Springer, Cham, 2018.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Toward real-time object detection with region proposal networks," *Adv. neural inf. process. syst.*, vol. 28, 2016.

[29] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[30] T. Bröcker and K. Jänich, *Introduction to Differential Topology*. Cambridge, U.K.: Cambridge Univ. Press, 1982.

[31] J.-C. Latombe, *Robot Motion Planning*, vol. 124. Berlin, Germany: Springer Science & Business Media, 2012.

[32] S. Ferrari and C. Cai, "Information-driven search strategies in the board game of clue," *IEEE Trans. Systems, Man, Cybern. Part B, Cybern.*, vol. 39, no. 3, pp. 607–625, Jun. 2009.

[33] D. P. Bertsekas, *Convex Analysis and Optimization*. Belmont, MA, USA: Athena Scientific, 2003.

[34] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[35] J. A. De Loera, R. Hemmecke, and M. Köppe, *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*. Philadelphia, PA, USA: SIAM, 2012.

[36] MATLAB, version 9.5.0 (R2019b). Natick, MA, USA: The MathWorks Inc., 2010.

[37] C. Cai and S. Ferrari, "Comparison of information-theoretic objective functions for decision support in sensor systems," in *Proc. IEEE Amer. Control Conf.*, 2007, pp. 3559–3564.

[38] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1498–1511, Dec. 2016.

[39] A. K. Budhiraja, "View point planning for inspecting static and dynamic scenes with multi-robot teams," Ph.D. dissertation, Virginia Tech, Blacksburg, VA, USA 2017.

[40] K. Elbassioni, A. V. Fishkin, N. H. Mustafa, and R. Sitters, "Approximation algorithms for Euclidean group TSP," in *Proc. Int. Colloq. Automata, Lang., Program.*, Berlin, Heidelberg: Springer, 2005.

[41] J. S. Mitchell *et al.*, "Geometric shortest paths and network optimization," in *Handbook of Computational Geometry*, vol. 334, Amsterdam, The Netherlands: Elsevier Science, 2000, pp. 633–702.

[42] J. Faigl, "GSOA: Growing self-organizing array-unsupervised learning for the close-enough traveling salesman problem and other routing problems," *Neurocomputing*, vol. 312, pp. 120–134, 2018.

[43] K. Vicencio, B. Davis, and I. Gentilini, "Multi-goal path planning based on the generalized traveling salesman problem with neighborhoods," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 2985–2990.

[44] J. Faigl, P. Váňa, and J. Deckerová, "Fast heuristics for the 3-D multi-goal path planning based on the generalized traveling salesman problem with neighborhoods," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2439–2446, Jul. 2019.

[45] D. P. Bertsekas and A. Scientific, "Dynamic programming and optimal control: 4th and earlier editions,".

[46] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958–2008*. Berlin, Heidelberg: Springer, 2010, pp. 105–132.

[47] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *J. Oper. Res. Soc. Amer.*, vol. 2, no. 4, pp. 393–410, 1954.

[48] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2, pp. 235–256, 2002.

[49] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[50] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Inf. Process. Lett.*, vol. 12, no. 3, pp. 133–137, 1981.

[51] N. H. Mustafa and S. Ray, "Improved results on geometric hitting set problems," *Discrete Comput. Geometry*, vol. 44, no. 4, pp. 883–895, 2010.

[52] P. K. Agarwal and J. Pan, "Near-linear algorithms for geometric hitting sets and set covers," in *Proc. 13th Annu. Symp. Comput. Geometry*, 2014, pp. 271–279.

[53] N. Bus, N. H. Mustafa, and S. Ray, "Practical and efficient algorithms for the geometric hitting set problem," *Discrete Appl. Math.*, vol. 240, pp. 25–32, 2018.

[54] S. Alatartsev, S. Stellmacher, and F. Ortmeier, "Robotic task sequencing problem: A survey," *J. Intell. Robot. Syst.*, vol. 80, no. 2, pp. 279–298, 2015.

[55] M. de Berg, J. Gudmundsson, M. J. Katz, C. Levcopoulos, M. H. Overmars, and A. F. van der Stappen, "TSP with neighborhoods of varying size," *J. Algorithms*, vol. 57, no. 1, pp. 22–36, 2005.

[56] K. Helsgaun, "Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun algorithm," *Math. Program. Comput.*, vol. 7, no. 3, pp. 269–287, 2015.

[57] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Comput. Oper. Res.*, vol. 87, pp. 1–19, 2017.

[58] W. K. Mennell, "Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem," Ph.D. dissertation, Univ. Maryland, College Park, MD, USA 2009.

[59] S. K. Ghosh and D. M. Mount, "An output-sensitive algorithm for computing visibility graphs," *SIAM J. Comput.*, vol. 20, no. 5, pp. 888–910, 1991.

[60] C. Nissoux, T. Siméon, and J.-P. Laumond, "Visibility based probabilistic roadmaps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Systems. Hum. Environ. Friendly Robots With High Intell. Emotional Quotients (Cat. No. 99CH36289)*, 1999, vol. 3, pp. 1316–1321.

[61] D. G. Kirkpatrick and R. Seidel, "The ultimate planar convex hull algorithm," *SIAM J. Comput.*, vol. 15, no. 1, pp. 287–299, 1986.
[62] Y. Chen and D. van Vijk, "LISC drone experiments on occlusion avoidance," Cornell University, 2021. [Online]. Available: https://www.youtube.com/watch?v=RcrCskNsyZM
[63] E. J. Powley, D. Whitehouse, and P. I. Cowling, "Monte Carlo tree search with macro-actions and heuristic route planning for the physical travelling salesman problem," in *Proc. IEEE Conf. Comput. Intell. Games*, 2012, pp. 234–241.

Jake Gemerek (Member, IEEE) received the B.S. degree in mechanical and aerospace engineering from the University at Buffalo, Buffalo, NY, USA, in 2016, and the M.S. and Ph.D. degrees in mechanical engineering from Cornell University, Ithaca, NY, USA, in 2020.

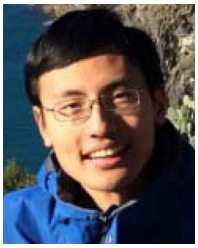
His Ph.D. research focused on active vision, perception, and planning for resource constrained autonomous systems equipped with directional sensors. He was a Research Assistant with the Laboratory for Intelligent Systems and Controls, Cornell University. He is currently a Systems Engineer with the Technology and Advanced Pursuits Group, Moog Inc., Elma, NY, USA, where he develops state-of-the-art perception and autonomy systems.

Bo Fu (Member, IEEE) received the Ph.D. degree in mechanical and aerospace engineering from the University of California, Davis, Davis, CA, USA, in 2016.

He is the founder of Oiler (Oiler Equation, Inc.), a startup company focused on delivering the next-generation innovative and intelligent gas detection solutions. He was a Postdoctoral Associate and Visiting Scientist with the Laboratory for Intelligent Systems and Controls, Cornell University, Ithaca, NY, USA, where his work focused on machine learning, computer vision, and image sciences. His current research focuses on video-based real-time detection algorithms.

Yucheng Chen received the bachelor's degree in aeronautics and astronautics and a minor in computer science from Purdue University, West Lafayette, IN, USA, in 2017. He is currently working toward the Ph.D. degree in the Laboratory for Intelligent Systems and Controls, Cornell University, Ithaca, NY, USA.

He is currently working on human decision modeling and its applications to robots. His research interests include machine learning and optimization.



Zeyu Liu received the B.S. degree (*magna cum laude*) in mechanical engineering from Tongji University, Shanghai, China, and Politecnico di Milano, Milan, Italy, in 2016. He worked on his M.S. degree with the Laboratory for Intelligent Systems and Controls, Cornell University, Ithaca, NY, USA.

His research interests include probabilistic reasoning, optimal control, computer vision, and machine learning, with a focus in unmanned ground vehicles.

Min Zheng received the B.S. degree in mechanical engineering from the University of Southern California, Los Angeles, CA, USA, in 2016, and the M.S. degree in mechanical engineering from Cornell University, Ithaca, NY, USA, in 2018.

She was a student with the Laboratory for Intelligent Systems and Controls, Cornell University. She is currently working as a Software Engineer in the industry. Her research focuses on autonomous vehicle.

David van Wijk received the B.S. degree in mechanical engineering from Cornell University, Ithaca, NY, USA, in 2021. He is currently working toward the Ph.D. degree with the Land, Air and Space Robotics Laboratory, Texas A&M University, College Station, TX, USA.

He was an Undergraduate Researcher with the Laboratory for Intelligent Systems and Controls, Cornell University. His research interests include robotics, autonomous systems, and machine learning.

Silvia Ferrari (Senior Member, IEEE) received the B.S. degree in aerospace engineering from Embry–Riddle Aeronautical University, Daytona Beach, FL, USA, in 1997, and the M.A. and Ph.D. degrees in mechanical and aerospace engineering from Princeton University, Princeton, NJ, USA, in 2002.

She was a Professor of Engineering and Computer Science with Duke University, and the Founder and Director of the NSF Integrative Graduate Education and Research Traineeship (IGERT) and Fellowship Program on Wireless Intelligent Sensor Networks (WISeNet). She is currently a John Brancaccio Professor of Mechanical and Aerospace Engineering with Cornell University, Ithaca, NY, USA. She is also the Director of the Laboratory for Intelligent Systems and Controls (LISC), Cornell University, and the Co-Director of the Cornell-Unibo Veho Institute on Vehicle Intelligence, Cornell Tech. She is the co-author of the book *Information-driven Path Planning and Control* (MIT Press, 2021), and of the TED talk "Do robots dreams of electric sheep?". Her principal research interests include active perception, robust adaptive control, learning and approximate dynamic programming, and control of multiscale dynamical systems.

Dr. Ferrari is a Fellow of ASME, Associate Fellow of AIAA, and a Member of SPIE and SIAM. She was the recipient of the ONR Young Investigator Award in 2004, the NSF CAREER Award in 2005, the Presidential Early Career Award for Scientists and Engineers (PECASE) Award in 2006, and the Cornell University Award for Research Excellence in 2020, to name a few.