MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Coordinated aerial inspection of infrastructure with heterogeneous drones

Folorunsho, Samuel; Hayner, Christopher; Di Cairano, Stefano; Vinod, Abraham P.

TR2026-101 July 02, 2026

Abstract

We consider coordinated aerial inspection of large-scale infrastructure using a team of explorer (LiDAR-equipped) and photographer (camera-equipped) drones. We propose a hierarchical framework that combines constrained trajectory generation with multi-agent task assignment to perform inspection efficiently and safely. A key feature of our approach is the use of sequential convex programming for photographer trajectory generation under dynamics, perception, and communication constraints. We also validate our approach in a ROS2/PX4 high-fidelity simulator that extends an existing ROS1-based CARIC benchmark (Cao et al., 2025). The proposed approach typically achieves shorter scoring duration than a baseline A*-based planner while maintaining comparable inspection quality and balanced drone utilization. The code for this work is publicly available at https://github.com/merlresearch/ros2_caric.

World Congress of the International Federation of Automatic Control (IFAC) 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Coordinated aerial inspection of infrastructure with heterogeneous drones

Samuel Folorunsho[1] [*] **Christopher Hayner**[1] [**]
Stefano Di Cairano [***] **Abraham P. Vinod**[2] [***]

[*] *The University of Illinois, Urbana-Champaign, IL, USA*
[**] *The University of Washington, Seattle, WA, USA*
[***] *Mitsubishi Electric Research Laboratories, Cambridge, MA, USA*

Abstract: We consider coordinated aerial inspection of large-scale infrastructure using a team of explorer (LiDAR-equipped) and photographer (camera-equipped) drones. We propose a hierarchical framework that combines constrained trajectory generation with multi-agent task assignment to perform inspection efficiently and safely. A key feature of our approach is the use of sequential convex programming for photographer trajectory generation under dynamics, perception, and communication constraints. We also validate our approach in a ROS2/PX4 high-fidelity simulator that extends an existing ROS1-based CARIC benchmark (Cao et al., 2025). The proposed approach typically achieves shorter scoring duration than a baseline A*-based planner while maintaining comparable inspection quality and balanced drone utilization. The code for this work is publicly available at `https://github.com/merlresearch/ros2_caric`.

1. INTRODUCTION

Regular infrastructure inspection is essential to ensure public safety and prevent disasters. Traditionally, such inspections rely on manual methods where human inspectors access facades via scaffolding or rope, and such methods are labor-intensive, time-consuming, and expose the inspectors to significant risks. Recently, drones have emerged as a promising alternative, capable of collecting high-resolution visual and sensor data while reducing human exposure to hazards. However, most of the current drone-based inspection systems still depend on manual piloting, limiting their scalability and reliability (Rakha and Gorodetsky, 2018). For autonomous *single* drone-based inspection, methods such as metaheuristic optimization (Bircher et al., 2015), model-based viewpoint planning (Lyu et al., 2023), hierarchical approaches (Zhou et al., 2021), and sequential convex programming (Hayner et al., 2025b) have enabled the generation of smooth drone trajectories under constraints. Some approaches even allow real-time replanning based on online 3D reconstructions to ensure coverage (Bircher et al., 2018; Zhou et al., 2021). However, using a single drone is often inefficient for inspecting large structures. Additionally, single drones may have limited sensing capabilities due to weight restrictions.

Using multiple drones with heterogeneous capabilities operating in parallel can overcome some of the limitations discussed above. To tackle the resulting coordination and task allocation problems, recent approaches utilize multi-vehicle routing and task-assignment (Michael et al., 2008; Wei and Zhao, 2022). However, they may have a high computational cost and may require detailed environmental models that may not be available *a priori*. Consequently, there has been a shift towards fast, adaptive, online coordination strategies that allow drones to gather information during the mission under constraints (Cao et al., 2025; Liu et al., 2022; Bai et al., 2025). For example, Bai et al. (2025) generates waypoints that coordinate multiple robots under line-of-sight constraints, while using relatively simple kinematic models. Cao et al. (2025) discusses lessons learned from a multi-robot inspection challenge, emphasizing the importance of robust communication, adaptive task allocation, and efficient trajectory planning for high-quality inspection. In this work, we focus on high-fidelity trajectory generation for camera-equipped drones tasked with infrastructure inspection, under perception and physical constraints (full 6-degree of freedom dynamics for drones, camera perception, and communication constraints).

This work *proposes an efficient, multi-agent inspection framework that combines continuous-time trajectory optimization with multi-agent task assignment.* Specifically, we adapt a sequential convex programming-based trajectory optimization (Hayner et al., 2025b) to design fast and safe trajectories for camera-equipped drones that respect safety, dynamics, and perception constraints arising from the inspection task. In parallel, we utilize a task assignment strategy that efficiently allocates inspection tasks to the camera-equipped drones with limited computational burden. We validate our framework using a high-fidelity simulation in a ROS2/PX4 framework (PX4, 2025), that extends the CARIC benchmark (Cao et al., 2025) in ROS1 (which has reached its end-of-life) to ROS2. We demonstrate that the proposed continuous-time trajectory optimization significantly reduces scoring duration while maintaining comparable inspection quality relative to a standard graph-based path-planning baseline.

2. PRELIMINARIES

Notation: For a finite set $\mathcal{S}$, we denote its cardinality by $|\mathcal{S}|$. For integers $a \leq b$, we use $\mathbb{N}_{[a,b]} = \{k \in \mathbb{N} \mid a \leq$

[1] This work was completed during the internships of Mr. Folorunsho and Mr. Hayner at Mitsubishi Electric Research Laboratories.
[2] Corresponding author. Email: `abraham.p.vinod@ieee.org`

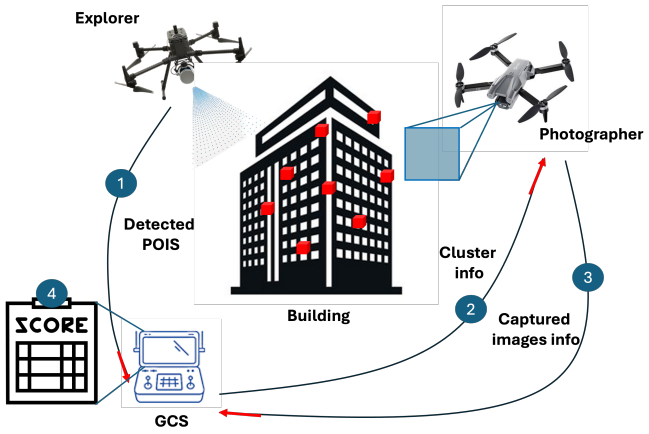

Fig. 1. Overview of the problem

$k \leq b\}$ to denote the set of natural numbers between (and including) a and b. We use the unit quaternion, $q_{A\to B} \in \mathbb{S}^3$ to parameterize the attitude of the frame $\mathcal{B}$ with respect to the frame $\mathcal{A}$, where $\mathbb{S}^3 \subset \mathbb{R}^4$ is the unit 3-sphere. We use the scalar-first convention for unit quaternion, $q = [q_w \; q_x \; q_y \; q_z]^\top$ (Hayner et al., 2025b).

2.1 Problem formulation

Similar to Cao et al. (2025), we consider the problem of inspecting a large-scale infrastructure using a team of heterogeneous drones with different sensing capabilities (see Figure 1). The team consists of two types of drones: explorers equipped with 3D LiDAR and RGB cameras for mapping and detecting points-of-interest (POIs), and photographers carrying high-resolution RGB cameras for detailed inspection of detected POIs. The quality of inspection is evaluated based on several criteria such as: 1) task completion time, 2) image quality (e.g., resolution, blur, and area of POI visible), and 3) team utilization. A ground control station (GCS) coordinates the team, and communication between the drones is possible only in the proximity of the station. Our objective is to design a coordinated strategy for the drone team that simultaneously maximizes inspection quality and minimizes mission time.

2.2 Modeling

Team of drones: We consider a heterogeneous team consisting of N_E explorer drones and N_P photographer drones. We define the index sets $\mathcal{E} \triangleq \{1, \ldots, N_E\}$, $\mathcal{P} \triangleq \{1, \ldots, N_P\}$, and $\mathcal{D} \triangleq \mathcal{E} \cup \mathcal{P}$. We control the explorer drones $\mathcal{E}$ via standard waypoint tracking for their navigation and mapping tasks (Cao et al., 2025).

A key focus of this paper is on designing continuous-time trajectories for the photographer drones under perception constraints to achieve high-quality inspection. Specifically, we model each photographer drone $i \in \mathcal{P}$ as a rigid body with 6 degrees of freedom (DoF) in a 3D space, with state,

$$x_i = [p_{i,\mathcal{I}}^\top \; v_{i,\mathcal{I}}^\top \; q_{i,\mathcal{B}\to\mathcal{I}}^\top \; \omega_{i,\mathcal{B}}^\top \; t]^\top \in \mathbb{R}^{14}. \tag{1}$$

In (1), $p_{i,\mathcal{I}} \in \mathbb{R}^3$ is the position of the drone i resolved in the inertial frame $\mathcal{I}$, $v_{i,\mathcal{I}} \in \mathbb{R}^3$ is the corresponding translational velocity resolved in the inertial frame $\mathcal{I}$, $q_{i,\mathcal{B}\to\mathcal{I}} \in \mathbb{S}^3$ is the unit quaternion representing attitude of the body frame $\mathcal{B}$ of drone i relative to the inertial frame $\mathcal{I}$, $\omega_{i,\mathcal{B}} \in \mathbb{R}^3$ is the corresponding angular velocity, and t is the scalar denoting the time. We include time t in (1) to facilitate time-optimal motion planning via time dilation (see Hayner et al. (2025b) or Section 3.3 for more details). We drop the subscript i when it is clear from the context.

The drone i has the following control inputs,

$$u = [f_\mathcal{B}^\top \; M_\mathcal{B}^\top \; s]^\top \in \mathbb{R}^7, \tag{2}$$

where $f_\mathcal{B} \in \mathbb{R}^3$ is the thrust vector, $M_\mathcal{B} \in \mathbb{R}^3$ is the moment vector, and $s \in \mathbb{R}, s \geq 0$ is the scalar input for time dilation. The continuous-time nonlinear dynamics of the drone are given by,

$$\dot{x}(t) = F(x(t), u(t)), \tag{3}$$

where $F : \mathbb{R}^{14} \times \mathbb{R}^7 \to \mathbb{R}^{14}$ is defined as,

$$\begin{aligned}
\dot{p}_\mathcal{I}(t) &= v_\mathcal{I}(t), && \text{(Position)}\\
\dot{v}_\mathcal{I}(t) &= \frac{1}{m}\left(C(q_{\mathcal{B}\to\mathcal{I}}(t)) f_\mathcal{B}(t)\right) + g_\mathcal{I}, && \text{(Velocity)}\\
\dot{q}_{\mathcal{I}\to\mathcal{B}}(t) &= \frac{1}{2}\Omega(\omega_\mathcal{B}(t)) q_{\mathcal{I}\to\mathcal{B}}(t), && \text{(Attitude)}\\
\dot{\omega}_\mathcal{I}(t) &= J_\mathcal{B}^{-1}\left(M_{\mathcal{B}(t)} - [\omega_\mathcal{B}(t)\times] J_\mathcal{B}\omega_\mathcal{B}(t)\right), && \text{(Angular vel.)}\\
\dot{t} &= s. && \text{(Time dilation)}
\end{aligned}$$

Here, $g_\mathcal{I}$ is the gravity of Earth expressed in the inertial reference frame, $J_\mathcal{B}$ is the inertial tensor of the drone expressed in the body frame, the operator $C : \mathbb{S}^3 \to \mathbb{SO}(3)$ represents the direction cosine matrix (DCM) with $\mathbb{SO}(3)$ denoting the special orthogonal group. For a vector $\xi \in \mathbb{R}^3$, the skew-symmetric operators $\Omega(\xi)$ and $[\xi\times]$ are defined in (Hayner et al., 2025b).

Workspace and POIs: The mission takes place in a 3D workspace $\mathcal{W} \subset \mathbb{R}^3$ containing static obstacles (e.g., building structures) and a fixed ground control station (GCS) at a known position $p_{\text{GCS}} \in \mathcal{W}$. The set of points of interest (POIs) is denoted by $\mathcal{I} \triangleq \{1, \ldots, N_I\}$, where each POI $i \in \mathcal{I}$ is associated with a fixed position $p_i \in \mathcal{W}$. The locations of the POIs are unknown a priori and must be detected by the explorer drones and photographed by the photographer drones during the mission.

We consider two cases for the environment and POI models — *known-model* and *unknown-model*. In the *known-model* case, the GCS has access to an exact map of $\mathcal{W}$. The role of the explorers in such a scenario is to detect POIs quickly and trigger their inspection via the photographer drones. In the *unknown-model* case, the explorers must build the map of $\mathcal{W}$ along with the detection of POIs. Let $\mathcal{M}_t$ denote the time-varying map estimate at time t, and it encodes free and occupied regions in $\mathcal{W}$ for collision-free motion planning. In the known-model case, $\mathcal{M}_t$ is a map with known obstacle locations, and remains unchanged. In the unknown-model case, $\mathcal{M}_t$ is updated from the explorers' LiDAR data.

Communication constraints: Recall that the GCS serves as the central coordinator, and we do not allow direct communication between any pair of drones. We allow communication between any drone (explorer or photographer) $i \in \mathcal{D}$ and the GCS, only when line-of-sight and range constraints are satisfied. Let $p_i(t) \in \mathcal{W}$ denote the position of the drone $i \in \mathcal{D}$ at time t. Formally, we allow for communication between drone i and the GCS if the

following conditions are met:
i) the distance between a drone i and the GCS is less than a maximum communication range $d_{\max} > 0$,

$$d_i(t) \triangleq \|p_{\mathcal{I}}(t) - p_{\mathrm{GCS}}\|_2 < d_{\max}, \tag{4}$$

where $\|\cdot\|_2$ is the Euclidean norm, and
ii) the line segment joining p_{GCS} and $p_i(t)$ does not pass through any obstacles in $\mathcal{W}$. See the *slab method* in (Kay and Kajiya, 1986) for an efficient implementation.

As illustrated in Figure 1, the explorers rely on the above communication model to report detected POIs and the mapping estimates to the GCS, the GCS uses it to assign tasks to photographer drones, and the photographer drones use it to report inspection images back to the GCS.

Other considerations/assumptions: The GCS can assign tasks (a collection of POIs to inspect) to photographer drones only after they have been detected by the explorer and communicated to the GCS. For the unknown-model case, the explorers have access to a box containing the unknown workspace $\mathcal{W}$.

3. COORDINATED AERIAL INSPECTION

Figure 2 shows a single iteration of the proposed approach. The team continues to operate until all POIs are inspected. The explorer drones explore the environment for mapping (if needed) and detect POIs. The detected POIs are sent to the GCS whenever the explorer drone satisfies communication constraints. The GCS accumulates detected POIs until a user-specified minimum number for effective utilization of the photographer drones. The GCS also determines the visiting order of the POIs within each cluster. Finally, the GCS uses proximity of the cluster center to the available photographer drones to determine the assignment of the cluster to a drone. Here, available photographer drones are those that are within communication range of the GCS and not currently executing an assignment. Once a photographer drone receives their assigned cluster and the visiting order of the POIs within the cluster, it uses sequential convex optimization to determine a dynamically-feasible, continuous-time trajectory that respects the perception constraints while visiting all the POIs, and returns within range of the GCS for the next assignment to meet the communication constraints.

The rest of this section is organized as follows. Section 3.1 briefly describes the motion planning and control for the explorer drones. Section 3.2 describes the motion planning and controller for the photographer drones in detail. Finally, we provide the implementation details of the proposed approach in Section 3.3.

3.1 Planning and control for explorer drones

We model the workspace $\mathcal{W}$ as an axis-aligned rectangular box, $\mathrm{BoundingBox} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}] \subset \mathbb{R}^3$, expressed in the inertial frame. We divide BoundingBox into N_E subregions and assign each explorer to one subregion $\mathrm{BoundingBox}_{\mathrm{sub}}$. Within each subregion, the explorer follows a predefined zigzag path (or boustrophedon path as discussed in (Choset and Pignon, 1998)) to ensure full coverage of the structure. Our approach for the explorer drones is similar to the strategies pursued by Team XXH and Team KIOS CoE that participated in CARIC (Cao et al., 2025). Alternatively, more sophisticated exploration strategies can be employed (e.g., frontier-based exploration (Karumanchi et al., 2025)), which is beyond the scope of this paper.

We use a standard position-based waypoint-tracking controller for the explorer drones to allow them to follow the pre-defined paths. Also, we use LiDAR-based SLAM to build a local occupancy map for the unknown-model case.

3.2 Planning and control for photographer drones: Theory

On receiving a cluster assignment and the visiting order for the POIs, a photographer drone must generate a motion plan that is dynamically feasible, visit the POIs in the correct sequence, and take photos that meet the desired quality criteria. To achieve this, we decompose the trajectory generation for the drone into three phases:
i) *Phase 1: Hover-to-First-POI.* Depart from its current hover state and travel to the first POI.
ii) *Phase 2: Intra-Cluster Motion.* Visit all remaining POIs in the assigned cluster, following the visiting order, and capture high-quality POI images.
iii) *Phase 3: Return-to-LoS.* Return to a location within range of line-of-sight (LoS) communication with the GCS.
Each of these phases is formulated as a variant of the following continuous-time trajectory generation problem,

$$\begin{aligned}
&\underset{(x,u,t_f)}{\text{minimize}} && t_f && \text{(Flight time)} && \text{(5a)}\\
&\text{subject to} && && && \\
&\forall t \in [0,t_f], && \dot{x}(t) = F\big(x(t),u(t)\big), && \text{(Dynamics)} && \text{(5b)}\\
&\forall t \in [0,t_f], && x(t) \in \mathcal{X},\ u(t) \in \mathcal{U}, && \text{(State-input bounds)} && \text{(5c)}\\
& && x(0) = x_i,\ x(t_f) \in \mathcal{X}_f, && \text{(Boundary cond.)} && \text{(5d)}\\
& && \text{Other constraints on } x. && && \text{(5e)}
\end{aligned}$$

The optimal control problem in (5) minimizes the flight time t_f subject to the nonlinear drone dynamics, state-input bounds, and boundary conditions. Here, $\mathcal{X}$ and $\mathcal{U}$ are the sets of allowable states and inputs, respectively, encoding operational limits such as position bounds, maximum speeds, and actuator saturations that are enforced at all times. The set $\mathcal{X}$ also encodes obstacle avoidance constraints based on the map $\mathcal{M}_t$. The initial state x_i is the current state of the drone. The terminal set $\mathcal{X}_f$ encodes phase-specific terminal conditions. The "Other constraints on x" collect phase-specific state constraints, such as perception constraints, as needed.

We now discuss how the boundary conditions $\mathcal{X}_f$ in (5d) and other constraints in (5e) are instantiated for each of the three phases (hover-to-first-POI, intra-cluster motion, and return-to-LoS). Let $\mathcal{C} = \{r^{(1)}, \ldots, r^{(N_C)}\}$ denote the assigned cluster of N_C POIs, ordered to minimize travel time, where $r^{(j)} \in \mathbb{R}^3$ is the position of the j-th POI.

Phase 1: Hover-to-First-POI. In this phase, we require the photographer drone to depart from its current hover state and travel to the first POI in the assigned cluster. Consequently, we require the drone to terminate Phase 1 at the first POI with zero velocity, and set

$$\mathcal{X}_f = \big\{\, x \,\big|\, p_{\mathcal{I}} = r^{(1)}, v_{\mathcal{I}} = 0 \,\big\}. \tag{6}$$

Phase 1 requires no additional constraints beyond the obstacle avoidance and state/input limits in $\mathcal{X}$ and $\mathcal{U}$.

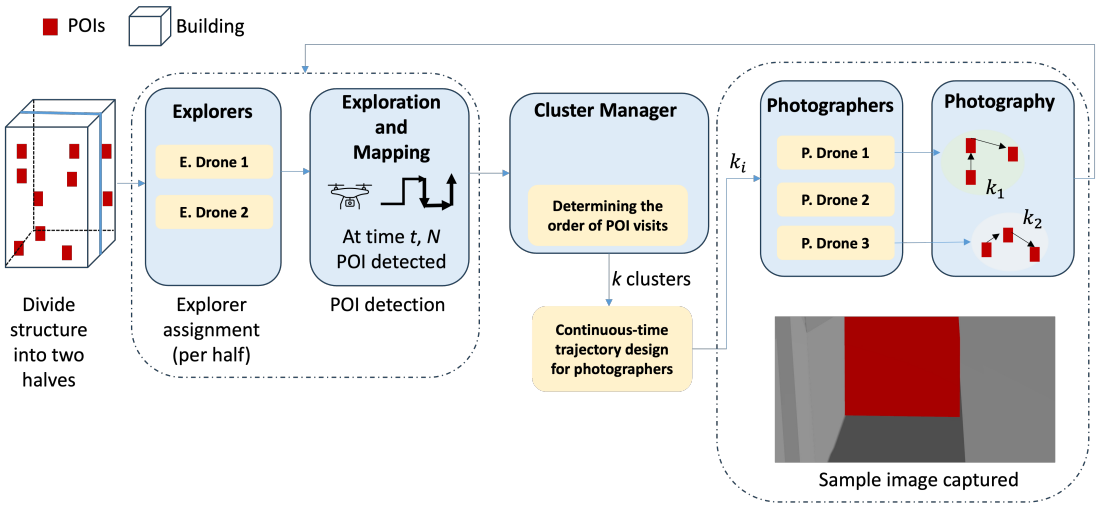

Fig. 2. Flowchart of the iterative algorithm, which combines trajectory design with multi-agent task allocation

Phase 2: Intra-cluster motion. Phase 2 requires the photographer drone to visit all POIs in a cluster in the specified order, and capture images. Here, we require the drone to terminate Phase 2 at the last POI with zero velocity,

$$\mathcal{X}_f = \left\{ x \,\middle|\, p_{\mathcal{I}} = r^{(N_C)}, v_{\mathcal{I}} = 0 \right\}. \tag{7}$$

We impose additional constraints on x in (5e) to enforce perception constraints in Phase 2. Specifically, we impose *pointwise-in-time* (nodal) state constraints to ensure that images captured at each POI meet the desired quality criteria. Consider a finite set of time instants $\mathcal{T} \subset [0, T_f]$, $|\mathcal{T}| = 2N_C$, and $t_{|\mathcal{T}|} = t_f$. Here, $\mathcal{T}$ contains the time instants that bound the time intervals (one for each POI) during which an image of a particular POI may be captured. Then, we require:
i) *for high resolution*: the drone must be within a certain distance of the POI $r_{\mathcal{I}}^{(j)}$ at $t_k, t_{k-1} \in \mathcal{T}$,
ii) *for low motion blur*: the drone must have a bounded velocity at $t_k, t_{k-1} \in \mathcal{T}$, and
iii) *for good visibility of POI*: the drone must have a camera orientation that points towards the POI $r_{\mathcal{I}}^{(j)}$ at $t_k, t_{k-1} \in \mathcal{T}$ to ensure visibility.

The first two constraints are convex constraints on position and velocity enforced at $\tau \in \{t_k, t_{k-1}\}$,

$$\left\| p_{\mathcal{I}}(\tau) - r_{\mathcal{I}}^{(j)} \right\|_2 \leq \varepsilon_{\text{pos}}, \qquad \left\| v_{\mathcal{I}}(\tau) \right\|_2 \leq \varepsilon_{\text{vel}}, \tag{8}$$

with $r^{(j)}$ denoting the position of j^{th} POI, and user-specified tolerances $\varepsilon_{\text{pos}}, \varepsilon_{\text{vel}} > 0$.

The third constraint is a non-convex constraint on the position $p_{\mathcal{I}}$ and the orientation $q_{\mathcal{B}\to I}$. We use the approach described in Hayner et al. (2025b) to encode the visibility constraint. Let $R_{\mathcal{S}\to\mathcal{B}} \in \mathbb{SO}(3)$ denote the orientation of the camera in the body frame (considered fixed and known for this work). Then, defining $C(q_{\mathcal{S}\to\mathcal{I}}(t)) = R_{\mathcal{S}\to\mathcal{B}} C(q_{\mathcal{B}\to\mathcal{I}}(t))$, we can ensure good visibility of the POI, when

$$\begin{aligned} &\left\| A_{\text{LOS}} C(q_{\mathcal{S}\to\mathcal{I}}(t)) \left(r_{\mathcal{I}}^{(j)} - p_{\mathcal{I}}(t) \right) \right\|_\infty \\ &\quad \leq c_{\text{LOS}}^\top C(q_{\mathcal{S}\to\mathcal{I}}(t)) \left(r_{\mathcal{I}}^{(j)} - p_{\mathcal{I}}(t) \right). \end{aligned} \tag{9}$$

Here, $(A_{\text{LOS}}, c_{\text{LOS}})$ are characterized by the camera field-of-view, with $c_{\text{LOS}} = [0\ 0\ 1]^\top$, and

$$R_{\mathcal{S}\to\mathcal{B}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad A_{\text{LOS}} = \begin{bmatrix} \frac{1}{\tan(\alpha_x)} & 0 & 0 \\ 0 & \frac{1}{\tan(\alpha_y)} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{10}$$

with α_x and α_y being half the horizontal and vertical field-of-view angles of the camera, respectively. As discussed in (Hayner et al., 2025b), ℓ_∞-norm accounts for the rectangular camera frame. Informally, the constraint (9) requires the vector joining the camera to the POI to lie within the camera's field of view after suitable transformations. The constraint (9) is non-convex due to the multiplication of the rotation matrix $C(q_{\mathcal{S}\to\mathcal{I}})$ with the drone position $p_{\mathcal{I}}$.

We enforce (8)–(9) at appropriate instants in $\mathcal{T}$ as part of (5e). While solving (5) for Phase 2, $\mathcal{T}$ may be scheduled as fixed, pre-determined time instants, or optimized via time dilation (see Section 3.3).

Phase 3: Return-to-line-of-sight. In this phase, we require the photographer drone to depart from the last POI in its assigned cluster and travel back to the communication range of the GCS. Consequently, we require the drone to terminate Phase 3 while satisfying the communication requirements, and set

$$\mathcal{X}_f = \left\{ x \,\middle|\, \|p_{\mathcal{I}} - p_{\text{GCS}}\|_2 \leq d_{\text{COMM}}, v_{\mathcal{I}} = 0 \right\}. \tag{11}$$

To simplify the implementation, we choose $d_{\text{COMM}} < d_{\max}$ such that all positions inside $\mathcal{X}_f$ are valid for communication. Phase 3 requires no additional constraints beyond $\mathcal{X}$ and $\mathcal{U}$, similarly to Phase 1.

3.3 Implementation details

We implement the trajectory generation for all three photographer phases using the continuous-time successive convexification (CT-SCvx) framework of (Hayner et al., 2025b), as implemented in `OpenSCvx` (Hayner et al., 2025a). CT-SCvx is an iterative method that solves a sequence of discrete-time, convex approximations to the continuous-time non-convex optimal control problem in (5), converging to a locally optimal solution. The

framework supports nonlinear dynamics, continuous-time state constraints, and nodal convex/non-convex state-input constraints, making it well-suited for our application. See (Hayner et al., 2025b) for further details on the CT-SCvx algorithm, and its implementation in `OpenSCvx`.

To handle the nodal perception constraints in Phase 2, we leverage the time dilation feature of `OpenSCvx`. Time dilation uses $\dot{t} = s$ to allow the solver to adjust the timing of the trajectory at specified nodes, thereby permitting a non-uniform time discretization which typically leads to better satisfaction of the nodal constraints. Additionally, we use first-order hold interpolation for the control inputs.

The obstacles in $\mathcal{M}_t$ are represented as bounding boxes in the inertial frame. We enforce obstacle avoidance as a differentiable, continuous-time constraint using a log-sum-exp approximation. See (Liu et al., 2022) for more details.

Given a solution to the optimal control problem (5), we use (acceleration-based) trajectory tracking for the photographer drone. Specifically, we set up the drone to track a trajectory characterized by a time-stamped sequence of waypoints,

$$\left\{p_{\mathcal{I}}(t_k),\, v_{\mathcal{I}}(t_k),\, a_{\mathcal{I}}(t_k),\, \psi(t_k),\, \dot{\psi}(t_k)\right\}_{k=0}^{N}, \tag{12}$$

where t_k are discrete time samples (nodes) along the trajectory with $t_0 = 0$ and $t_N = t_f$, $p_{\mathcal{I}}(t_k)$, $v_{\mathcal{I}}(t_k)$, and $a_{\mathcal{I}}(t_k)$ are the position, velocity, and acceleration in the inertial frame, and $\psi(t_k)$ and $\dot{\psi}(t_k)$ are the yaw angle and yaw rate. For ease of implementation, we concatenate the trajectories from the three phases and track the resulting full trajectory using the same tracking controller. The concatenation is justified by the fact that each phase ends with the drone at rest (zero velocity), ensuring smooth transitions between phases.

4. HIGH-FIDELITY SIMULATOR: A ROS2 EXTENSION OF CARIC

To evaluate the proposed framework, we developed a high-fidelity ROS2 simulator inspired by the CARIC benchmark (Cao et al., 2025). CARIC uses realistic modeling of multi-drone inspection tasks, including sensing, communication, and scoring metrics that closely mimic real-world scenarios. However, given that CARIC's original software stack is based on ROS1 and `RotorS` (ETHZ-ASL, 2021) and both of these software tools have reached their end-of-life, we undertook a comprehensive re-implementation to ensure compatibility with modern tools like ROS2 and `OpenSCvx` (Hayner et al., 2025a).

Our high-fidelity simulation environment, which will be published at `https://github.com/merlresearch/ros2_caric`, is developed in `gz-sim` (Harmonic) with PX4 (PX4, 2025) and a fully ROS2-native communication graph. We reuse one of the CARIC environments, Marina Bay Sands, using the original bounding boxes and world scales, and develop a new Powerline environment as part of this work. We control the drones through PX4's offboard interface, and process the inspection data in real time and under communication constraints through custom ROS2 nodes. We retained the original CARIC scoring for inspection, which simulates a pinhole camera with configurable resolution, field of view, and motion blur effects (Cao et al., 2025).

While we have preserved the core principles and evaluation metrics of CARIC, we have also introduced enhancements to better suit our framework's requirements. For every POI $j \in \mathcal{I}$, let k be the number of photos taken by the team. The total score upon completion of the inspection task is $Q = \sum_{j\in\mathcal{I}} \max_{k\in\{1,\dots,N\}} q_{j,k}$, where we consider the best inspection quality score $q_{j,k}$ for each POI across all photos taken by the team. We formally define the inspection quality score as the product of four metrics, similarly to Cao et al. (2025),

$$q_{j,k} = q_{\text{seen}} \cdot q_{\text{blur}} \cdot q_{\text{res}} \cdot q_{\text{area}}, \tag{13}$$

- *Seen metric* $q_{\text{seen}} \in \{0, 1\}$, which is a binary score indicating whether the POI is included in the field-of-view of the photographer's camera,
- *Motion blur metric* $q_{\text{blur}} \in [0, 1]$, approximated analytically using pixel displacement during exposure that depends on the drone's speed and direction when capturing the image,
- *Resolution metric* $q_{\text{res}} \in [0, 1]$, computed from the depth-dependent ground sampling distance (GSD) that depends on the camera parameters and the distance to the POI, and
- *Visible-area metric* $q_{\text{area}} \in [0, 1]$, which measures the fraction of the POI's projected face that lies within the camera image plane.

Note that the first three metrics are directly adopted from CARIC (Cao et al., 2025), while the last metric is newly introduced in this work to better capture the visibility of POIs in our simulator. We implemented POIs as small mesh cubes, which we believe is more representative of real-world inspection tasks. The visible-area metric addresses scenarios where a POI may be technically within the camera's field-of-view but only marginally visible at the edge of the frame. By design, the maximum total score Q a team can obtain is the number of POIs.

5. RESULTS

We validate our framework in the high-fidelity simulator using two environments (MBS and Powerline). For each environment, we consider both scenarios — known-model and unknown-model. For each of these scenarios, we vary the number of POIs (10 and 50) to assess the scalability of our approach. To account for possible variations across runs, we ran each experiment three times and reported the median values. All simulations were run on a standard desktop computer (Intel i9-12900KF CPU, 24 cores, 64 GB RAM) running Ubuntu 22.04. For the validation, we used: $N_E = 2$, $N_P = 3$, $d_{\max} = 50$ m, $d_{\text{COMM}} = 15$ m, $\beta = 20.0$, $\alpha_x = 80°$, $\alpha_y \approx 45°$, $\varepsilon_{\text{pos}} = 0.5$ m, and $\varepsilon_{\text{vel}} = 0.1$ m/s. We also require the number of points in each cluster $N_C \geq \min(3, N_{\text{not-visit}})$, where $N_{\text{not-visit}}$ is the number of POIs that are yet to be visited by any photographer drone.

We compare the proposed continuous-time trajectory generation method for the photographer drones with a standard graph-based planner (A*) that plans over a grid world. Here, the A*-based plan yields waypoints for the photographer drone that is then tracked via standard waypoint tracking. Due to its inherent "stop-and-go" nature, we typically observed A* yielding a high score at the cost of higher overall inspection time.

Table 1. Inspection performance comparison across models (MBS, Powerline), trajectory generation methods for the photographer (A* and our approach), a priori known vs unknown world, and varying POI counts N_I. We report the median values across 3 runs.

Model	N_I	World	Method	Score $[0,1]$	Scoring time (s)	Mission time (s)	Compute (s)
MBS	10	Known	A*	0.867	342.6	397.4	0.217
			Ours	0.981	166.4	215.4	4.984
		Unknown	A*	1.000	219.5	260.2	7.019
			Ours	0.892	99.2	485.9	3.230
	50	Known	A*	1.000	1096.8	1193.8	1.007
			Ours	0.988	450.8	537.6	20.092
		Unknown	A*	1.000	616.8	670.2	21.533
			Ours	0.894	483.1	670.1	12.899
Power line	10	Known	A*	1.000	375.4	414.4	0.536
			Ours	0.893	270.2	297.9	2.511
		Unknown	A*	1.000	323.4	354.8	3.751
			Ours	0.911	255.9	310.1	3.363
	50	Known	A*	1.000	944.2	999.3	0.857
			Ours	0.903	262.0	708.4	16.791
		Unknown	A*	1.000	521.3	558.7	11.668
			Ours	0.942	449.0	749.5	13.901

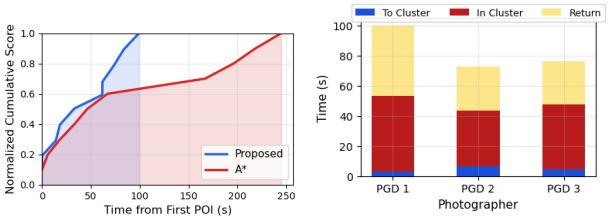

Fig. 3. (Left) Score evolution. (Right) Per-drone work time and phase breakdown.

Table 1 reports performance metrics across environments, prior knowledge settings, POI counts, and methods. We use two temporal metrics: (1) scoring duration, the time from the first POI detection to the last home return; and (2) mission duration, the total time from the first drone takeoff to the last home return. While the baseline generally achieves a perfect score (1.0) due to its inherent stop-and-go nature, the proposed approach achieves a comparable score (> 0.89) in a significantly shorter scoring duration. The proposed approach typically has a higher per-assignment compute time, possibly due to the use of non-optimized Python code to solve (5). Consequently, the mission duration, which includes setup times, for the proposed approach is sometimes higher than the baseline.

Additionally, we analyze the temporal evolution of the cumulative score, the workload distribution among the photographer drones, and the inter-drone separation for the MBS unknown-model case with 10 POIs. Figure 3 (left) illustrates how the cumulative score evolves over the course of the mission. Here, the proposed method achieves a relatively high score significantly faster than the Baseline A*, which we attribute to the use of continuous-time trajectory optimization. Figure 3 (right) shows how the drones were utilized over the course of the inspection in this scenario. As expected, the proposed method's use of clustering and task assignment distributes the workload fairly evenly. Here, the per-drone breakdown includes total work time and the time spent in each phase (travel to cluster, in-cluster imaging, and return). Figure 4 shows the minimum and average of the pairwise separation $d_{ij}(t) = \left\|p_i(t) - p_j(t)\right\|_2$ over all photographer drone pairs over the duration of the mission. The mission is collision-free with separation remaining above 1m throughout.

6. CONCLUSION

In this paper, we presented an approach for coordinated aerial inspection of infrastructure with heterogeneous drones using continuous-time trajectory optimization and task assignment. We formulated the motion planning problem for the photographer drones as a continuous-time optimal control problem that can be solved using a sequential convex programming framework. We validated our approach in a high-fidelity ROS2/PX4 simulator inspired by the CARIC benchmark, demonstrating effective inspection performance, balanced workload distribution, and safe inter-drone operation.

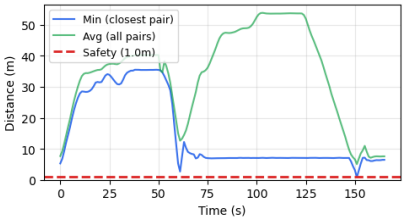

Fig. 4. Inter-drone separation over time.

REFERENCES

Bai, R. et al. (2025). Realm: Real-time line-of-sight maintenance in multi-robot navigation with unknown obstacles. In *IEEE Int'l Conf. Rob. Autom.* (accepted).

Bircher, A. et al. (2015). Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *IEEE Int'l Conf. Rob. Autom.*, 6423–6430.

Bircher, A. et al. (2018). Receding horizon path planning for 3D exploration and surface inspection. *Auto. Robots*, 42(2).

Cao, M. et al. (2025). Cooperative aerial robot inspection challenge: A benchmark for heterogeneous multi-uncrewed-aerial-vehicle planning and lessons learned. *IEEE Rob. Auto. Mag.*

Choset, H. and Pignon, P. (1998). Coverage path planning: The boustrophedon cellular decomposition. In *Field Serv. Robot.*

ETHZ-ASL (2021). RotorS. `https://github.com/ethz-asl/rotors_simulator`. Accessed: 2025-12-03.

Hayner, C. et al. (2025a). OpenSCvx. `https://github.com/haynec/OpenSCvx`. Accessed: 2025-12-03.

Hayner, C.R., Carson, J., Açıkmeşe, B., and Leung, K. (2025b). Continuous-time line-of-sight constrained trajectory planning for 6-degree of freedom systems. *IEEE Rob. Auto. Lett.*

Karumanchi, S., Rokaha, B., Schperberg, A., and Vinod, A. (2025). Energy-constrained multi-robot exploration for autonomous map building. In *IEEE Int'l Conf. Intell. Robots Syst.*

Kay, T.L. and Kajiya, J.T. (1986). Ray tracing complex scenes. *ACM Comp. Graph.*, 20(4), 269–278.

Liu, T. et al. (2022). Star-convex constrained optimization for visibility planning with application to aerial inspection. In *IEEE Int'l Conf. Rob. Autom.*

Lyu, Y., Cao, M., Yuan, S., and Xie, L. (2023). Vision-based plane estimation and following for building inspection with autonomous UAV. *IEEE Trans. Syst., Man, and Cyber.*, 53(12), 7475–7488.

Michael, N., Zavlanos, M.M., Kumar, V., and Pappas, G.J. (2008). Distributed multi-robot task assignment and formation control. In *IEEE Int'l Conf. Rob. Autom.*

PX4 (2025). PX4-Autopilot. `https://github.com/PX4/PX4-Autopilot`. Accessed: 2025-12-03.

Rakha, T. and Gorodetsky, A. (2018). Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones. *Autom. Constr.*

Wei, Z. and Zhao, X. (2022). Multi-UAVs cooperative reconnaissance task allocation under heterogeneous target values. *IEEE Access.*

Zhou, B., Zhang, Y., Chen, X., and Shen, S. (2021). FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Rob. Auto. Lett.*, 6(2), 779–786.