IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

VACNA: Visibility-Aware Cooperative Navigation With Application in Inventory Management

Houman Masnavi, *Graduate Student Member, IEEE*, Jatan Shrestha, Karl Kruusamäe, and Arun Kumar Singh, *Member, IEEE*

***Abstract*—This letter presents an online trajectory planning algorithm for an Unmanned Aerial Vehicle (UAV) to autonomously scan warehouse racks for inventory management. Our main motivation is to make small-sized UAVs with limited computing and sensing hardware capable of reliably performing the scanning task in cluttered environments. To this end, we propose a cooperative system where an Unmanned Ground Vehicle (UGV) guides the UAV using the novel template of visibility-aware cooperative navigation (VACNA). We propose a Cross-Entropy Method (CEM) based approach for solving the trajectory optimization underpinning VACNA. In particular, our CEM projects sampled vehicle trajectories onto the constraint sets before evaluating the cost functions. We further learn a deep generative model in the form of a Conditional Variational Autoencoder (CVAE) from expert demonstrations to warm-start our optimizer. We improve the state-of-the-art in the following respects. First, we present a detailed analysis of the role of our proposed cost and constraint functions for cooperative occlusion-free navigation. Second, we compare our custom CEM optimizer with conventional variants and show significantly reduced collision and occlusion rates. Finally, our CVAE initialization allows our optimizer to operate with smaller batch sizes and achieve real-time performance even on embedded hardware devices like NVIDIA Jetson Xavier.**

***Index Terms*—Aerial systems: Applications, machine learning for robot control, optimization and optimal control.**

I. Introduction

WAREHOUSE automation has emerged as one of the prominent applications for Unmanned Aerial Vehicles (UAV) [1], [2]. Herein, the UAV needs to scan the racks of the warehouses through its monocular camera while navigating through narrow, cluttered corridors. There are two core challenges to it from the perspective of autonomous navigation. First, UAVs can carry limited computing power onboard [3]. Second, mounting sensors like LiDAR and RGBD cameras on the UAV can increase its form factor and, at the same time, also reduce its flight time. An increased UAV size may also make it unsuitable for navigation in tight indoor spaces.

Our primary motivation in this letter is to make small-sized UAVs with only monocular cameras capable of reliably performing rack scanning tasks in warehouses. Thus, this letter proposes and validates a cooperative system wherein a UAV is guided by an Unmanned Ground Vehicle (UGV). With its more significant form factor, the latter can carry more extensive computing and sensing hardware. Therefore, the UGV can plan motions for both itself and the UAV provided it can always keep the latter in its field of view (FoV).

At the core of our approach lies a novel motion planning concept that we refer to as *visibility-aware cooperative navigation (VACNA)*. It is characterized by cost and constraint functions that allow a UAV-UGV pair to navigate smoothly in cluttered environments while maintaining each other in their field of view and within a specified distance threshold. Moreover, these functions also model the primary task: UAV maintaining visibility of the rack during navigation.

We adopt a gradient-free, sampling-based optimization called Cross-Entropy Method (CEM) to solve the trajectory optimization underpinning VACNA. We propose several modifications over the baseline variant that dramatically improve the quality of cooperation between the UAV-UGV and the efficiency of the scanning task. Our main innovations and their benefits are summarised below.

Algorithmic and Empirical Contribution:

- We augment a baseline CEM [4] with a batch projection optimization to efficiently handle the constraints for the VACNA task.
- We learn a Conditional Variational Autoencoder (CVAE) to warm-start our CEM.
- Our ablation study shows that in contrast to [5], we find that both the UAV and UGV need to take responsibility for keeping their line-of-sight (LoS) occlusion-free (Section IV-F2).

State-of-the-art (SOTA) Performance:

- We show that our projection-augmented CEM outperforms the baseline variant in all benchmark metrics such as occlusion time, traversal distance, etc. Moreover, we show similar improvements over another SOTA sampling-based optimization called Covariance Matrix Adaptation Evolution Strategy (CMA-ES). We adopt the customization for CMA-ES proposed in [6] and re-implement it to leverage GPU parallelization.
- We also compare our approach with A^* algorithm-based planning and demonstrate speed-up of more than one order of magnitude in computation time.

Manuscript received 25 February 2023; accepted 9 August 2023. Date of publication 7 September 2023; date of current version 22 September 2023. This letter was recommended for publication by Associate Editor S. Mohan and Editor P. Pounds upon evaluation of the reviewers' comments. This work was supported in part by European Social Fund through IT Academy Program in Estonia and in part by Estonian Research Council under Grant PSG753. *(Corresponding author: Houman Masnavi.)*

The authors are with the University of Tartu, 50411 Tartu, Estonia (e-mail: houman.masnavi@ut.ee; jatan.shrestha@ut.ee; karl.kruusamae@ut.ee; aks1812@gmail.com).

Link to the GitHub repository: https://github.com/Houman-HM/VACNA.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3312969, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3312969

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE I
IMPORTANT SYMBOLS

$\mathbf{p}_r = (x_r(t), y_r(t))$	Position of the UAV at time t.
$\mathbf{p}_g = (x_g(t), y_g(t))$	Position of the UGV at time t.
$\mathbf{p}_{sc} = (x_{sc}(t), y_{sc}(t))$	Position of the scanning point at time t.
$\mathbf{p}_{ref} = (x_{ref}(t), y_{ref}(t))$	Position of the reference point at time t.
$\mathbf{p}_{o,i} = (x_{o,i}, y_{o,i})$	Position of the i^{th} point cloud.
n, m	Batch-size and planning horizon, resp.

- We show that the CVAE-based initialization allows for smaller batch size and consequently real-time performance even on embedded hardware devices like NVIDIA Jeston Xavier.

II. Problem Formulation and Related Works

Symbols and Notations: Normal-faced, small-case letters represent scalars, while bold-font variants denote vectors. The matrices are represented in bold-font, upper-case letters. The variables t and T are timestamp and transpose, respectively. The main symbols are summarized in Table I. Some of the symbols are also defined in their first place of appearance.

Assumptions

- The UAV and UGV are considered to be holonomic. Furthermore, we assume that the cameras orientations are independent of the robots' linear motions.
- The UAV operates in 3D but can avoid collisions or occlusions only through motions in the $X-Y$ plane.
- The main task is reduced to UAV taking occlusion-free images of the racks under the guidance of the UGV. We note that the entire inventory pipeline would have additional components but these are beyond the scope of the letter.

A. Optimization for UAV-UGV Warehouse Inventory

We can formalize VACNA between a UAV-UGV pair for scanning warehouse racks in the following manner:

$$\min \sum_{t=t_0}^{t=t_m} w_1(\ddot{\mathbf{p}}_r(t))^2 + w_1(\ddot{\mathbf{p}}_g(t))^2 + w_2 f_{oc}(\mathbf{p}_r(t), \mathbf{p}_g(t)) + w_3 f_{oc}(\mathbf{p}_r(t), \mathbf{p}_{sc}(t)) \tag{1}$$

$$(\mathbf{p}_r(t_0), \dot{\mathbf{p}}_r(t_0), \ddot{\mathbf{p}}_r(t_0)) = \mathbf{b}_{0,r}, (\dot{\mathbf{p}}_r(t_m), \ddot{\mathbf{p}}_r(t_m)) = \mathbf{b}_{f,r} \tag{2a}$$

$$(\mathbf{p}_g(t_0), \dot{\mathbf{p}}_g(t_0), \ddot{\mathbf{p}}_g(t_0)) = \mathbf{b}_{0,g}, (\dot{\mathbf{p}}_g(t_m), \ddot{\mathbf{p}}_g(t_m)) = \mathbf{b}_{f,g} \tag{2b}$$

$$f_v(\dot{\mathbf{p}}_r(t)) \leq v_r, f_a(\ddot{\mathbf{p}}_r(t)) \leq a_r \tag{2c}$$

$$f_v(\dot{\mathbf{p}}_g(t)) \leq v_g, f_a(\ddot{\mathbf{p}}_g(t)) \leq a_g \tag{2d}$$

$$f_{o,i}(\mathbf{p}_r, \mathbf{p}_{o,i}) \leq 0, f_{o,i}(\mathbf{p}_g, \mathbf{p}_{o,i}) \leq 0, \forall i \tag{2e}$$

$$\underline{s}_{los} \leq \left\| [\mathbf{p_r}(\mathbf{t}) - \mathbf{p_g}(\mathbf{t})] \right\|_2 \leq \overline{s}_{los} \tag{2f}$$

$$\underline{s}_{ref} \leq \left\| [\mathbf{p_r}(\mathbf{t}) - \mathbf{p_{ref}}(\mathbf{t})] \right\|_2 \leq \overline{s}_{ref} \tag{2g}$$

$$f_v = \|\dot{\mathbf{p}}_{(.)}(t)\|_2, f_a = \|\ddot{\mathbf{p}}_{(.)}(t)\|_2 \tag{3}$$

$$f_{o,i} = -\frac{(x_{(.)} - x_{o,i})^2}{a^2} - \frac{(y_{(.)} - y_{o,i})^2}{a^2} + 1 \leq 0 \tag{4}$$

The vector $\mathbf{p}_{(.)} = [x_{(.)}, y_{(.)}]^T, (.) \in \{r, g\}$ represents the 2D position of either of the vehicles. The first two terms in the cost function (1) minimize the sum of the acceleration magnitude for the UAV and UGV. The last two terms minimize the occlusion between UAV and UGV and the occlusion between the UAV and the rack computed at discrete time instants in the planning horizon. f_{oc} is a learned occlusion model derived in our previous work in [7]; we discuss its inner working at the end of this section. $w_1 - w_3$ are user-defined weights utilized to trade off the relative importance of each cost term. Constraints (2a)–(2b) define the initial and final boundary conditions on the trajectory for the UAV and UGV. Therefore, the vectors $\mathbf{b}_{0,r}, \mathbf{b}_{f,r}, \mathbf{b}_{0,g}, \mathbf{b}_{f,g}$ are simply stacking of initial and final positions, velocities, and accelerations for the UAV and UGV, respectively. The inequality constraints (2c) and (2d) maintain the norm of velocities and accelerations within specified bounds for the UAV and UGV, respectively. The definitions of velocity and acceleration constraints are in (3). The inequality constraints (2e) ensure collision avoidance for the UAV and UGV. The definition of $f_{o,i}$ is brought in (4), where $x_{(.)}, y_{(.)}$ are the positions of UAV/UGV and $x_{o,i}, y_{o,i}$ are the position of the i^{th} obstacle point obtained from the LiDAR. The constant a in (4) is the footprint radius of UAV/UGV. The inequality (2f) ensures that the UAV and UGV would stay within certain bounds $(\underline{s}_{los}, \overline{s}_{los})$ from each other. Likewise, the inequality (2g) makes sure that the UAV would stay within certain bounds $(\overline{s}_{ref}, \underline{s}_{ref})$ from the reference trajectory. The reference trajectory, shown in black in Fig. 1(a), provides the desired path and velocity that the UAV needs to follow to scan the warehouse racks. The tracking bounds on the reference trajectory can encode the minimum and maximum distance from which the UAV should view the racks. We also project the reference trajectory on the surface of the racks to obtain the so-called scanning trajectory $(x_{sc}(t), y_{sc}(t))$, which is utilized to compute occlusion between the UAV and the rack at any time t.

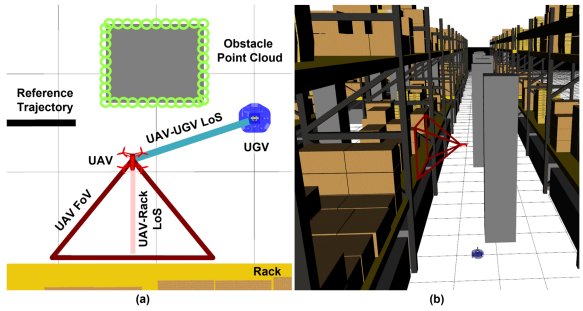

Fig. 1. (a), (b) Show top-down and third person views of a typical warehouse inventory setting.

Occlusion Model: Our learned model f_{oc} takes in a vehicle and a moving target point along with the obstacle point cloud to predict the occlusion between the two. Thus, it can predict the occlusion of the LoS between UAV-UGV. Either can be the vehicle of interest, while the other can be treated as the moving target point. We use f_{oc} to compute the occlusion between UAV and the rack by assuming $x_{sc}(t), y_{sc}(t)$ as the moving target point.

B. Related Works

Warehouse inventory using UAVs: Authors in [8] present a fast planning approach for a UAV to perform a warehouse inventory task. Their method makes use of several high-resolution cameras along with a 3D LiDAR. Furthermore, it requires prior map building of the environment. Likewise, authors in [9] present a navigation system for a UAV in a warehouse-like environment that requires a UAV equipped with two LiDARs and an RGBD camera. Although the use of a single UAV is more convenient,

it leads to a bigger form factor, as the UAV would need to have sensors like RGBD cameras and LiDARs on board for navigation and localization purposes. In addition, onboard sensing and computing can also reduce the flight time of the UAV.

In contrast, our cooperative navigation setting allows the UGV to do the heavy lifting in terms of sensing and planning for both itself and the UAV. As a result, UAVs with just a monocular camera can be used for scanning tasks. Moreover, our approach directly works with point clouds and does not require a prior map of the environment.

Cooperative Navigation: Authors in [10] propose a tethered UAV-UGV cooperative system for extended search and rescue missions. Likewise, the work in [11] presents a UAV-UGV system for exploration in confined environments coupled with a wire. In both of the works, the wire is used to match the flight time of the UAV with the UGV. However, having a physical coupling between the UAV and UGV limits the agility and maneuverability of the UAV. Our VACNA approach allows the UAV and UGV to maintain the visibility of each other with no physical connection. In some cooperative approaches, the responsibility of maintaining the visibility is taken by only one vehicle. Authors in [5] show an exploration scenario where a UAV traverses the environment while another UAV maintains visibility and avoids occlusion of their LoS. However, we show in IV-F that maintaining occlusion-free LoS becomes challenging in more cluttered environments if the vehicles do not share the responsibility of avoiding occlusions. Our proposed work is also related to [12] where a larger UAV equipped with LiDAR guides a smaller UAV with lesser sensing and compute. However, unlike our work, [12] does not address the challenge of preventing occlusions in obstacle-rich environments.

III. Main Algorithmic Results

A. Trajectory Parametrization

Optimizers like CEM typically operate by sampling in the space of trajectory waypoints [13] or control inputs [14]. We adopt a different approach, wherein we parametrize the trajectories in the space of Berenstein polynomials in the following form:

$$[x_r(t_1),..,x_r(t_m)] = \mathbf{W}\mathbf{c}_{x,r}, [y_r(t_1),..,y_r(t_m)] = \mathbf{W}\mathbf{c}_{y,r}, \tag{5}$$

$$[x_g(t_1),..,x_g(t_m)] = \mathbf{W}\mathbf{c}_{x,g}, [y_g(t_1),..,y_g(t_m)] = \mathbf{W}\mathbf{c}_{y,g}, \tag{6}$$

where $\mathbf{W}$ is a matrix formed with a time-dependent polynomial Bernstein-basis function. The vectors $\mathbf{c}_{x,r}, \mathbf{c}_{y,r}$ $\mathbf{c}_{x,g}, \mathbf{c}_{y,g}$ are the coefficients of the polynomial for the UAV and the UGV, respectively. The derivatives are also defined in terms of $\dot{\mathbf{W}}, \ddot{\mathbf{W}}$. We define $\boldsymbol{\xi}_r = (\mathbf{c}_{x,r}, \mathbf{c}_{y,r})$ and $\boldsymbol{\xi}_g = (\mathbf{c}_{x,g}, \mathbf{c}_{y,g})$ for later developments.

The polynomial representation has the following advantages. First, a long trajectory (≈ 100 steps) can be defined in terms of a handful of coefficients (≈ 22). This in turn reduces the dimensionality of the CEM optimization. Second, it ensures that any randomly generated samples of $\boldsymbol{\xi}_r, \boldsymbol{\xi}_g$ lead to continuous and differentiable trajectories.

Using (5) and (6) the following compact representations can be derived for (1)–(2e).

$$\min_{\boldsymbol{\xi}_r,\boldsymbol{\xi}_g} \frac{1}{2}\boldsymbol{\xi}_r^T w_1 \mathbf{Q}\boldsymbol{\xi}_r + w_2 \left\|\mathbf{f}_{oc}(\boldsymbol{\xi}_r, \boldsymbol{\xi}_g)\right\|_1 + w_3 \left\|\mathbf{f}_{oc}(\boldsymbol{\xi}_r, \mathbf{x}_{sc}, \mathbf{y}_{sc})\right\|_1 \tag{7}$$

$$\mathbf{A}_{eq}\boldsymbol{\xi}_r = \mathbf{b}_{eq,r}, \quad \mathbf{A}_{eq}\boldsymbol{\xi}_g = \mathbf{b}_{eq,g},$$
$$\mathbf{h}_1(\boldsymbol{\xi}_r) \leq \mathbf{0}, \quad \mathbf{h}_1(\boldsymbol{\xi}_g) \leq \mathbf{0}, \quad \mathbf{h}_2(\boldsymbol{\xi}_r, \boldsymbol{\xi}_g) \leq \mathbf{0}, \tag{8}$$

$$\mathbf{A}_{eq} = \begin{bmatrix}\mathbf{A} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}\end{bmatrix}, \mathbf{A} = \begin{bmatrix}\mathbf{W_0}|\dot{\mathbf{W}}_\mathbf{0}|\ddot{\mathbf{W}}_\mathbf{0}|\dot{\mathbf{W}}_\mathbf{m}|\ddot{\mathbf{W}}_\mathbf{m}|\end{bmatrix}^T,$$

$$\mathbf{b}_{eq,(.)} = \begin{bmatrix}\mathbf{b}_{\mathbf{0},(.)}\\ \mathbf{b}_{f,(.)}\end{bmatrix} \tag{9}$$

Constraints (8) are the matrix representations of (2a) - (2g). The algebraic form of $\mathbf{h}_1$ and $\mathbf{h}_2$ can be easily obtained through substituting (5) and (6) into (2e)–(2g). The vector $\mathbf{f}_{oc}$ is formed by stacking the f_{oc} at different time instants. The $\mathbf{W}_0, \mathbf{W}_m$ are the first and the last rows of $\mathbf{W}$. Similar notation follows for their derivatives.

B. Proposed Optimizer

1) Sequential Minimization With Warm-Start: Jointly solving (7)–(8) for $(\boldsymbol{\xi}_r, \boldsymbol{\xi}_g)$ can be computationally prohibitive. Thus, we adopt a sequential minimization approach where $\boldsymbol{\xi}_r, \boldsymbol{\xi}_g$ are optimized one at a time. This is summarized in Algorithm 1, wherein superscript l is used to track the values of variables across planning steps. For example, ${}^l\boldsymbol{\xi}_g$ represents the value of the variable obtained at planning step l. While optimizing for $\boldsymbol{\xi}_g$, we only consider the occlusion cost of UAV-UGV LoS, since there is no requirement for the UGV to maintain the visibility of the warehouse racks.

As shown in line 3–4 of Algorithm 1, while optimizing for $\boldsymbol{\xi}_r$, we fix $\boldsymbol{\xi}_g$ at the values obtained in previous step. The resulting $\boldsymbol{\xi}_r$ is then fixed in the cost and constraint functions to optimize over $\boldsymbol{\xi}_g$ in lines 5. At each optimization step, we only consider constraints that explicitly depend on the variables currently being optimized.

We use Algorithm 1 in a receding horizon manner. That is, at each time instant, we compute optimal trajectories over a horizon and then UAV/UGV executes a part of it before initiating a replanning. Under real-time restrictions, we perform only one or two iterations of Algorithm 1 but leverage the warm-start to initialize $\boldsymbol{\xi}_g$ with the optimal values obtained in the previous planning cycle.

2) CEM-Based Optimization: The optimizations in lines 2–3 and 4–5 in Algorithm 1 over $\boldsymbol{\xi}_r, \boldsymbol{\xi}_g$ are solved through a CEM-based approach. Algorithm 2 shows the process for $\boldsymbol{\xi}_r$, where the left superscript i is used to track variables across the iterations. For instance, ${}^i\boldsymbol{\mu}$ denotes the mean of the sampling distribution at iteration i. The CEM begins by drawing n samples of $\boldsymbol{\xi}_r$ from a Gaussian distribution in line 4. The j^{th} drawn sample is denoted as $\boldsymbol{\xi}_{r,j}$. An empty list is initialized in line 5 to store the cost function corresponding to the drawn samples. Lines 7–8 represent the core difference between our CEM and baseline variants in the existing literature. Herein, $\boldsymbol{\xi}_{r,j}$ is projected to $\bar{\boldsymbol{\xi}}_{r,j}$ to satisfy the boundary conditions and other inequality constraints. The cost is later evaluated for each of the projected samples in lines 9–10. $w_1 - w_3$ are user-defined weights to trade off the relative importance of the cost terms. The q samples of $\bar{\boldsymbol{\xi}}_{r,j}$ that lead to the lowest cost are selected in line 11. These samples are conventionally called the $Eliteset$. In line 12, we fit a Gaussian distribution to this set, and the resulting mean and covariance will be used for sampling in the next iteration. Although the above steps are taken to find the solution for the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Algorithm 1: Sequential Minimization with Warm-Start.

1: Input ${}^{l}\boldsymbol{\xi}_g$, at l^{th} planning step.
2: ${}^{l+1}\boldsymbol{\xi}_r = \arg\min \frac{1}{2}\boldsymbol{\xi}_r^T w_1 \mathbf{Q} \boldsymbol{\xi}_r + w_2 \mathbf{f}_{oc}(\boldsymbol{\xi}_r, {}^{l}\boldsymbol{\xi}_g) + w_3 \mathbf{f}_{oc}(\boldsymbol{\xi}_r, \mathbf{x}_{sc}, \mathbf{y}_{sc})$
3: • $\mathbf{A}_{eq}\boldsymbol{\xi}_r = \mathbf{b}_{eq,r}$, $\mathbf{h}_1(\boldsymbol{\xi}_r) \leq \mathbf{0}, \quad \mathbf{h}_2(\boldsymbol{\xi}_r, {}^{l}\boldsymbol{\xi}_g) \leq \mathbf{0}$
4: ${}^{l+1}\boldsymbol{\xi}_g = \arg\min \frac{1}{2}\boldsymbol{\xi}_g^T w_1 \mathbf{Q} \boldsymbol{\xi}_g + w_2 \mathbf{f}_{oc}(\boldsymbol{\xi}_g, {}^{l+1}\boldsymbol{\xi}_r)$
5: • $\mathbf{A}_{eq}\boldsymbol{\xi}_g = \mathbf{b}_{eq,g}$, $\mathbf{h}_1(\boldsymbol{\xi}_g) \leq \mathbf{0}, \quad \mathbf{h}_2(\boldsymbol{\xi}_g, {}^{l+1}\boldsymbol{\xi}_r) \leq \mathbf{0}$

Algorithm 2: CEM-Based Optimization.

1 N_{CEM} = Number of CEM iterations
2 Initiate mean ${}^{i}\boldsymbol{\mu}, {}^{i}\boldsymbol{\Sigma}$, at $i = 0$
3 **for** $i = 1, i \leq N_{CEM}, i++$ **do**
4 Draw n Samples $(\boldsymbol{\xi}_{r,1}, \boldsymbol{\xi}_{r,2}, \boldsymbol{\xi}_{r,j}, .., \boldsymbol{\xi}_{r,n})$ from $\mathcal{N}({}^{i}\boldsymbol{\mu}, {}^{i}\boldsymbol{\Sigma})$
5 Initialize $CostList$ = []
6 Project $\boldsymbol{\xi}_{r,j}$ to $\overline{\boldsymbol{\xi}}_{r,j}$ through the following optimization $\forall j$:
7 • $\overline{\boldsymbol{\xi}}_{r,j} = argmin_{\overline{\boldsymbol{\xi}}_{r,j}} \left\|\overline{\boldsymbol{\xi}}_{r,j} - \boldsymbol{\xi}_{r,j}\right\|_2^2$
8 • $\boldsymbol{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j} = \boldsymbol{b}_{eq,r}, \quad \mathbf{h}_1(\overline{\boldsymbol{\xi}}_{r,j}) \leq 0, \quad \mathbf{h}_2(\overline{\boldsymbol{\xi}}_{r,j}, {}^{l}\boldsymbol{\xi}_g) \leq 0$
9 $cost \leftarrow \frac{1}{2}\overline{\boldsymbol{\xi}}_{r,j}^T w_1 \mathbf{Q} \overline{\boldsymbol{\xi}}_{r,j} + w_2 \|\mathbf{f}_{oc}(\overline{\boldsymbol{\xi}}_{r,j}, \boldsymbol{\xi}_g)\|_1 + w_3 \|\mathbf{f}_{oc}(\overline{\boldsymbol{\xi}}_{r,j}, \boldsymbol{x}_{sc}, \boldsymbol{y}_{sc})\|_1$, over n samples
10 append $cost$ to $CostList$
11 $EliteSet \leftarrow$ Select top q lowest cost samples
12 $({}^{i+1}\boldsymbol{\mu}, {}^{i+1}\boldsymbol{\Sigma}) \leftarrow$ Fit Gaussian distribution $EliteSet$
13 **end**
14 **return** *Sample corresponding to the lowest* $\overline{\boldsymbol{\xi}}_{r,j}$ *in the* $EliteSet$

UAV, the same procedures apply to those of the UGV. The only difference is that the UGV's cost will not have the third term in line 9, since there is no occlusion requirement between the UGV and the warehouse racks.

3) Efficient Batch Projection: The projection optimizations in lines 7–8 of the Algorithm 2 can be done in parallel for each sample $\boldsymbol{\xi}_{r,j}$. To make this process computationally efficient, we reformulate the constraints (2c)–(2g) to induce a suitable structure in the projection optimization. The details are covered in Appendix VI but we present the core idea here. The main computation of the projection optimization can be shown to be equivalent to solving a batch of Quadratic Programs (QP) of the form (10).

$$\frac{1}{2}\overline{\boldsymbol{\xi}}_{r,j}^T \overline{\mathbf{Q}}\, \overline{\boldsymbol{\xi}}_{r,j} + \overline{\mathbf{q}}_{r,j}^T \boldsymbol{\xi}_{r,j}, \mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j} = \mathbf{b}_{eq,r}, \tag{10}$$

where only the vector $\overline{\mathbf{q}}_{r,j}$ should be recomputed for each batch. The other matrices $\overline{\mathbf{Q}}, \mathbf{A}_{eq}$ are constant across all the batches. Therefore, the solution to (10) can be computed in one shot across all the batches through (11).

$$\begin{bmatrix} \boldsymbol{\xi}_{r,j} & .. & \boldsymbol{\xi}_{r,n} \\ \boldsymbol{\nu}_{r,j} & .. & \boldsymbol{\nu}_{r,n} \end{bmatrix} = \overbrace{\left(\begin{bmatrix} \overline{\mathbf{Q}} & \mathbf{A}_{eq}^T \\ \mathbf{A}_{eq} & \mathbf{0} \end{bmatrix}^{-1}\right)}^{constant} \begin{bmatrix} \overline{\mathbf{q}}_{r,j} & .. & \overline{\mathbf{q}}_{r,n} \\ \mathbf{b}_{eq,r} & .. & \mathbf{b}_{eq,r} \end{bmatrix} \tag{11}$$

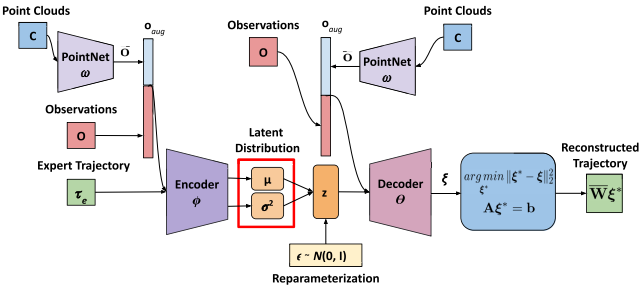

Fig. 2. Figure depicts the pipeline for our learned CVAE.

wherein $\boldsymbol{\nu}_{r,j}$ is the dual-variable associated with the equality constraints. Irrespective of the batch size, the inverse on the right-hand side of (11) needs to be computed only once. Thus the batch solution of (10) reduces to just a large matrix-vector product that can be trivially parallelized over GPUs.

C. CVAE Initialization

With naive Gaussian initialization, the CEM in Algorithm 2 may require a large batch size and many iterations to converge to an optimal solution. Thus, in this section, we adopt a data-driven approach to derive a more informed initialization for the CEM. In particular, we derive a Behaviour Cloning (BC) framework to learn a policy $\boldsymbol{\pi_\theta}$ that maps state observations $\mathbf{o}$ and LiDAR point clouds $\mathbf{c}$ to a distribution over optimal trajectory coefficients for UAV/UGV, $\boldsymbol{\xi}^* = (\boldsymbol{\xi}_r^*, \boldsymbol{\xi}_g^*)$. The observations $\mathbf{o}$ have information about the positions and velocities of the UAV/UGV.

We model $\pi_{\boldsymbol{\theta}}$ as a probabilistic generative model in the form of CVAE [15]. To train our model, we assume access to a dataset with different samples of $(\mathbf{o}, \mathbf{c})$ and their corresponding expert demonstration trajectories $\boldsymbol{\tau}_e$.

Our CVAE model, illustrated in Fig. 2, consists of an encoder-decoder architecture augmented with PointNet [16] to extract features $\overline{\mathbf{o}}$ from unstructured and unordered LiDAR point clouds $\mathbf{c}$. The learnable parameters of PointNet are represented as $\boldsymbol{\omega}$. We stack the features from PointNet and state observations to create an augmented vector $\mathbf{o}_{aug} = (\mathbf{o}, \overline{\mathbf{o}})$, which is then fed to an encoder MLP along with the expert trajectory $\boldsymbol{\tau}_e$. The encoder compresses this information to a latent variable $\mathbf{z}$ with distribution $\mathbf{q}_\phi(\mathbf{z}|\mathbf{o}_{aug}, \boldsymbol{\tau}_e) = \mathcal{N}(\boldsymbol{\mu}_\phi, diag(\boldsymbol{\sigma^2}_\phi))$

The decoder model consists of an MLP with parameters $\boldsymbol{\theta}$ and a differentiable optimization layer. It takes in $\mathbf{o}_{aug}$ and maps $\mathbf{z} \sim \mathbf{q}_\phi(\mathbf{z}|\mathbf{o}_{aug}, \boldsymbol{\tau}_e)$ to the output distribution $\boldsymbol{\pi}_\theta(\boldsymbol{\xi}^*|\mathbf{z}, \mathbf{o}_{aug})$ over the optimal trajectory coefficients. The role of the optimization layer is to project the UAV/UGV coefficient distribution resulting from the MLP to the vehicles' current positions and velocities. Thus, the optimization layer is simply a differentiable equality-constrained QP. Both the encoder and decoder along with PointNet are trained in an end-to-end fashion using the loss function presented in (12), where $\overline{\mathbf{W}} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix}$ (recall (5)–(6)). The first term is called the reconstruction loss and aims to make the trajectory predicted by the CVAE as close as possible to the expert trajectory. The second term aims to make the latent distribution very similar to an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix.

$$\min \sum \left\|\overline{\mathbf{W}}\boldsymbol{\xi}^* - \boldsymbol{\tau}_e\right\|_2^2 + \beta\, D_{\mathbf{KL}}[\mathbf{q}_\phi(\mathbf{z}\,|\,\mathbf{o}_{aug},\, \boldsymbol{\tau}_e)\,|\,\mathcal{N}(\mathbf{0}, \mathbf{I})] \tag{12}$$

The back-propagation, required for updating the parameters $(\boldsymbol{\omega}, \boldsymbol{\phi}, \boldsymbol{\theta})$, traces the gradient with respect to the loss function through the QP layer. In the inference phase, we draw samples from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (instead of $\mathbf{q}_\phi(\mathbf{z}|\mathbf{o}_{aug}, \boldsymbol{\tau}_e)$) and then pass them through the decoder network to generate a distribution over optimal trajectory coefficients $\boldsymbol{\xi}^*$, conditioned on state observations $\mathbf{o}$ and point clouds $\mathbf{c}$. The β hyper-parameter acts as a trade-off between the two cost terms. We tackle the KL-vanishing issue by applying a monotonic annealing schedule of β hyper-parameter [17], starting with 0 and gradually annealing the β at optimizer step.

IV. Validation and Benchmarking

A. Implementation Details

Algorithms 1 and 2 are implemented in Python using JAX [18] library as our GPU-accelerated linear algebra back-end. The polynomial matrix $\mathbf{W}$ is constructed from a 10^{th} order polynomial. Our simulation pipeline is built on top of Robot Operating System (ROS) [19] and the physics simulator Gazebo. The benchmarks are run on AORUS Laptop with Intel core $i7-11800H$ and NVIDIA RTX 3080. For the real-world experiments, Parrot Bebop 2 and Robotont robot [20] are used. We use the learned occlusion model from [7] that takes in a robot and a target and obstacle positions as inputs and outputs an occlusion value. Thus, this model is evaluated over the UAV-UGV pair and between the UAV and the moving point of interest on the specified warehouse rack. The expert demonstration to learn CVAE was collected through teleoperation. The network architecture is presented in the accompanying video.

A Model Predictive Control (MPC) pipeline is developed on top of our optimizer. It takes the UGV and UAV's current positions and velocities as feedback along with LiDAR point clouds and generates occlusion-free trajectories for both vehicles in real-time. We do not use any warm-starting for our CEM optimizer in Algorithm 2.

1) Hyper-Parameter Selection: The batch size n of CEM in Algorithm 2 is 100 and we choose top 20 of them as the *EliteSet* in each iteration. We scale the smoothness and occlusion weights to obtain a trade-off between fluent motion and occlusion avoidance. It is worth mentioning that this scaling is kept the same throughout all the experiments.

2) Metrics: The following metrics are utilized for benchmarking:

UAV-UGV Occlusion Time: The total time the UAV and UGV's line of sight is occluded by the obstacles.

UAV-Rack Occlusion Time: The total time during which UAV cannot scan the racks due to the occlusions stemming from the obstacles.

Acceleration: This metric measures how fast the UAV and UGV change their speed during experiments.

Computation Time: The time taken to compute the optimal trajectories for the UAV and UGV.

Normalized Distance Traversed: The distance traveled before the occurrence of any collisions divided by the total distance of that particular experimental run.

B. Gaussian and CVAE Initialization

In this subsection, we present a qualitative demonstration of Algorithm 2 for both Gaussian and learned CVAE initializations. Fig. 3 represents the top-down view of a typical warehouse inventory scenario where the UAV needs to scan the racks while following the reference trajectory (black) and avoiding the obstacles (gray). Thus, both UAV and UGV need to avoid the obstacle from the bottom so that the UAV can keep seeing the rack and ensure that the UAV-UGV LoS is not occluded. As can be seen in Fig. 3, both initializations work but the learned CVAE initialization is more concentrated towards the feasible region from the very first iteration. As a result, the CVAE initialization leads to faster convergence.

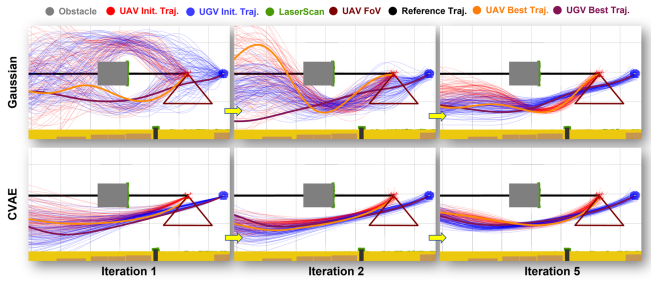

Fig. 3. Our CEM optimizer Algorithm 2 with naive Gaussian and learned CVAE initialization. The latter produces distribution more focused around regions that are collision-free and maintain UAV-UGV and UAV-rack visibility.

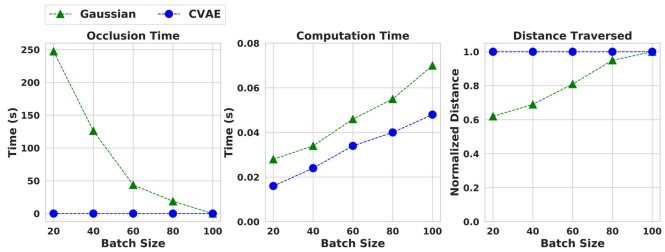

Fig. 4. Figure shows the comparison of different metrics of our optimizer when it is warm-started with Gaussian sampling and CVAE. Occlusion time refers to combined UAV-UGV and UAV-rack occlusion time.

C. Effect of CVAE

Here, we quantify the effect of initializing our CEM-based optimizer with a learned CVAE distribution vis-a-vis Gaussian. The results of different metrics are shown in Fig. 4. The CVAE initialization can match the performance of the Gaussian variant with $1/5^{th}$ of the batch (sample) size and fewer iterations. This, in turn, allows the CVAE variant to run on embedded devices such as NVIDIA Jetson Xavier with a computation time of 0.09 s. It is important to point out two additional experimental details. First, during our experiments with Jetson Xavier, the entire Gazebo simulation pipeline and our optimizer ran on the embedded board. Second, we allocated only 50% of the available GPU memory to our optimizer using JAX's internal parameters. Both these experimental settings give a reliable indication that Algorithm 2 for both UAV and UGV can run in real-time alongside another software stack for perception, state estimation, etc.

D. Comparison With SOTA Sampling Based Optimizers

In this subsection, we compare our Algorithm 2 with Gaussian and CVAE initializations with the following baselines.

- *CEM:* This is the most popular optimizer for planning with learned neural-network models such as our occlusion cost (e.g. see [14]). We recall that our Algorithm 2 improves

TABLE II
COMPUTATION TIME COMPARISON OF A* PLANNERS

Method	Horizon 2m	Horizon 4m	Horizon 6m	Horizon 8m	Horizon 10m
A*	0.002s	0.004s	0.006s	0.018s	0.028s
A* with Occ. Cost	0.24s	0.5s	0.8s	0.1.42s	1.86s
Alg. 2 (Ours)	0.028s	0.036s	0.048s	0.056s	0.07s

TABLE III
OVERALL DISTANCE TRAVELED / SIM. TIME: 1045.2 M / 1628 S

Ablations	UAV-UGV/ UAV-Rack Occ. time(s)	Normalized Distance Mean / STD	Linear Acc. Mean / Max	Comp. Time (s)
CEM Batch 100	15.4 / 290.5	0.62 / 0.15	0.17 / 0.49	0.15
CEM Batch 400	10.2 / 25.7	0.94 / 0.1	0.14 / 0.52	0.36
CMA-ES Batch 100	18.4/ 153.7	0.66 / 0.14	0.19 / 0.42	0.14
CMA-ES Batch 250	7.8 / 12.4	0.95 / 0.12	0.12 / 0.68	0.19
Alg. 2 - Gauss. Batch 100 (Ours)	**0 / 0**	**1 / 0**	**0.32 / 0.83**	**0.07**
Alg. 2 - CVAE Batch 20 (Ours)	**0 / 0**	**1 / 0**	**0.34 / 0.52**	**0.016**

CEM by augmenting a projection optimization and thus, this comparison essentially presents the benefit of the said augmentation.

- *CMA-ES:* This sampling-based optimizer differs from CEM in the way it adapts the covariance matrix of the sampling distribution. We use the customization of CMA-ES proposed in [6] and re-implemented it with the JAX GPU accelerated library [21].

The results are summarized in Table III. The UAV-rack and UAV-UGV occlusion times achieved for the baseline CEM with 100 batch size were 15.4 and 290.5 seconds, respectively. These figures decreased considerably to 10.2 and 25.7 seconds when the batch size increased to 400. CMA-ES, expectedly performed better than CEM on the occlusion time metric. In fact CMA-ES with a batch size of 250 outperformed CEM operating with a batch size of 400. Our projection-augmented CEM of Algorithm 2 with Gaussian initialization and the batch size of 100 outperformed both CEM and CMA-ES on the occlusion time metric. When warm-started with the learned CVAE initialization, Algorithm 2 could achieve the same performance with a batch size of only 20.

The normalized distance metric exhibits the same trends for all the baselines and our approach. Both CEM and CMA-ES require a large batch size to reach the mean normalized distance of around 0.95. In contrast, our Algorithm 2 with both Gaussian and CVAE initializations achieve the perfect value of 1 in the runs across all the benchmark environments.

Our optimizer, Algorithm 2, however, used more acceleration effort than both CEM and CMA-ES to adapt the motions quickly to unforeseen obstacles. Both CEM and CMA-ES required more iterations to obtain a feasible solution. This coupled with a larger batch size make their computation times two to five times that of our Algorithm 2.

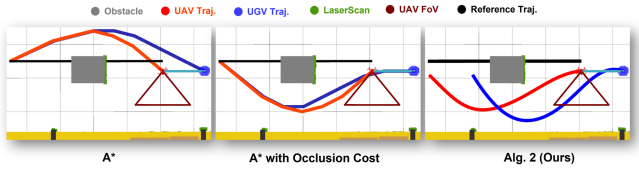

Fig. 5. Comparison of trajectories generated for UAV and UGV using A* and our optimizer presented in Algorithm 2.

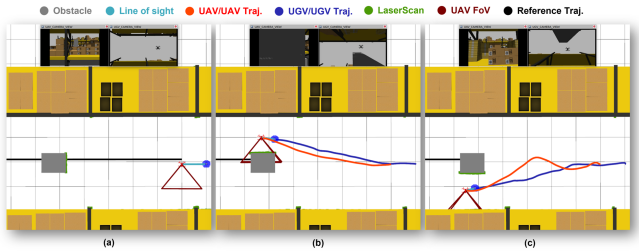

Fig. 6. (a)–(c) Show an experiment demonstrating the importance of having an occlusion cost between the UAV and the rack. Fig. (b) shows the result when the UAV's FoV is occluded due to absence of the occlusion cost. Fig. (c) shows how adding the occlusion cost makes the vehicles overtake the obstacle from the bottom to maintain the rack's visibility.

E. Comparison With A^* Planners

Here, we compare our approach with A^*-based planning. More specifically, we retain the sequential optimization framework of Algorithm 1 but replace the projection augmented CEM of Algorithm 2 with an A^*-based approach. We gave the end point of the reference trajectory as the goal point for both UAV and UGV. The qualitative results are summarized in Fig. 5. As shown, the A^* planner can find the correct occlusion-free homotopy for collision avoidance. However, the main difference from our optimizer lies in the computation time. The improvement provided by our approach is particularly stark while planning over longer horizons, which is indeed critical while navigating in unknown environments. Furthermore, the A^*-computed trajectories need additional smoothing. In contrast, our approach directly produces the smooth velocity commands that can be executed on the UAV and UGV.

The difference in the computation time can be explained in the following manner. A^* performs an incremental search over a discretized grid representation of the workspace and thus requires several sequential queries of occlusion and tracking costs. The latter is particularly computationally expensive. In contrast, our optimizer performs a more focused search along smooth trajectories. Moreover, cost computation along all the sampled trajectories is parallelized over GPUs.

F. Ablation Studies

1) Importance of Occlusion Cost With Respect to Racks: This section studies the importance of incorporating the UAV-rack occlusion cost $(f_{oc}(\mathbf{p}_r(t), \mathbf{p}_{sc}(t))$ presented in (1). As shown in Fig. 6(a)–(c), not including this occlusion cost, makes the UAV-UGV pair choose a homotopy that obstructs the view of the warehouse racks. We further quantify this behavior in Table IV. The simulation benchmarks spanned approximately 1890 seconds, covering a total distance of 1508.4 meters with 10 different obstacle configurations. As shown, having no rack occlusion cost results in 140.4 seconds where obstacles occlude

TABLE IV
DISTANCE TRAVELED / SIM. TIME : 1508.4 M / 1890 S

Ablations	UAV-UGV/ UAV-Rack Occ. time(s)	Normalized Distance Mean / STD	Linear Acc. Mean / Max	Comp. Time (s)
With rack occlusion	0 / 0	1 / 0	0.32 / 0.83	0.07
Without rack occlusion	0 / 140.4	1 / 0	0.15 / 0.36	0.052

TABLE V
OEVRALL DISTANCE TRAVELED / SIM. TIME : 1508.4 M / 1890 S

Ablations	UAV-UGV/ UAV-Rack Occ. Time(s)	Normalized Distance Mean / STD	Linear Acc. Mean / Max	Comp. Time (s)
Occ. handled by UAV	6.2 / 120.6	0.73 / 0.31	0.27 / 0.64	0.034
Occ. handled by UGV	8.2 / 15.3	0.76 / 0.32	0.24 / 0.63	0.034
Occ. handled by UAV-UGV	0 / 0	1 / 0	0.32 / 0.83	0.07

the UAV's view of the racks. In contrast, when UAV-rack occlusion cost is introduced to the optimizer, the vehicles finish the scanning task perfectly. In both cases, the vehicles finish the runs with zero collisions.

2) Importance of Incorporating Occlusion Costs for Both UAV and UGV: In a cooperative navigation setting, authors in [5] recommend putting the complete responsibility of keeping the inter-vehicle LoS occlusion-free on either of the vehicles. Mathematically, it means that LoS occlusion cost $f_{oc}(\mathbf{p}_r(t), \mathbf{p}_g(t))$ appears for either UGV or UAV but not both. In this subsection, we test this hypothesis for our problem setting. We ran all the ablations across the same benchmark scenarios as in Section IV-F1. The quantitative results are presented in Table V. We present the qualitative results in the accompanying video. As shown, it is not possible for either UAV or UGV to single-handedly take care of occlusions and both vehicles need to cooperate (or include $f_{oc}(\mathbf{p}_r(t), \mathbf{p}_g(t))$ in their optimization) to maintain perfect occlusion and collision-free task execution.

V. CONCLUSIONS, LIMITATIONS, AND FUTURE WORK

We presented VACNA, a motion planning algorithm for a UAV to scan warehouse racks under the guidance of a UGV. The latter can carry more sensing and computing hardware and plan for both the UAV and itself, provided it can keep the UAV in its field of view. Thus, VACNA characterizes a trajectory optimization problem with a carefully constructed set of cost and constraint functions to achieve this. We performed extensive simulations in a high-fidelity simulator to show how each of these functions affects the cooperative navigation and fulfillment of the scanning task. We also developed a projection-augmented CEM for solving the optimization underpinning VACNA and showed its improvement over the vanilla CEM and CMA-ES. Finally, we proposed a CVAE model learned from expert demonstrations that can be used to warm-start the CEM and results in a massive improvement in sample complexity and computation time.

Our work can be easily extended to a general cooperative navigation task and we present some preliminary results in this regard in the accompanying video. We are also looking to extend VACNA to a single-UGV-multi-UAV setting.

One of the limitation of our approach is that the CVAE initialization can get trapped in local minima due to its low variance. One workaround is to append some high-variance Gaussian samples to the ones drawn from the CVAE.

APPENDIX

In this section, we provide a more in-depth analysis of the projection optimization in lines 7–8 of Algorithm 2. We extend the derivations in [7] to the cooperative navigation setting. In particular, the collision avoidance and the UAV-UGV separation constraints are new additions to the optimizer presented in [7].

We begin by reformulating the inequality constraints (2c)–(2g) by noting that in lines 7–8, we have access to the optimal UGV trajectory coefficients ${}^{l}\boldsymbol{\xi}_g$ from the previous planning step. In other words, the waypoints ${}^{l}\mathbf{p}_g = ({}^{l}x_g(t), {}^{l}y_g(t))$ for the UGV are known. For notational simplicity, in the following, we drop the superscript l.

We reformulate the quadratic collision constraints (2e) into the following polar form.

$$\mathbf{f}_{o,i} = 0, d_{o,i} \geq 1, \forall i \tag{13}$$

$$\mathbf{f}_{o,i} = \begin{Bmatrix} x_r(t) - x_i(t) - a d_{o,i}(t) \cos \alpha_{o,i}(t) \\ y_r(t) - y_i(t) - a d_{o,i}(t) \sin \alpha_{o,i}(t) \end{Bmatrix}, \tag{14}$$

where $\alpha_{o,i}(t)$ represents the angle that the vector connecting the UAV centre and the i^{th} obstacle makes with the X axis. The variable $d_{o,i}(t)$ denotes the normalized magnitude of this vector. It can be easily verified that satisfaction of (13) ensures the satisfaction of original constraints (2e) based on the Euclidean distance.

Following a similar approach, constraints (2c)–(2d) and (2f)–(2g) can be reformulated to the forms (15)–(18).

$$\mathbf{f}_v = 0, d_v(t) \leq v_r, \tag{15}$$

$$\mathbf{f}_a = 0, d_a(t) \leq a_r, \tag{16}$$

$$\mathbf{f}_{los} = 0, \underline{s}_{los} \leq d_{los}(t) \leq \overline{s}_{los}, \tag{17}$$

$$\mathbf{f}_{ref} = 0, \underline{s}_{ref} \leq d_{ref}(t) \leq \overline{s}_{ref}, \tag{18}$$

$$\mathbf{f}_v = \begin{Bmatrix} \dot{x}_r(t) - d_v(t) \cos \alpha_v(t) \\ \dot{y}_r(t) - d_v(t) \sin \alpha_v(t) \end{Bmatrix}, \tag{19}$$

$$\mathbf{f}_a = \begin{Bmatrix} \ddot{x}_r(t) - d_a(t) \cos \alpha_a(t) \\ \ddot{y}_r(t) - d_a(t) \sin \alpha_a(t) \end{Bmatrix}, \tag{20}$$

$$\mathbf{f}_{los} = \begin{Bmatrix} x_r(t) - x_g(t) - d_{los}(t) \cos \alpha_{los}(t) \\ y_r(t) - y_g(t) - d_{los}(t) \sin \alpha_{los}(t) \end{Bmatrix}, \tag{21}$$

$$\mathbf{f}_{ref} = \begin{Bmatrix} x_r(t) - x_{ref}(t) - d_{ref}(t) \cos \alpha_{ref}(t) \\ y_r(t) - y_{ref}(t) - d_{ref}(t) \sin \alpha_{ref}(t) \end{Bmatrix} \tag{22}$$

Remark 1: $\alpha_{o,i}$, $\alpha_{los}(t)$, $\alpha_{ref}(t)$, $\alpha_v(t)$, $\alpha_a(t)$, $d_{o,i}$, $d_{los}(t)$, $d_{ref}(t)$, $d_v(t)$, and $d_a(t)$ are extra variables that will be obtained by our batch projection optimizer along with $\overline{\boldsymbol{\xi}}_{r,j}$.

Reformulation of Projection: Using the formulation developed in the previous section and the parametrizations in (5)–(6), we can replace the projection optimization in lines 7–8 of Algorithm 2 with the following:

$$\min_{\overline{\boldsymbol{\xi}}_{r,j}} \frac{1}{2} \left\| \overline{\boldsymbol{\xi}}_{r,j} - \boldsymbol{\xi}_{r,j} \right\|_2^2 \tag{23a}$$

$$\mathbf{A}_{eq} \overline{\boldsymbol{\xi}}_{r,j} = \mathbf{b}_{eq,r} \tag{23b}$$

$$\mathbf{F} \overline{\boldsymbol{\xi}}_{r,j} = \mathbf{e}_j(\boldsymbol{\alpha}_j, \mathbf{d}_j), \qquad \underline{\mathbf{d}} \leq \mathbf{d}_j \leq \overline{\mathbf{d}} \tag{23c}$$

$$\mathbf{F} = \begin{bmatrix} \begin{bmatrix} \mathbf{F_o} \\ \dot{\mathbf{W}} \\ \ddot{\mathbf{W}} \\ \mathbf{W} \\ \mathbf{W} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} \mathbf{F_o} \\ \dot{\mathbf{W}} \\ \ddot{\mathbf{W}} \\ \mathbf{W} \\ \mathbf{W} \end{bmatrix} \end{bmatrix}, \mathbf{e} = \begin{bmatrix} \mathbf{x_o} + \mathbf{ad_{o,j}} \cos \boldsymbol{\alpha}_{\mathbf{o,j}} \\ \mathbf{d}_{v,j} \cos \boldsymbol{\alpha}_{v,j} \\ \mathbf{d}_{a,j} \cos \boldsymbol{\alpha}_{a,j} \\ \mathbf{x}_g + \mathbf{d}_{los,j} \cos \boldsymbol{\alpha}_{los,j} \\ \mathbf{x}_{ref} + \mathbf{d}_{ref,j} \cos \boldsymbol{\alpha}_{ref,j} \\ \mathbf{y}_o + a\mathbf{d}_{o,j} \sin \boldsymbol{\alpha}_{o,j} \\ \mathbf{d}_{v,j} \sin \boldsymbol{\alpha}_{v,j} \\ \mathbf{d}_{a,j} \sin \boldsymbol{\alpha}_{a,j} \\ \mathbf{y}_g + \mathbf{d}_{los,j} \sin \boldsymbol{\alpha}_{los,j} \\ \mathbf{y}_{ref} + \mathbf{d}_{ref,j} \sin \boldsymbol{\alpha}_{ref,j} \end{bmatrix} \tag{24}$$

$$\boldsymbol{\alpha}_j = (\boldsymbol{\alpha}_{o,j}, \boldsymbol{\alpha}_{los,j}, \boldsymbol{\alpha}_{ref,j}, \boldsymbol{\alpha}_{v,j}, \boldsymbol{\alpha}_{a,j}) \tag{25}$$

$$\mathbf{d}_j = (\mathbf{d}_{o,j}, \mathbf{d}_{los,j}, \mathbf{d}_{ref,j}, \mathbf{d}_{v,j}, \mathbf{d}_{a,j}) \tag{26}$$

The matrix $\mathbf{F}_o$ is obtained by stacking the matrix $\mathbf{W}$ from (5) as many times as the number of obstacles in the point cloud for collision avoidance at a given planning cycle. The vectors $\mathbf{x}_o, \mathbf{y}_o$ are formed by appropriately stacking $x_{o,i}(t), y_{o,i}(t)$ at different time instants. Similar construction is followed to obtain $\boldsymbol{\alpha}_o, \boldsymbol{\alpha}_{los}, \boldsymbol{\alpha}_{ref}, \boldsymbol{\alpha}_v, \boldsymbol{\alpha}_a, \mathbf{d}_o, \mathbf{d}_{los}, \mathbf{d}_{ref}, \boldsymbol{d}_v, \boldsymbol{d}_a$. The vectors $\underline{\mathbf{d}}, \overline{\mathbf{d}}$ are constructed by stacking the lower and upper bounds of individual d's from (13)–(22).

The subscript j signifies that (23a)–(23c) are defined for projection of the j^{th} sample of $\boldsymbol{\xi}_{r,j}$. In total, there would be n such optimization problems. Second, the constraints (23c) serve as the replacement for $\mathbf{h}_1(\overline{\boldsymbol{\xi}}_{r,j}) \leq 0$ and $\mathbf{h}_2(\overline{\boldsymbol{\xi}}_{r,j}, {}^l\boldsymbol{\xi}_g) \leq 0$ in line 8 of the Algorithm 2

1) Solution Process: We relax the non-convex equality constraints (23c) as l_2 penalties and augment them into the projection cost (23a).

$$\begin{aligned} \mathcal{L}\left(\overline{\boldsymbol{\xi}}_{r,j}\right) &= \frac{1}{2}\left\|\overline{\boldsymbol{\xi}}_{r,j} - \boldsymbol{\xi}_{r,j}\right\|_2^2 - \left\langle \boldsymbol{\lambda}_j, \overline{\boldsymbol{\xi}}_{r,j} \right\rangle \\ &\quad + \frac{\rho}{2}\left\|\mathbf{F}\overline{\boldsymbol{\xi}}_{r,j} - \mathbf{e}(\boldsymbol{\alpha}_j, \mathbf{d}_j)\right\|_2^2 \end{aligned} \tag{27}$$

Note the introduction of the so-called Lagrange multipliers $\boldsymbol{\lambda}_j$. These play a crucial role in driving the residuals to zero for any arbitrary ρ [22].

We apply Alternating Minimization to minimize (27) subject to (23b). It reduces to the following recursive steps, wherein the left superscript k is used to track the values across iterations.

$${}^{k+1}\overline{\boldsymbol{\xi}}_{r,j} = \arg\min_{\overline{\boldsymbol{\xi}}_{r,j}} \mathcal{L}({}^{k}\overline{\boldsymbol{\xi}}_{r,j}, {}^{k}\boldsymbol{\alpha}_j, {}^{k}\mathbf{d}_j), \text{s.t } \mathbf{A}_{eq}\overline{\boldsymbol{\xi}}_{r,j} = \mathbf{b}_{eq,r} \tag{28a}$$

$${}^{k+1}\boldsymbol{\alpha}_j = \arg\min_{\boldsymbol{\alpha}_j} \mathcal{L}\left({}^{k+1}\overline{\boldsymbol{\xi}}_{r,j}, \boldsymbol{\alpha}_j, {}^{k}\mathbf{d}_j\right) \tag{28b}$$

$${}^{k+1}\mathbf{d}_j = \arg\min_{\mathbf{d}_j} \mathcal{L}\left({}^{k+1}\overline{\boldsymbol{\xi}}_{r,j}, {}^{k+1}\boldsymbol{\alpha}_j, \mathbf{d}_j\right) \tag{28c}$$

Essentially, we optimize only one variable amongst $\overline{\boldsymbol{\xi}}_{r,j}$, $\boldsymbol{\alpha}_j$, and $\mathbf{d}_j$, while the other two are held fixed at the previous update. Minimization (28a) is a convex QP with the same structure as (10), with $\overline{\mathbf{Q}} = \mathbf{I} + \rho\mathbf{F}^T\mathbf{F}$, $\overline{\mathbf{q}}_{r,j} = -\rho\mathbf{F}^T\,{}^{k}\mathbf{e}_j - {}^{k}\boldsymbol{\lambda}_j$. Thus, the solution across the entire batch can be calculated in parallel using (11). The steps (28b) and (28c) have a closed-form solution that can be trivially batched [7].

The projection operator can be trivially extended to the UGV case. It will be used in line 4 of Algorithm 1 for optimizing UGV trajectory with CEM of Algorithm 2.

REFERENCES

[1] W. Kwon, J. H. Park, M. Lee, J. Her, S.-H. Kim, and J.-W. Seo, "Robust autonomous navigation of unmanned aerial vehicles (UAVs) for warehouses' inventory application," *IEEE Robot. Automat. Lett.*, vol. 5, no. 1, pp. 243–249, Jan. 2020.

[2] S. Barlow, Y. Choi, S. Briceno, and D. N. Mavris, "A multi-UAS trajectory optimization methodology for complex enclosed environments," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2019, pp. 596–604.

[3] X. Li, J. Tan, A. Liu, P. Vijayakumar, N. Kumar, and M. Alazab, "A novel UAV-enabled data collection scheme for intelligent transportation system through UAV speed control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2100–2110, Apr. 2021.

[4] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of Statistics*, vol. 31. Amsterdam, The Netherlands: Elsevier, 2013, pp. 35–59.

[5] Y. Lee, J. Park, B. Jeon, and H. J. Kim, "Target-visible polynomial trajectory generation within an MAV team," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1982–1989.

[6] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, "VP-STO: Via-point-based stochastic trajectory optimization for reactive robot behavior," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 10125–10131.

[7] H. Masnavi, J. Shrestha, M. Mishra, P. B. Sujit, K. Kruusamäe, and A. K. Singh, "Visibility-aware navigation with batch projection augmented cross-entropy method over a learned occlusion cost," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9366–9373, Oct. 2022.

[8] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, "Fast autonomous flight in warehouses for inventory applications," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3121–3128, Oct. 2018.

[9] E. Welburn, H. Hakim Khalili, A. Gupta, S. Watson, and J. Carrasco, "A navigational system for quadcopter remote inspection of offshore substations," in *Proc. 15th Int. Conf. Autonomic Auton. Syst.*, 2019, pp. 2–6.

[10] X. Xiao, J. Dufek, and R. Murphy, "Benchmarking tether-based UAV motion primitives," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot.*, 2019, pp. 51–55.

[11] X. Xiao, J. Dufek, and R. R. Murphy, "Autonomous visual assistance for robot operations using a tethered UAV," in *Field and Service Robotics*, G. Ishigami and K. Yoshida, Eds. Singapore:Springer, 2021, pp. 15–29.

[12] V. Pritzl, M. Vrba, P. Štěpán, and M. Saska, "Cooperative navigation and guidance of a micro-scale aerial vehicle by an accompanying UAV using 3D LiDAR relative localization," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2022, pp. 526–535.

[13] H. Bharadhwaj, K. Xie, and F. Shkurti, "Model-predictive control via cross-entropy and gradient-based optimization," in *Proc. Learn. Dyn. Control*, 2020, pp. 277–286.

[14] C. Pinneri et al., "Sample-efficient cross-entropy method for real-time planning," in *Proc. Conf. Robot Learn.*, 2021, pp. 1049–1065.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6114v10

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.

[17] I. Higgins et al., "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl

[18] J. Bradbury et al., "JAX: Composable transformations of Python+ NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[19] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3, Jan. 2009.

[20] R. Raudmäe et al., "ROBOTONT–Open-source and ROS-supported omnidirectional mobile robot for education and research," *HardwareX*, vol. 14, 2023, Art. no. e00436.

[21] R. T. Lange, "evosax: JAX-based evolution strategies," in *Proc. Companion Conf. Genet. Evol. Comput.*, 2022, pp. 659–662.

[22] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable ADMM approach," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2722–2731.