CoCap: Coordinated motion Capture for multi-actor scenes in outdoor environments

Aditya Rauniyar[1], Micah Corah[2], and Sebastian Scherer[1]

***Abstract*—Motion capture has become increasingly important, not only in computer animation but also in emerging fields like the metaverse and humanoid training. Capturing outdoor environments offers extended horizons for tracking but introduces challenges with occlusions and obstacles. Recent approaches using multi-drone systems to capture multi-actor scenes often fail to account for multi-view consistency and reasoning across cameras in cluttered environments. CoCap, inspired by Conflict-Based Search (CBS), addresses this issue by coordinating view planning to ensure multi-view reasoning during conflicts. In comparison to Sequential Planning and unconstrained methods, CoCap delivers similar performance while introducing a real-time, coverage-based heuristic, making it well-suited for dense environments.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) equipped with cameras have garnered considerable attention for their wide range of applications, including surveillance, search and rescue, environmental monitoring, mapping, inspection, and 3D reconstruction [1–6].

Motivation: Recently, UAV teams have demonstrated effective view gathering for 4D pose reconstruction of single actors in outdoor environments [6]. This is particularly significant as traditional motion capture, typically confined to indoor studios, faces space limitations and incurs high costs for setting up the stage for stunts or other video shoots. Further research has extended this approach to multi-actor scenarios, where sequential view planning by a team of cameras ensures diverse and comprehensive coverage of a group of actors [7]. Moreover, development of a robust multi-camera aerial system still requires study.

Challenges: Such applications of motion capture in the wild present various challenges and requirements. First, since multi-view capture has been shown to improve 4D pose reconstruction for multiple actors, this demands a system of multiple camera-equipped drones capable of capturing diverse views of moving actors. Second, natural environments are filled with obstacles and occlusions, especially when cameras are maneuvered to obtain optimal coverage. This requires the drone system to not only manage obstacles and occlusions but also optimize for maximum coverage. Third, in these operational scenarios, as more camera-equipped drones are deployed, the system must also coordinate the views to avoid redundancy and prevent deadlocks in real-time.

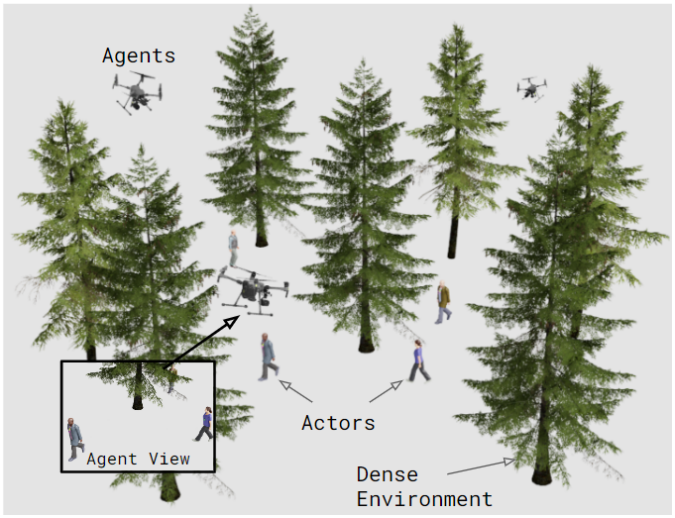


Fig. 1: **Coordinated View Planning**: Coverage optimization on dynamic actors with flying cameras in an occlusion-aware and obstacle-clustered environment where camera extrinsic positions across robots are negotiated.

CoCap: To address these challenges, we develop an efficient method for incorporating constraints into the system as more drones are deployed. Drawing inspiration from Conflict-Based Search (CBS) [8], we introduce constraints to the agents (drones) as conflicts arise. We define conflicts as instances where two agents enter a collision state, and we resolve this by applying constraints to one of the agents while maintaining a constraint tree. Our proposed approach of Coordinated motion Capture (CoCap) is tested in simulation, utilizing the reward structure employed by GreedyPerspectives [7]. Additionally, for real-time applications, we propose a single-agent view search method that prioritizes actor coverage and uses a heuristic to explore near-optimal viewpoints. More details on the simulation setup and reward structure can be found in our earlier research with GreedyPrespectives [7].

Contributions: Main contributions are as follows:

- A multi-camera view planning system that encourages multi-view coverage and discourages ego-centric positions of cameras.
- Scenarios of multi-actor obstacle clustered environments that are particularly challenging for multi-camera

[1] A. Rauniyar, and S. Scherer are with the Robotics Institute, School of Computer Science at Carnegie Mellon University, Pittsburgh, PA, USA ..{rauniyar, basti}@cmu.edu

[2]M. Corah is with the Department of Computer Science at the Colorado School of Mines, Golden, CO, USA micah.corah@mines.edu

This work is supported by the National Science Foundation under Grant No. 2024173 and supported by Defense Science and Technology Agency Singapore contract DST000EC124000205.

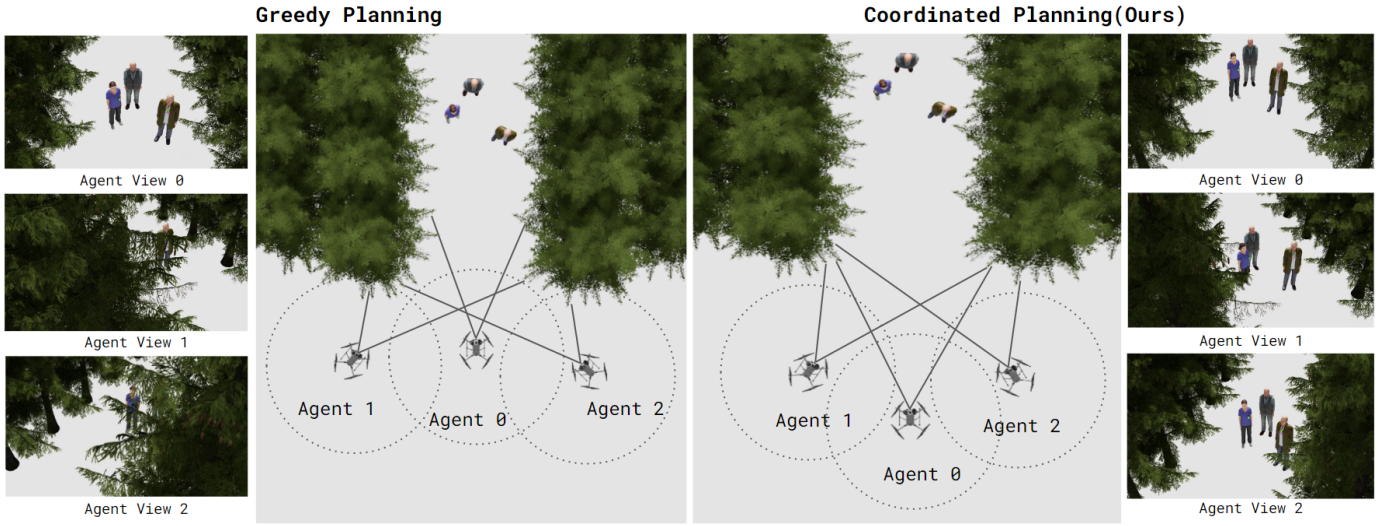


Fig. 2: **Sequential (Greedy) View Planning**: On the left, there is the sequential view planning of multiple camera positions, where there are egocentric behaviors across multiple viewpoints as seen in the three camera outputs on the left under greedy planning. **Coordinated planning**, on right: we propose a coordinated view planning approach where there is pixel-level negotiation amongst view positions to allow non-egocentric behaviors as seen in the three camera outputs on the right under coordinated planning.

equipped drones.

- Heuristic-based single agent view planning framework for real-time applications.

II. Related Work

Motion capture of actors in natural settings: Recently, there has been growing interest in developing motion capture systems using aerial cameras, particularly for outdoor environments. AirCapRL [9] introduces a deep reinforcement learning approach that learns an optimal policy for camera formations, focusing on achieving ideal viewing angles. However, this method struggles with reasoning about obstacles and occlusions in the environment and does not address scenarios involving multiple actors. Ho et al. [6] tackled the first limitation by developing a formation planner constrained to optimal views around an actor using a spherical grid, though it still does not account for multiple actors. One approach that addresses the multiple actor scenario is by Hughes et al. [10], who maximize Pixels-Per-Area (PPA) over multiple actors represented as polygonal cylinders. This method has been further refined to account for obstacles and occlusions with the GreedyPerspectives algorithm [7]. However, while these approaches consider multiple actors, they lack multi-view consistency and are not designed for real-time operations. CoCap addresses these challenges by introducing system constraints as conflicts arise and employing a heuristic-driven view planning method, guided by rewards from maximizing coverage across multiple actors.

Collision-Free Navigation for Multi-Robot Teams: Designing dense multi-robot systems presents significant challenges, particularly in conflict resolution. One widely used approach is prioritized planning, where robots are assigned a fixed priority sequence and planned sequentially based on their priority ID. This method has been employed in multi-actor view planning, such as in GreedyPerspectives [7]. However, prioritized planning often leads to suboptimal results, as the preset order can cause later robots to encounter an increasing number of constraints, potentially resulting in deadlocks or planning failures. A more efficient approach to constraint development is Conflict-Based Search (CBS) [8], where constraints are dynamically introduced as conflicts arise during the expansion of a constraint tree. In multi-robot systems, conflicts typically occur when robots attempt to occupy the same position at the same timestep. Despite the success of CBS in navigation, its application in perception planning systems remains underexplored. CoCap extends the CBS approach to motion capture in multi-actor scenarios, specifically in outdoor environments with complex obstacles and occlusions.

III. Problem Formulation

Consider a set of robots $\mathcal{R} = \{1, \ldots, N_r\}$, and a set of actors $\mathcal{A} = \{1, \ldots, N_a\}$ each with a set of faces $\mathcal{F}_j = \{1, \ldots, N_{j,f}\}$ where $j \in \mathcal{A}$. Also, let all the sets of faces across all the actors be $\mathcal{F} = \{\mathcal{F}_1, ..., \mathcal{F}_{N_a}\}$. Each of the faces, $\mathcal{F}_j : j \in N_a$, are associated with a set of pixel coverage by agents, $\mathcal{P}\mathrm{x} = \{\mathcal{P}\mathrm{x}_{111}, ..., \mathcal{P}\mathrm{x}_{ijk}\}$, where $i \in N_r$, $j \in N_a$, and $k \in N_{j,f}$. All the robots move in a workspace represented as a finite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and share a global clock that start at $t = 0$. Vertex set $\mathcal{V}$ represents the set of all the possible locations of i^{th} robot at time t as $v_t^i \in \mathcal{V}$ and the edge set $\mathcal{E}$ as actions $e_t^i \in \mathcal{E}$ of i^{th} robot at time $t \in \{0, \ldots, T\}$ where $i \in \mathcal{R}$. The reward for each edge is a M-dimensional nonnegative vector reward$(e_t^i) \in \mathbb{R}^{+M} \setminus \{0\}$.

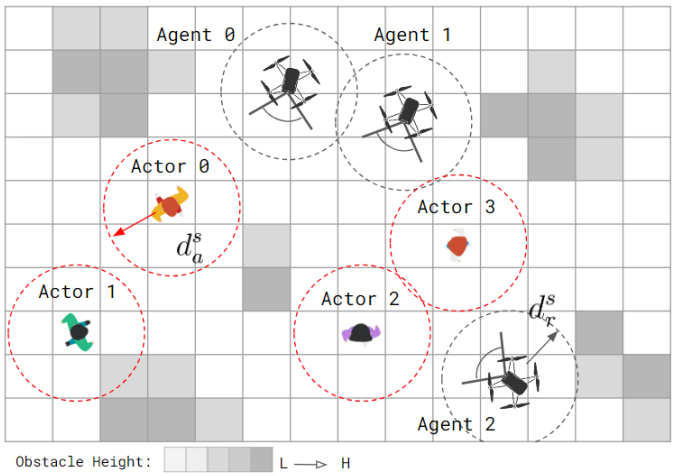


Fig. 3: Problem representation of the gimbaled camera (also formulated as a robot, $\mathcal{R}$) with projection matrix facilitating coverage on dynamic targets with occlusion and obstacles.

Let $\xi^i(v_1^i, v_l^i)$ be a path that connects the vertices v_1^i and v_l^i via a sequence of vertices $(v_1^i, v_2^i, ..., v_l^i) \in \mathcal{G}$. Let $g^i(\xi^i(v_1^i, v_l^i))$ denote the M-dimensional reward vector associated with the path $\xi^i(v_1^i, v_l^i)$ as the sum of all the reward vectors of all the edges present in the path, *i.e.*, $g^i(\xi^i(v_1^i, v_l^i)) = \sum_{j=1,...,l-1}(v_j^i, v_{j+1}^i)$.

Let $v_o^i, v_f^i \in \mathcal{V}$ represent the initial location and destination of robot i respectively. For simplicity, we denote a path from v_o^i to v_f^i for robot i as ξ^i. A joint path for all robots, denoted as $\xi = (\xi^1, \xi^2, ..., \xi^{N_r})$, is referred to as a solution. The cost vector of this solution is defined as the sum of individual path costs over all agents, i.e., $g(\xi) = \sum_{i=1}^{N_r} g^i(\xi^i)$.

Objective: We employ submodular optimization techniques to synchronize the actions of a multi-robot team and to study the associated objective. The observation of each actor j is quantified in terms of the pixel density ($\frac{px}{m^2}$) obtained from a linear camera model's image. We define two functions, $\texttt{cov}(x_{i,t}, j, f) \rightarrow \mathbb{R}$, returning the pixel density for a specific actor's face observed from a robot's position, and $\texttt{covp}(j, f) \rightarrow \mathbb{R}$, returning the cumulative pixel density from all past observations. This enables us to express the incremental coverage gain:

$$\texttt{covm}(x_{i,t}) = \sum_{j \in \mathcal{T}} \sum_{f \in \mathcal{F}_j} \sqrt{\texttt{cov}(x_{i,t}, j, f) + \texttt{covp}(j, f)} - \sqrt{\texttt{covp}(j, f)}$$

IV. Brief Overview on Conflict-Based Search

Conflict-Based Search [8] is a two level search that creates a binary binary tree to resolve conflicts in the agent paths on a high level and runs an optimal low level search algorithm for a Multi-Agent Path Finding (MAPF) problem.

Conflict Resolution: Consider a pair of agents, i and j, each following their respective paths ξ^i and ξ^j. To detect conflicts between these paths, we utilize the function $\Psi(\xi^i, \xi^j)$. This function returns either an empty set if no conflict is present, or it provides details about the first conflict encountered along the paths. A conflict occurs at time t when agent i is at position v_t^i and agent j is at position v_t^j. This conflict is denoted by (i, j, v_t^i, v_t^j, t). To prevent such conflicts, a corresponding constraint is added to the path of either agent i or agent j. This constraint is represented as $\omega^i = (i, u_a^i, u_b^i, t)$, where u_a^i and u_b^i are selected from the set of possible locations V. This constraint is associated with agent i and ensures that at time t, agent i follows the specified path, thus avoiding the potential conflict.

Given a set of constraints Ω, let $\Omega_i \subseteq \Omega$ represent the subset of all constraints in Ω that belong to agent i (i.e., $\Omega = \bigcup_{i \in N_r} \Omega_i$). A path ξ_i is consistent with respect to Ω if ξ_i satisfies every constraint in Ω_i. A joint path ξ is consistent with respect to Ω if every individual path $\xi_i \in \xi$ is consistent.

Two level Search: In the high-level while creating the constraint tree with each node containing $(\xi, \Omega, g(\xi))$, each of them are in a priority queue. Initially, parent node $\mathcal{P}\texttt{x}_0 \in \texttt{TREE}$ is computed for all the agents such that the constraint set, $\Omega_o = \phi$, and $P_o = (\xi_0, \Omega_o, g_o)$ gets pushed to $\texttt{TREE}$. Let, total number of nodes created be denoted as N_G, and total number of nodes expanded as $N_E : N_E \leq N_G$.

As each node, k, in the $\texttt{TREE}$ gets explored based on the least g value, $P_k = (\xi_k, \Omega_k, g_k)$, $\Psi(\xi_k^i, \xi_k^j)$ gets called for all the sequence pairs $(i, j) \in N_r : (i \neq j)$. If there is no conflict detected, the solution, ξ, is found and the algorithm terminates.

Algorithm 1: Coordinated View Planner

Data: $X_{\texttt{init}}$, $\mathcal{T}$ trajectories , envHeightMap, agentMaxMotion
Result: $U^\dagger$

Initialization()
$\texttt{TREE} \leftarrow GreedyPlanningWithoutConstraints()$
Initialize $U_{\text{seq}} \leftarrow \{\}$
while *TREE not empty* **do**
  $P_k \leftarrow (\xi_k, \Omega_k, \mathcal{P}\texttt{x}_k, g_k)$ // $\texttt{TREE}$.pop()
  if *no breach detected in* ξ_k **then**
    return P_k
  end
  $\Omega \leftarrow$ Split detected breach
  forall $\omega_i \in \Omega$ **do**
    $\Omega_{li} \leftarrow \Omega_k \cup \{\omega_i\}$
    $\mathcal{P}\texttt{x}_l \leftarrow \mathcal{P}\texttt{x} \setminus \{\mathcal{P}\texttt{x}_{itk}\}$
    $\xi^{i*} \leftarrow$ LowLevelViewSearch$(i, \Omega_l, \mathcal{P}\texttt{x}_l)$
    $\mathcal{P}\texttt{x}_l^* \leftarrow \mathcal{P}\texttt{x}_l \cup GetCoverage(\xi^{i*})$
    $\xi_l^* \leftarrow \xi_l \setminus \{\xi^i\} \cup \{\xi^{i*}\}$
    $g_l \leftarrow GetObjectiveValue(\xi_l^*, \mathcal{P}\texttt{x}_l^*)$
    $P_l \leftarrow \{\xi_l^*, \Omega_l, \mathcal{P}\texttt{x}_l^*, g_l\}$
    $\texttt{TREE} \leftarrow P_l$
  end
end
return U_{seq}

If there is a conflict detected, (i, j, v_t^i, v_t^j, t), constraints to the agents are created in the following way. Each condition is created where constraint is added to either i or j, such that $\omega^i = (i, u_a^i = v_t^i, u_b^i = v_t^j, t)$ and $\omega^j = (j, u_a^j = v_t^j, u_b^j = v_t^i, t)$. Each of these constraints leads to a new node in the TREE, where $P_{l^i} = (\xi_{l^i}, \Omega_{l^i}, g_{l^i})$, and $P_{l^j} = (\xi_{l^j}, \Omega_{l^j}, g_{l^j})$, such that $l^i \in N_G$ where new set of constraints is added to i or j. Here, $\Omega_{l^i} = \Omega_k \cup \omega^i$, and $\Omega_{l^j} = \Omega_k \cup \omega^j$. In each of these cases, for agent i, the algorithm updates the path ξ_k^i in ξ_{l^i} using the low-level search with a set of constraints Ω_{l^i}. A similar call is made for agent j leading to ξ_k^j. If the low-level search is unable to find a solution for any of these cases, respective P_{l^n} is discarded.

CBS solves a single objective cost function, g, optimally by iterative expansion of a node in the constraint tree based on the least g val, resolving first agent-agent conflict (if exists) and creating a new node in the tree with new Ω, or returning the solution $P = (\xi, \Omega, g)$.

V. Coordinated View Planning

Our proposed planner Alg. 1 also works in a similar structure to CBS. With the following key differences:

- **Greedy planning**: Rather than initializing with solutions to single-agent path-finding problems we initialize with an approximate solution to the joint multi-agent view planning problem that relaxes constraints between robots.
- **Tree Node($\mathbf{P}_k$)**: The node contains additional information on the pixels coverage by the agents over the faces of corresponding actors. This is given by $P = (\xi, \Omega, \mathcal{P}\mathrm{x}, g)$.
- **Cost(g)**: The cost, g, consists of a multi-objective cost function that is given by eq. **??**, i.e.,
- **Termination criteria**: Taking computational expense into consideration, we return the first set of the solution P without any conflicts. If there is no valid solution, a failure is reported.
- **Replanning**: As a new conflict between any two agents (i, j) is detected using Ψ, the low level plans for agent i, with constraint set Ω_{l^i} and $\mathcal{P}\mathrm{x} \setminus \{\mathcal{P}\mathrm{x}_{ijk}\} : j \in N_a$, and $k \in N_{j,f}$. Similarly, for agent "j".
- **Low-level search**: Instead of having A^* as a low-level search, we want a multi-objective low-level solver. Here, we use a value iteration solver at the low level, whose reward structure considers coverage over actors, smoothness, stationarity, actor placement in the image frame, and proximity to the nearest actor. All of which is an important criteria for a single-agent view planner.

A. Constraint Tree Formation

Initialization: During this state, with $(\Omega_o = \phi, \mathcal{P}\mathrm{x}_o = \phi)$, as we plan for a series of agents, $i \in$, we plan each agent greedily towards maximizing its reward using the single-agent view planner to produce the set of trajectories ξ_o. The root node, $P_o = \{\xi_o, \Omega_o, \mathcal{P}\mathrm{x}_o, g_o\}$ gets added to TREE. As each agent, j, gets planned, the pixel context as passed such that $\mathcal{P}\mathrm{x}_j = \{\mathcal{P}\mathrm{x}_{jkl}\} : j < i, k \in \mathcal{T}, l \in N_{k,f}$.

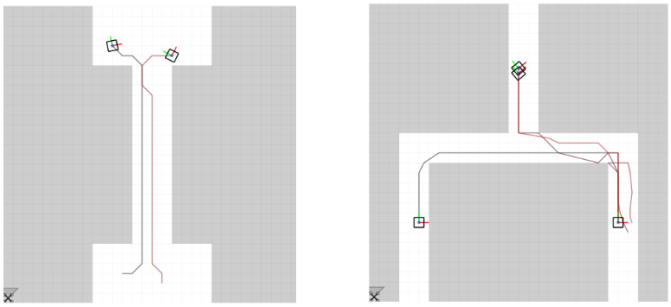
Finding a solution: As node is popped from the TREE, $P_k = \{\xi_k, \Omega_k, \mathcal{P}\mathrm{x}_k, g_k\}$. Ψ checks for all the breaches amongst all the sequence pair of agents $(i, j) : i, j \in N_r$, and $i \neq j$. If there are no breaches found, such that $||(v_t^i - v_t^j)|| > s^{ia}$ where $v_t^i \in S_t^{ia}$ and $v_t^j \in \mathcal{V}_t^j : j \in \mathcal{R} \setminus \{i\}$, P is the solution, and the high-level search terminates. There are multiple conflict-free solutions to a set of valid trajectories maximizing g, however, we are interested in the first solution for computational performance and guide the tree expansion by the multi-objective function $g = J(\Xi)$.

Resolving conflicts: If there exists a breach between any two agent, $(i, j) : i \neq j$, then agent i is re-planned with constraint set $\Omega_{l^i} = \Omega_k \cup S_t^{ja}$ and $\mathcal{P}\mathrm{x} \setminus \{\mathcal{P}\mathrm{x}_{itk}\} : t \in \mathcal{T}$, and $k \in N_{t,f}$, and agent j is re-planned with constraint set $\Omega_{l^j} = \Omega_k \cup S_t^{ia}$ and $\mathcal{P}\mathrm{x} \setminus \{\mathcal{P}\mathrm{x}_{jtk}\} : t \in N_a$, and $k \in N_{t,f}$. The output trajectory of each agent "i" from the low-level solver is updated to $(\xi_{l^i}^i)$, and pixel coverage set to $\mathcal{P}\mathrm{x}_{l^i}$, such that $P_{l^i} = (\xi_{l^i}, \Omega_{l^i}, \mathcal{P}\mathrm{x}_{l^i}, g_{l^i})$, where g_{l^i} is calculated with the updated set of Ω_{l^i} and $\mathcal{P}\mathrm{x}_{l^i}$. Similarly, for agent "$j$", new constraints and pixel-set leads to $P_{l^j} = (\xi_{l^j}, \Omega_{l^j}, \mathcal{P}\mathrm{x}_{l^j}, g_{l^j})$. P_{l^i}, and P_{l^j} are added to the TREE with their corresponding g, and the tree is rearranged such that the root node has max(g= $J(\Xi)$).

Algorithm 2: Single-Agent View Search

Data: ξ_0^i, $\mathcal{T}$ trajectories , envHeightMap(H), agentMaxMotion(m^i), $\mathcal{P}\mathrm{x}_i$, Ω_i
Result: ξ^i

Initialize $\xi^i \leftarrow \{\}$
TREE $\leftarrow \{\mathbf{R}_0, \xi_0^i, e_0^i, t\}$
while *TREE.top().t is not T* **do**
  $Q_k \leftarrow (\overline{\mathbf{R}_k}, \xi_k^i, e_{k-1}^i, k)$ TREE.pop()
  updateProcessedStates[ξ_k^i] = true
  updateStatesFromPrevAction(Q_k)
  $e_{k_1}^i, ..., e_{k_{S^T}}^i \leftarrow$ $availableActions(\xi_k^i, k, \Omega_i, H, m^i)$
  forall $e_{k_1}^i, ..., e_{k_{S^T}}^i$ **do**
    $\overline{\mathbf{R}_{k+1}} \leftarrow (\mathbf{R}_{k+1} + \overline{\mathbf{R}_k})\gamma^{k-1}$
    // Cumulative reward x discount factor (< 1) for encouraging to reach higher reward states in lesser timesteps
    $Q_{k+1} \leftarrow \{(\overline{\mathbf{R}_{k+1}}, \xi_{k+1}^i, e_{k_n}^i, k+1\}$
    if $processedStates[\xi_{k+1}^i] == true || \overline{\mathbf{R}_{k+1}} <$ $TREE[\xi_{k+1}^i].\mathbf{R}$ **then**
      continue
    end
    TREE $\leftarrow Q_{k+1}$
  end
end
$\xi^i \leftarrow BackTracking(processedStates)$
return ξ^i



(a) Corridor (b) Bottleneck

Fig. 4: (a) Corridor and (b) Bottleneck

B. Single-Agent View Planner

Here we discuss the proposed single-agent view search technique that is developed using $A*$, however, using perceptual metric as a heuristics to form a max heap that explores the next node in the max heap that contains the maximum perceptual reward as described in algorithm 2. The algorithm starts by exploring the start position of the camera and adds all the nodes the `TREE` that has a valid action to it. However, the reward is added multiplied with a discount factor to value near rewards more than later in the timestep, which also encourages finding max reward in shorter timesteps. Using this new reward, we create a new P_{k+1} and check if it's in the `TREE`, if its not in the `TREE`, or its new reward is lesser than that from the one that is already present in the `TREE`, we skip adding the Pto the `TREE`, else we add. At the end the trajectory is computed through backtracking from the last state.

VI. Experimental Results

In this section, we present the experimental results of the multi-UAV camera system designed for dynamic actors. The simulations were conducted using a custom-built simulation environment to test the system under bottleneck and corrridor scenarios.

A. Ego-Centric Test

In this test, we evaluated scenarios where the agents exhibited non-egoic behavior, such as one robot getting out of the way to allow another robot to pass. The objective was to assess the system's ability to handle interactions and coordination among agents. We also evaluate the constraints developed in the system when comparing the no-inter-robot constraints, while adding inter-robot constraints, and while doing a conflict-based constraint assignment.

Figure 5 presents the scaled rewards of three view planning methods in a corridor environment with 8 timesteps, 2 robots, and 2 actors. The robots are randomly initialized outside the corridor while the actors move through it in the same direction. From timesteps 3 to 8, when the robots are inside the corridor, a clear gap emerges between the performance of the proposed sequential planning and planning without system

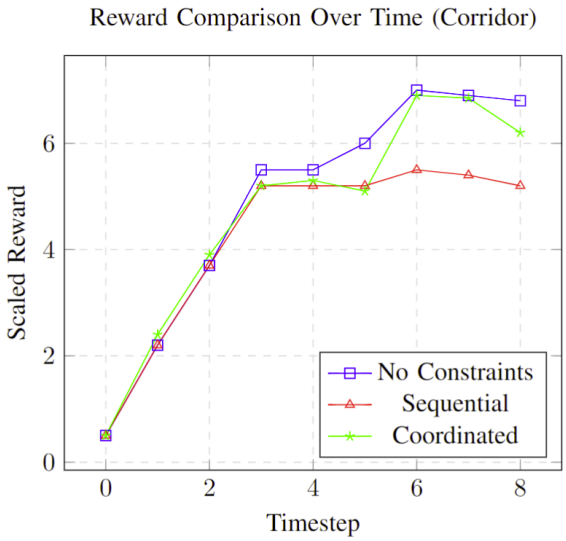


Fig. 5: Reward comparison amongst different methods in corridor environment.

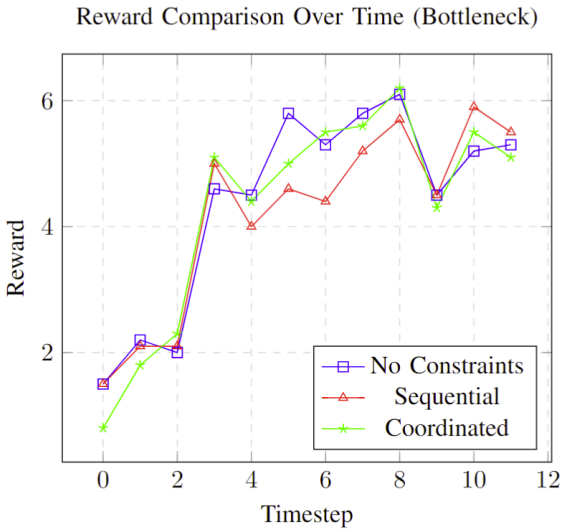


Fig. 6: Reward comparison amongst different methods in bottleneck environment.

Fig. 7: Scale rewards using multiple cameras performing view planning using No inter-robot Constraint, Sequential Constraint, and Conflict Based MDP Value Iteration, over total planning horizon for the environment.

constraints. The proposed Alg. 1 aims to narrow this gap, bringing performance closer to the unconstrained system.

Similarly, Figure 6 compares the three methods in a bottleneck environment over 11 timesteps with 4 robots and 4 actors, where actors from different corridors merge in the bottleneck area. From timesteps 3 to 9, the gap between the sequential planning view reward and planning without inter-robot constraints highlights the system's reward decrease as constraints are introduced. This indicates the need for an

Method	Total Reward (scaled)	Compute Time (s)
Sequential w/ Value Itr.	4127	482
CoCap w/ Value Itr.	**4662**	3981
CoCap w/ View Search	3922	**2.7**

TABLE I: Comparison of Total Reward and Compute Time across different methods for corridor.

intelligent approach to adding constraints without significantly reducing the view reward. The proposed coordinated capture effectively bridges this gap, achieving performance comparable to the unconstrained method but with added inter-robot constraints.

Together, these two scenarios in Figure 7, with high obstacles and occlusions, demonstrate the proposed methods' effectiveness in maintaining high-view rewards despite added constraints.

B. Single-Agent Value Iteration vs Search

For offline planning approaches, the value iteration solver can compute sub-optimal view positions to achieve high view rewards as shown in GreedyPerspectives [7]. However, in more online settings, there is a need for more efficient algorithms. The proposed single-agent view search, described in Alg. 2, aims to address these use cases.

The Table I compares the total reward (scaled) and compute time (in seconds) for different view planning methods in a corridor environment. The Sequential method with Value Iteration achieved a total reward of 4127 and a compute time of 482 seconds. The Coordinated method with Value Iteration yielded the highest total reward of 4662 but required significantly more compute time, at 3981 seconds. In contrast, the Coordinated method with View Search had a lower total reward of 3922 but was the fastest, with a compute time of only 2.7 seconds. This highlights a trade-off between achieving higher rewards and compute efficiency, showcasing the effectiveness of using single-agent view search in online settings.

VII. Conclusion and Future Work

In conclusion, this research tackles the challenge of coordinated motion capture of multiple actors in complex, obstructed environments using camera-equipped UAVs. We introduced a novel system for multi-robot coordinated view planning, inspired by Conflict-Based Search (CBS) with an occlusion-aware objective. Evaluations in two scenarios demonstrate that our approach, CoCap, outperforms sequential planning, especially in obstacle-dense environments. Coordinated view planning closely matches the performance of a system without collision constraints, while outperforming sequential greedy planning. Additionally, our single-agent view search method provides a significant computational advantage over the single-agent value iteration solver.

While promising, the system has limitations. It relies on a 2.5D height map, limiting its handling of overhangs and coverage at doors and windows. Actors are oversimplified as cuboids, and real multi-UAV deployment hasn't been tested, potentially facing communication issues with the centralized planner. More testing is needed in denser, occluded environments, and the search heuristic lacks exploration strategies, especially when high rewards emerge later. Ensuring communication reliability is also crucial for real-world deployment.

Acknowledgment

We would also like to thank Krishna Suresh, Yuechuan Hou, and Micah Nye for their assistance in developing parts of the work. Micah Corah had primarily been part of the research during his PostDoc at CMU.

References

[1] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, "Multiple Moving Targets Surveillance Based on a Cooperative Network for Multi-UAV," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 82–89, Apr. 2018, conference Name: IEEE Communications Magazine.

[2] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "LSAR: Multi-UAV Collaboration for Search and Rescue Missions," *IEEE Access*, vol. 7, pp. 55 817–55 832, 2019, conference Name: IEEE Access.

[3] S. McCammon, G. Marcon dos Santos, M. Frantz, T. P. Welch, G. Best, R. K. Shearman, J. D. Nash, J. A. Barth, J. A. Adams, and G. A. Hollinger, "Ocean front detection and tracking using a team of heterogeneous marine vehicles," *Journal of Field Robotics*, vol. 38, no. 6, pp. 854–881, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.22014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22014

[4] X. Tong, X. Liu, P. Chen, S. Liu, K. Luan, L. Li, S. Liu, X. Liu, H. Xie, Y. Jin, and Z. Hong, "Integration of UAV-Based Photogrammetry and Terrestrial Laser Scanning for the Three-Dimensional Mapping and Monitoring of Open-Pit Mine Areas," *Remote Sensing*, vol. 7, no. 6, pp. 6635–6662, May 2015. [Online]. Available: http://www.mdpi.com/2072-4292/7/6/6635

[5] Y. Kompis, L. Bartolomei, R. Mascaro, L. Teixeira, and M. Chli, "Informed Sampling Exploration Path Planner for 3D Reconstruction of Large Scenes," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7893–7900, Oct. 2021, conference Name: IEEE Robotics and Automation Letters.

[6] C. Ho, A. Jong, H. Freeman, R. Rao, R. Bonatti, and S. Scherer, "3D Human Reconstruction in the Wild with Collaborative Aerial Cameras," Aug. 2021, arXiv:2108.03936 [cs]. [Online]. Available: http://arxiv.org/abs/2108.03936

[7] K. Suresh, A. Rauniyar, M. Corah, and S. Scherer, "Greedy perspectives: Multi-drone view planning for collaborative perception in cluttered environments," Oct. 2024.

[8] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, Feb. 2015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0004370214001386

[9] R. Tallamraju, N. Saini, E. Bonetto, M. Pabst, Y. T. Liu, M. J. Black, and A. Ahmad, "AirCapRL: Autonomous Aerial Human Motion Capture using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6678–6685, Oct. 2020, arXiv:2007.06343 [cs]. [Online]. Available: http://arxiv.org/abs/2007.06343

[10] S. Hughes, R. Martin, M. Corah, and S. Scherer, "Multi-robot planning for filming groups of moving actors leveraging submodularity and pixel density," Milan, Italy, Dec. 2024, to appear.