Learning to select multi-viewpoints for Filming Dynamic Targets

Aditya Rauniyar (arauniya)
Yu Quan Chong (yuquanc)
Jay Karhade (jkarhade)

Abstract:

This paper addresses the crucial task of enhancing the reconstruction quality of dynamic scenes by leveraging UAVs equipped with high-definition cameras. Despite their proven efficacy, deploying such systems for human-operated filming necessitates careful consideration of collision avoidance with the environment and other drones, ensuring human safety. We propose an adaptive viewpoint selection approach to collect distributed pixel-level metrics like Pixel-Per-Area (PPA) and account for the diversity of seen pixels. The methodology involves converting the scene representation into a low-level geometric mesh, enabling reasoning for each pixel. We formulate a Markov Decision Process (MDP) model using Q-learning and SARSA, offering scalability and safer camera pose estimates for UAV navigation. We conduct deeper studies to explore the performance of effective robot learning algorithms against value-iteration policies, and provide explanation of the intuitive advantages and limitations of these methods in terms of generalizability and effectiveness in human-operated environments.

1 Introduction

The increased availability of cameras on UAVs has invoked deep interest in its use for various applications such as search and rescue, environmental monitoring, 3D reconstruction [1, 2, 3, 4]. In such applications, it is critical for any UAV to maximize its coverage over the environment, and ensure optimal coverage of dynamic targets. We formulate the problem as a multi-agent multi-target coverage problem, where the objective is to calculate a collision-free path for the agents that maximize pixel coverage of dynamically moving targets.

While one can use value iteration, it requires known MDP of the environment. This is a non-ideal candidate for large-scale multi-UAV planning and dynamic target tracking given that the policy extraction requires a known transition function, which is oftten unkown in real-world scenarios. This prompts us to investigate the use of sampling based Q-value iteration functions, namely Q-learning and SARSA. By using a sampling-based method, we hope to scale to larger and unkown scenes effectively.

2 Related Work

Our work builds upon recent work in multi-view drone planning and dynamic target tracking.

[3] proposed a system to capture multiple views of a single actor for human pose reconstruction, by using a preconfigured actor-centric formation.[5] proposed a spherical discretization of the robot state space, allowing for rapid greedy single-robot planning, which could generate joint-plans in a sequential manner. To extend this to a multi-actor space, [**?**] proposed allowing robots to plan over the full-environment, but did not consider inter-robot collisions. [6] proposes to learn control policies for actively classifying moving targets, building upon [7],[8].

A key challenge is the intractable nature of multi-robot perception and planning problems, which are often solved through greedy sub-modular optimization, guaranteeing a lower bound of half of the optimal solution in polynmial time. Subsequently, [9] removed these limitations by implementing a planner for optimize camera viewpoints for multi-drone multi-actor scenarios, building upon prior work from [10]and [**?**]. In particular, [9] develops upon perception objectives proposed by [10] on using pixel densities (PPA) as a reconstruction quality proxy and plan views by maximizing the PPA of the actor's geometrical shape.

3 Methodology

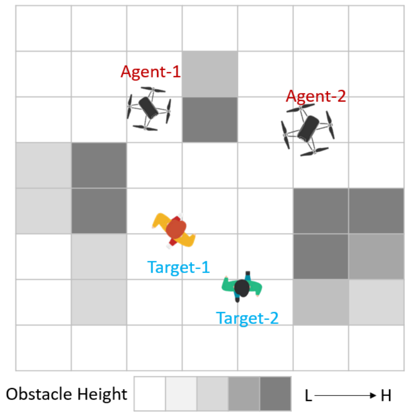


Figure 1: Problem Formulation: We aim to calculate joint paths for a group of agents to maximize pixel coverage of the dnyamic targets

Our aim is to jointly plan collision-free control input sequences for a team of agents for maximizing the reconstruction coverage of a dynamically moving agents in a particular environment.

Particularly, we have r agents, denoted by $\{1, \dots, N_r\}$, and a set of dynamic targets $\mathcal{T} = \{1, \dots, N_t\}$. Each of these targets have faces $\mathcal{F}_j = \{1, \dots, N_{j,f}\}$ where $j \in \mathcal{T}$ and we represent the set of all faces for all targets as $\mathcal{F} = \{\mathcal{F}_1, ..., \mathcal{F}_{N_t}\}$. Furthermore, we associate these faces with a set of pixel coverage by agents $= \{_{111}, ..., _{ijk}\}$, where $i \in N_r$, $j \in N_t$, and $k \in N_{j,f}$.

The movement of these agents is synchronized with a common global clock that starts at $t = 0$, and we represent this movement in a workspace as a finite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V}$ consists of all possible locations for the i-th agent at time t ($v_t^i \in \mathcal{V}$), while $\mathcal{E}$ represents the set of actions ($e_t^i \in \mathcal{E}$) taken by the i-th agent at discrete times $t \in 0, \dots, T$, where i belongs to the set of robots, . Each action e_t^i has an associated reward, denoted as $reward(e_t^i)$, which is a non-zero vector in R^{+M} with M dimensions. $\xi^i(v_1^i, v_l^i)$ defines a path linking vertices v_1^i and v_l^i through a sequence of vertices $(v_1^i, v_2^i, ..., v_l^i) \in \mathcal{G}$. Furthermore, $g^i(\xi^i(v_1^i, v_l^i))$ is the M-dimensional reward vector associated with the path $\xi^i(v_1^i, v_l^i)$, which is calculated by summation of the reward vectors from all edges present in the path, and is expressed as $g^i(\xi^i(v_1^i, v_l^i)) = \sum_{j=1,\dots,l-1}(v_j^i, v_{j+1}^i)$.

We define $v_o^i, v_f^i \in \mathcal{V}$ as the initial location of the i-th agent and its corresponding destination. Then, a path from v_o^i to v_f^i for agent i is represented by as ξ^i, and the joint path (solution) for all agents is denoted by $\xi = (\xi^1, \xi^2, ..., \xi^{N_r})$. The solution's cost vectors is then calculated as the sum of all agents' individual path costs $g(\xi) = \sum_{i=1}^{N_r} g^i(\xi^i)$.

For each agent, the state transitions is specified as a constant velocity motion model:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t})$$

Our objective is then to maximise the pixel coverage of the dynamic targets, which is calculated by the sum of pixel coverage of all agents' faces $J_{cov}(\xi) = \sum_i^{N_t}(\sum_i J_{cov}(F_{ji}))$.

To solve this objective function we have the reward function defined by the number of pixels covered by a render, using this reward function we create an MDP model for each robots and solve using the algorithms 1 and 2, that is of SARSA and Q-learning.

Q-Learning is employed to learn Q-values for state-action pairs, enabling agents to autonomously plan optimal paths. The algorithm operates in an environment defined by states (S), actions (A), a transition model (P), and a reward function (R). Hyperparameters, including the learning rate (α), discount factor (γ), and exploration rate (ϵ), are set to control the learning process.

The SARSA algorithm complements our approach by learning Q-values for state-action pairs in a sample-based variant of Q-value iteration. Similar to Q-Learning, SARSA operates in an environment defined by states, actions, a transition model, and a reward function. Hyperparameters are set to ensure effective learning.

```
Algorithm 2: Q-Learning
Data: Environment with states $S$, actions $A$, transition model $P$, reward function $R$
Result: Learned Q-values for each state-action pair, $Q(s, a)$
Initialize $Q(s, a)$ arbitrarily for all $s \in S$ and $a \in A$;
Set hyperparameters: learning rate $\alpha$, discount factor $\gamma$, exploration rate $\epsilon$;
for each episode do
    Initialize the state $s$;
    while not terminal state is reached do
        Choose action $a$ using an exploration-exploitation strategy, e.g., $\epsilon$-greedy;
        Take action $a$, observe reward $r$ and next state $s'$;
        Update Q-value: $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$;
        Update the current state: $s \leftarrow s'$;
    end
end
```

```
Algorithm 4: SARSA
Data: Environment with states $S$, actions $A$, transition model $P$, reward function $R$
Result: Learned Q-values for each state-action pair, $Q(s, a)$
Initialize $Q(s, a)$ arbitrarily for all $s \in S$ and $a \in A$;
Set hyperparameters: learning rate $\alpha$, discount factor $\gamma$, exploration rate $\epsilon$;
for each episode do
    Initialize the state $s$;
    Choose action $a$ using an exploration-exploitation strategy, e.g., $\epsilon$-greedy;
    while not terminal state is reached do
        Take action $a$, observe reward $r$ and next state $s'$;
        Choose next action $a'$ using an exploration-exploitation strategy, e.g., $\epsilon$-greedy;
        Update Q-value: $Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot Q(s', a'))$;
        Update the current state and action: $s \leftarrow s'$, $a \leftarrow a'$;
    end
end
```

Figure 2: Psuedocode of Q-Learning and SARSA algorithms, as taken from [11]

4 Experiments

In our experimental setup, we employed a system equipped with a NVIDIA GeForce RTX 3090 Ti GPU to facilitate the training and evaluation of our reinforcement learning models. For the Q-learning approach, we utilized standard parameters, including a learning rate of 0.001, a discount factor (gamma) of 0.99, and an exploration-exploitation trade-off represented by the epsilon-greedy strategy with an initial epsilon value of 1.0 and a decay rate of 0.995 per episode. The neural network architecture employed for Q-learning consisted of two fully connected layers with ReLU activation functions. Additionally, we implemented experience replay with a buffer size of 10,000 samples to enhance learning stability.

Similarly, for the SARSA solver, we utilized comparable parameters to ensure a fair comparison. The SARSA algorithm incorporated a learning rate of 0.001, a discount factor of 0.99, and an epsilon-greedy strategy with an initial epsilon value of 1.0, decaying at a rate of 0.995 per episode. The neural network architecture for SARSA followed the same configuration as the Q-learning model. Both Q-learning and SARSA were subjected to a maximum of 1,000 episodes for training, with each episode representing interactions within the environment. These standardized parameters aimed to establish a consistent and comparable experimental framework for evaluating the proposed adaptive viewpoint selection system.

5 Results

The experimental evaluation compared Value Iteration, SARSA, and Q-Learning in a corridor environment. Figure 3 showcased qualitative visualizations of the pixel coverage by Value Iteration(modeled by a defined transition function, SARSA and Q-Learning(both of which demonstrated Q function learning without defined transitions). In Figure 4, we provide quantitative results of these 3 methods. These results highlight a further need for hyperparameter tuning in SARSA and Q-Learning to improve against the state-of-art value Iteration's advantage in smaller environments, as well as testing in large-scale scenes to demonstrate scalability.

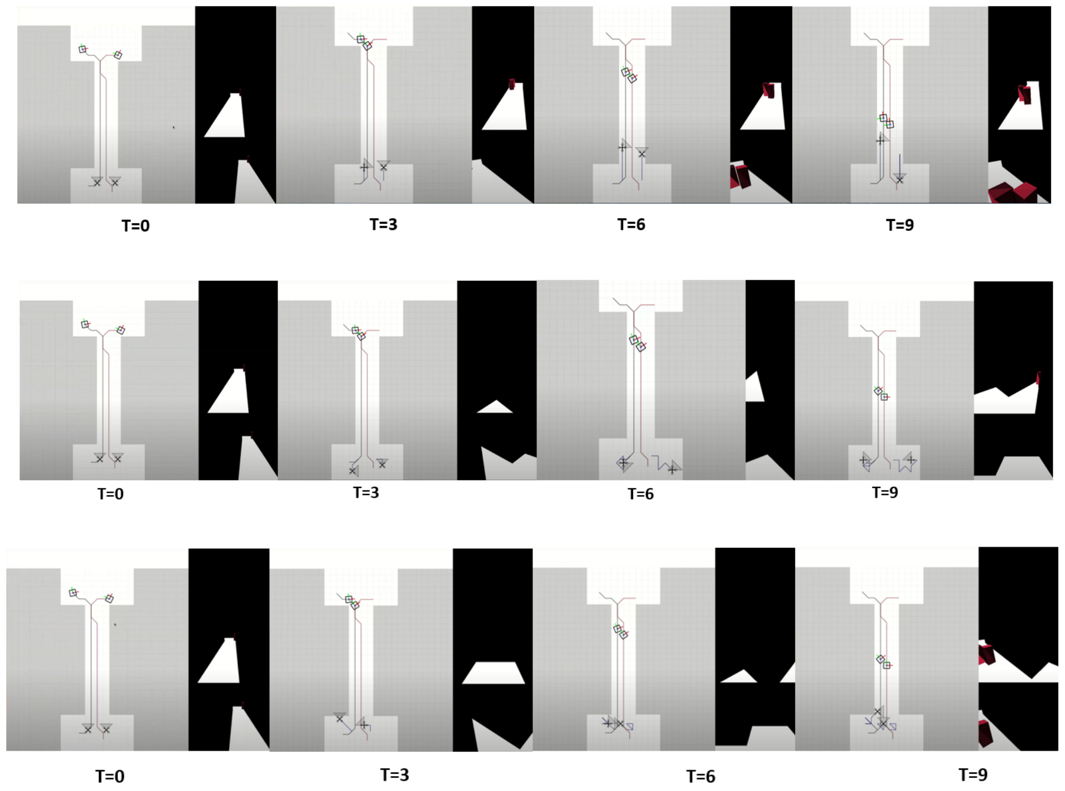


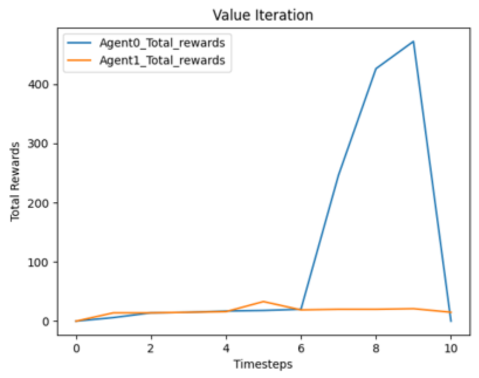
Figure 3: Performance Comparison in a Corridor Environment: The top row depicts results from Value Iteration, the middle row illustrates outcomes obtained using the SARSA algorithm, and the bottom row exhibits results achieved through Q-Learning. In the corridor environment, the left robot represents Robot 1, and the right robot is Robot 2. The side black panel shows renders from Robot 2 at the top and Robot 1 at the bottom. Square boxes represent two targets from a top view, and at T=0, their corresponding trajectories are visible with black and red lines. The results highlight that with a well-defined transition function, the robots exhibit distributed pixel coverage. In contrast, results from SARSA and Q-Learning, without explicitly defined transition functions, showcase the direct learning of the Q function.

6 Conclusion and Future Work

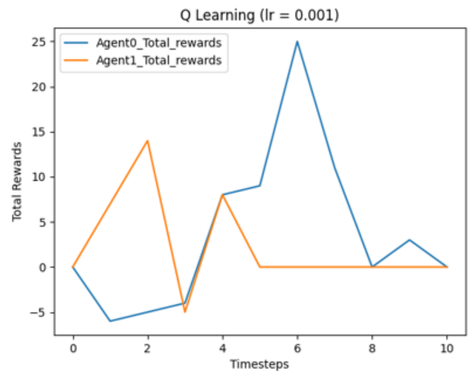
In conclusion, this paper introduced an adaptive viewpoint selection approach for enhancing the reconstruction quality of dynamic scenes using UAVs equipped with high-definition cameras. The proposed methodology, employing Q-learning and SARSA within a Markov Decision Process framework, demonstrated its effectiveness in achieving distributed pixel coverage. Comparative analyses with Value Iteration highlighted the adaptability of the reinforcement learning solvers. While exhibiting promising results, further refinement through hyperparameter tuning is acknowledged, particularly for Q-Learning and SARSA. Future work will involve extensive testing on diverse and larger scenes, affirming the method's potential for human-operated filming in dynamic and challenging environments.

Acknowledgments

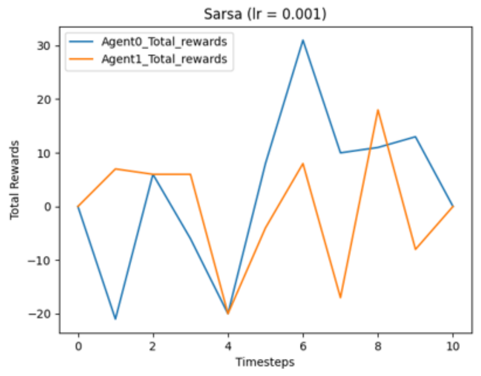
We extend our heartfelt thanks to Deepak Pathak, Lili Chen, Tanmay Shankar, Eliot Xing, and M. Nomaan Qureshi for their invaluable contributions and guidance.



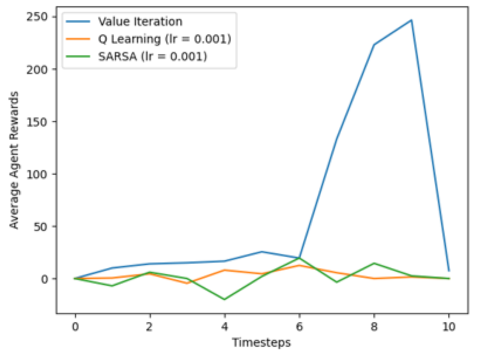
(a) Agent Returns – Value Iteration



(b) Agent Returns - Q Learning



(c) Agent Returns - SARSA



(d) Average Agent Returns

Figure 4: Performance Comparison of Reinforcement Learning Solvers: The top-left figure displays maximum rewards achieved with the Value Iteration solver. However, Q-Learning and SARSA algorithms exhibit the need for hyperparameter tuning, particularly in terms of decreasing learning rates and increasing episodes, as seen in the top-right and bottom-left figures. Notably, the performance benchmarks in larger grids for SARSA and Q-Learning are expected to be intuitively better, as Value Iteration would require a transition function that becomes impractical in real-world scenarios with larger environments.

References

[1] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa. Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832, 2019. doi:10.1109/ACCESS.2019.2912306.

[2] K. Shimizu, T. Nishizono, F. Kitahara, K. Fukumoto, and H. Saito. Integrating terrestrial laser scanning and unmanned aerial vehicle photogrammetry to estimate individual tree attributes in managed coniferous forests in japan. *International Journal of Applied Earth Observation and Geoinformation*, 106:102658, 2022. ISSN 1569-8432. doi:https://doi.org/10.1016/j.jag.2021.102658. URL `https://www.sciencedirect.com/science/article/pii/S0303243421003652`.

[3] C. Ho, A. Jong, H. Freeman, R. Rao, R. Bonatti, and S. Scherer. 3d human reconstruction in the wild with collaborative aerial cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5263–5269. IEEE, 2021.

[4] 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*, 6(4):7893–7900, 2021. doi:10.1109/LRA.2021.3101856.

[5] A. Bucker, R. Bonatti, and S. Scherer. Do you see what i see? coordinating multiple aerial cameras for robot cinematography. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7972–7979. IEEE, 2021.

[6] Á. Serra-Gómez, E. Montijano, W. Böhmer, and J. Alonso-Mora. Active classification of moving targets with learned control policies. *IEEE Robotics and Automation Letters*, 2023.

[7] B. F. Jeon, D. Shim, and H. J. Kim. Detection-aware trajectory generation for a drone cinematographer. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1450–1457. IEEE, 2020.

[8] A. Alcántara, J. Capitán, R. Cunha, and A. Ollero. Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles. *Robotics and Autonomous Systems*, 140: 103778, 2021.

[9] K. Suresh, A. Rauniyar, M. Corah, and S. Scherer. Greedy perspectives: Multi-drone view planning for collaborative coverage in cluttered environments. *arXiv preprint arXiv:2310.10863*, 2023. URL https://arxiv.org/pdf/2310.10863.pdf. Under Review at IEEE (ICRA) 2024.

[10] Q. Jiang and V. Isler. Onboard view planning of a flying camera for high fidelity 3d reconstruction of a moving actor. *arXiv preprint arXiv:2308.00134*, 2023.

[11] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.