ASEN 5264 Final Project
Drone Surveillance POMDP Solving

Collin Hudson
Ann & H.J. Smead Department of Aerospace Engineering Sciences
Universty of Colorado
Boulder, Colorado 80303
Email: Collin.Hudson@Colorado.edu

Luke Roberson
Ann & H.J. Smead Department of Aerospace Engineering Sciences
Universty of Colorado
Boulder, Colorado 80303
Email: Luke.Roberson@Colorado.edu

Amanda Marlow
Ann & H.J. Smead Department of Aerospace Engineering Sciences
Universty of Colorado
Boulder, Colorado 80303
Email: Amanda.Marlow@Colorado.edu

I. Introduction

Surveillance has many applications such as search and rescue, security, and defense. We have defined a Drone surveillance problem as a POMDP, where the rewards and hazards are initially unknown by the agent and must be discovered by exploring and observing the environment.

A. Problem Definition

A Drone must determine the position of a target and photograph it while avoiding detection by a detector object, and return to base by exiting the environment. The positions of the target and detector are unknown, and must be determined via imperfect observations. This uncertainty in the state with uncertain measurements make it ideally suited to be formulated as a POMDP. The environment consists of an $n \times n$ grid world with a controllable Drone and three stationary entities with random positions. The Drone is able to move one square at a time in the cardinal directions within the grid, and the problem immediately terminates once the Drone steps outside the defined grid world.

1) Entities: There are three types of entities: a Target, a Detector, and a Benign entity. We refer to these entity types as the identity of an entity. The Drone receives a significant positive reward for finding/photographing the Target. The Drone will receive negative rewards for getting within a certain distance of the Detector. The Benign entity provides no rewards, but has some probability of being observed as the Target or Detector.

2) Drone Observations: After any move action, the Drone observes its current cell and 8-neighbor cells, i.e. cells sharing either an edge or a vertex with the current cell of the Drone. The Drone observes both the identity of an entity and its position relative to the Drone for all entities within the one-cell observation range, otherwise known as its field of view (FOV).

II. Background and Related Work

As a starting point for this model, the team began with the Drone Surveillance problem defined in [1]. This problem began with a Drone and an adversary that the Drone had to avoid in the environment while attempting to reach a reward grid cell. The observations, rewards, and state were heavily modified to achieve our POMDP model as described in the following sections.

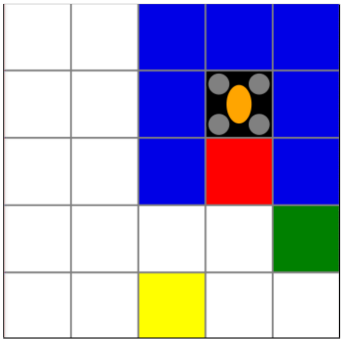

Fig. 1: Basic Grid World Formulation

We also looked at existing literature to get a sense of how POMDPs are already being used for Drone path planning and surveillance. Bravo et. al provides a good example of how POMDPs can be used to improve the speed at which surveillance Drones used in disaster response locate victims in need of assistance [2]. Although our POMDP formulation is focused on surveillance in an adversarial environment instead of humanitarian relief, this paper provides evidence that theoretical reward based POMDPs like ours can be translated into real world UAV operations to improve outcomes over existing path planning algorithms.

As with many POMDPs, we quickly noticed our problem experienced the curse of dimensionality. When we attempted to use Bayesian belief updaters with solvers like SARSOP, no solution could be reached without significantly reducing the uncertainty in the problem by removing uncertainty in entity positions. This lead us to use a POMCP solver which was proposed in [3]. Their paper discusses how the use of an unweighted particle filter belief update and a black-box simulator allows the POMCP solver to scale to much larger state spaces. These benefits make POMCP an ideal solver for us to use in this project.

The primary belief updater we used in this project is a bootstrap filter, which is a version of a weighted particle filter originally proposed in [4]. Replacing POMCPs unweighted

particle filter with a bootstrap filter avoids particle depletion issues.

The article "A framework for multiple ground target finding and inspection using a multirotor UAS" details a very similar problem to the one we set out to solve [5]. Like our project, they are using a quadrotor to identify the location and identity of multiple targets. This paper is useful to connect our problem to implementation on real hardware. They also talk about the use of their framework in a wide variety of problems like emergency response and agriculture. Our POMDP methods could also translate to these uses as well. This article is also interesting because it briefly cites a paper that applies a POMDP to a similar problem in agriculture, but dismisses it because it required known target/obstacle positions. They use an Observe, Orient, Decide and Act decision framework instead with separate software modules for each step. We were able to use a POMDP formulation to identify targets with unknown positions, so this project could possibly be used in the future to better inform decision making frameworks like this one.

III. Problem Formulation

1) State Space: The state consists of the position of the Drone, the static positions of all entities, each entities' actual identity, and a boolean for if a photo was taken.

$$\mathcal{S} = \{\vec{x}, [\vec{x_1}, \vec{x_2}, \vec{x_3}], [\mathrm{id}_1, \mathrm{id}_2, \mathrm{id}_3], \mathrm{photo}\} \tag{1}$$

2) Action Space: The action space is simply the movement of the Drone in one of the cardinal directions. After each action, the Drone takes an observation. If the Drone moves onto the Target entity the Drone takes a photograph automatically.

$$\mathcal{A} = \{\text{Up, Down, Left, Right}\} \tag{2}$$

3) Transition Probabilities: All actions are deterministic – the drone will always move to the cell dictated by the action. In order to handle rewards and the terminal state, we defined two off-grid states that are reached deterministically when the Drone chooses an action that leads to a position outside the grid bounds. If the Drone has taken a photograph of the Target and moves out of bounds, the Drone transitions to the reward state from which it can receive a reward, then transitions to the terminal state. If the Drone hasn't taken a photograph and still moves out of bounds, the Drone immediately transitions to the terminal state with no reward.

4) Reward Function: A positive reward is given for the Drone exiting the grid with a photo of the Target, and a negative reward is given for the drone occupying the same cell as the Detector.

$$\mathcal{R}(s,a) = \begin{cases} -1 \text{ if } \vec{x} = \vec{x_i} \text{ and } \mathrm{id}_i = \text{Detector} \\ +2 \text{ if } \vec{x} = \text{off grid and photo} = \text{true} \end{cases} \tag{3}$$

5) Observation Space: A single observation encodes the probability of an observed entity being the Target, Detector, or Benign identity, and the relative position of that entity to the drone. The Observation Space was constructed from the Cartesian product of the observation spaces of the simultaneous and independent observations of each entities' identity and relative position.

$$\begin{aligned} &O \in \text{Identities}^{n_{\text{entities}}} \times \text{Directions}^{n_{\text{entities}}} \\ &\text{Identities} = \{\text{T, D, B, N}\} \\ &\text{Directions} = \{\text{N, NE, E, SE, S, SW, W, N}\} \end{aligned} \tag{4}$$

where ID is an observation taken from the Identities space, x_{ID} is an observation taken from the Directions space, and n_{entities} is the number of entities in the environment. For the first two levels of success, only three entities were considered to be in the environment. In the last level of success we explored adding more entities to the environment, but for much of the report 3 will be used for clarity.

Often, n_{entities} is assumed to be 3 and 3 will be used instead of using n_{entities} in many parts of this report.

- Identity Observation Space
  The identity observation space is defined as all the permutations with repetition of the identity set {Target, Detector, Benign, None} for each of the three entities. In other words, the identity observation space has a dimension equal to the number of entities in the environment where each dimension is comprised of the four possible observations. This results in the position observation space having a size of $P(4,3)$ where $P(n,k)$ denotes all permutations of length k from n elements. In this case, the None identity corresponds to an entity not being observed within the range of the Drone. Thus, the set of permutations describes all possible identity observation outcomes for every entity.
- Position Observation Space
  The identity observation space is defined as all the permutations with repetition of the position set {North, Northeast, East, Southeast, South, Southwest, West, Northwest, None} for each of the three entities. This will have size $P(10,3)$. In this case, the None observation corresponds to an entity not being observed within the range of the Drone. Thus, the set of permutations describes all possible relative position observation outcomes for every entity.

6) Observation Probabilities: Because the observation probabilities for location and identity are independent, the joint probability distribution was calculated by multiplying all possible combinations of possible identity and position measurements together. Because these distributions can be vectorized, the joint distribution can be calculated as the outer product of the identity and position observation distributions. The observation distributions for identities and locations are shown below.

$$\mathcal{Z}_{\text{ID}}(o|ID) = \begin{cases} 0.7 \text{ if } o = ID \text{ and in FOV} \\ 0.1 \text{ otherwise if in FOV} \\ 1.0 \text{ if } o = N \text{ and out of FOV} \\ 0.0 \text{ otherwise if out of FOV} \end{cases} \tag{5}$$

$$\mathcal{Z}_{x_{\text{ID}}}(o|x_{ID}) = \begin{cases} 1.0 \text{ if } o = x_{\text{ID}} \text{ and in FOV} \\ 0 \text{ otherwise} \end{cases} \quad (6)$$

In words, Equation 5 says that if an entity is within the Drone's field of view, it is measured correctly 70% of the time and is uniformly random over the other three option in the Identities set. If the entity is outside of the FOV, it is measured as None deterministically

For Equation 6, the position of every entity within the Drone FOV is always measured with perfect accuracy. Essentially, this means that the drone does not know where entities are until they are within its FOV, upon which it should know their exact location.

IV. Solution Approach

A. Modifying off-the-shelf POMDP implementation

Following Professor Sunberg's recommendation, we based our POMDP implementation in `DroneSurveillance.jl` [6], which implemented the POMDP defined in [1] with the Julia POMDPs package. The drone in this problem must survey two regions/targets while avoiding flying over a ground agent. The drone has a limited field of view, and the ground agent can move. This existing repository provided the general structure for our code, and provided a helpful way to test solvers before finishing our own POMDP. We made the following modifications to the POMDP structure to use it for our problem.

1) Observation Space: In order to create the permutation sets for the position observation, a dictionary of directions was defined with the cardinal and intercardinal directions mapped to the difference in position of the adjacent cell in that direction and the drone, e.g. (0,+1) $\rightarrow$ North and (-1,-1) $\rightarrow$ Southwest. Then for each entity, the range and direction of an entity relative to the Drone during an observation was determined using the difference in position of the Drone and an entity. The distribution of position observations was assigned as a vector whose elements corresponded to the probability of receiving the matching direction observation.

In a similar fashion, the permutation sets for the identity observation was created by checking the true identity of an in-range entity, and returning an observation probability vector. For both the position and identity observations, any entity that was out of range of the Drone was deterministically assigned an "out-of-range" observation corresponding to None. The observation vectors for all the entities were then combined in an outer product, one for each observation type, then the outer products were combined using the Cartesian Product and returned as a sparse probability distribution for all possible observations.

2) State Encoding: Every possible state must be assigned a unique index to be referred to within `POMDPs.jl`. When each entity's position was uncertain, the size of the state space increased by a factor of $(5 \times 5)^4 = 390,625$. Additionally, the identities of the entities were randomized and uncertain but were restricted such that each identity only appeared in the environment once. I.e entities 1, 2 and 3 could be a Target, Benign, and Detector respectively, but not Detector, Detector, and Detector. This results in $3! = 6$ permutations. With the addition of the boolean state representing if a photo was "taken", and two extra states for the terminal state and reward state, the total size of the state comes to $4,687,502$. We also performed runs with 6×6 and 7×7 grids which resulted in $20,155,393$ and $69,177,613$ states respectively. To go to and from state representations, Julia's `CartesianIndices` and `LinearIndeces` were used.

B. Solvers

We used a variety of built in solvers and belief updaters from `POMDPs.jl`. When only considering uncertain entity identities with certain positions, we started by using an off the shelf version of SARSOP with value iteration. We expected this to be a good solver to use because it's an offline method that can find near optimal policies. However, SARSOP is slow and limited on large state spaces. Even for the unmodified Drone Surveillance problem, the algorithm was very slow and produced very large model files. To combat this, we tried using the QMDP algorithm, which was able to handle some larger grid sizes and more entities. We used the default Bayesian updater with `QMDPsolver.jl`.

When we moved on to modifying the POMDP with position uncertainty, the state space increased by a factor of $(\text{dim}_x * \text{dim}_y)^{(\text{num entities})} = 15,625$. This larger state space was required to accommodate all possible entity locations, and it resulted in the POMDP becoming intractable for the offline solvers.

We accommodated for the large state space by implementing POMCP. This online method uses particle filters instead of Bayesian updates and allows the solver to be much faster and operate on large state spaces. The off the shelf version of POMCP that we implemented first used a basic particle filter that was unweighted. This unweighted particle filter most likely ran into particle depletion because the observation space was quite large with a support of 64,000 unique observations. Even though we used as many as 10,000 particles, there would still be particle depletion because there were so many possible observations and particles with incorrect observations are rejected. To alleviate this, the solver was updated to use a bootstrap filter, which is a form of weighted particle filter that resamples according to the likelihood of particles being the true state.

C. Level 3 - Further POMDP Modifications

After achieving our base level 2 requirements of uncertain locations and identities for the entities, we wanted to explore other reward and action functions. The main feature that we added was a "memory" for the drone to store photos. Essentially the drone could take up to $n_{\text{max photos}}$ and would get rewards for each unique entity it photographed. In terms of the POMDP, a "photo" action could be taken by the drone, which would then store a value indicating what the photo was of (what entity was present on the square that it photographed).

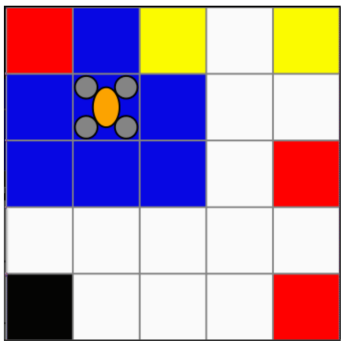

Fig. 2: Level 3 Grid World Formulation

1) State Encoding: The most challenging aspect of encoding the state for Level 3 was keeping track of the photos taken by the drone in the environment. This was encoded as a vector with a length equal to the max number of photos that could be taken by the drone. Each element of this vector contained an integer which encoded the entity of which a "photo" was taken. If the "photo" action was taken on a non-entity grid square, a unique number not equal to any of the entity numbers was placed instead. A zero in this vector meant that a photo had not been taken yet in that element.

2) Rewards: With a similar reward function used in the earlier section of the report, the agent would rarely go to more than two target entities to take photos before exiting and receiving its rewards. To combat this, the reward for each additional photo of a unique Target increased. For example, the reward for the first photo would be 10, the next 20, and the third 30 for a total reward of 60. This incentivized the drone to stay active longer and seek out Targets before collecting its reward for exiting the environment with photos.

V. Results

We generated gifs to demonstrate the behavior from various solvers and POMDP setups. You can view them in a slide deck by clicking this link or going to the url in appendix A.

A. Uncertain Identities Only

In this case, the drone knew the location of all entities on the grid world, but had no knowledge of their identities. This allowed it to move directly toward the entities to observe them and then determine their identity.

With uncertainty only in the identity, we observed the drone approaching each entity, then sometimes hovering next to it to gather more measurements, and finally moving onto the next entity or moving over it and off the grid if the entity was actually the target in order to take a photograph. This behavior is demonstrated well in the "Corners" gif from the presentation.

The following tables contain some quantitative results of QMDP on many variations of our POMDP.

Grid Size	Average Reward	Std. Dev.	SEM	Average Time [s]
5x5	1.223	0.310	0.031	4.01e-3
6x6	1.084	0.534	0.053	3.46e-3
7x7	1.010	0.280	0.028	10.24e-3
10x10	0.760	0.304	0.030	66.94e-3

TABLE I: Average reward and average time per run for 100 runs with 3 entities using QMDP.

Grid Size	Average Reward	Std. Dev.	SEM	Average Time [s]
5x5	1.01	1.179	0.117	2.346
6x6	0.137	1.288	0.128	6.752
7x7	-0.513	0.880	0.088	9.364
10x10	-0.074	1.161	0.116	14.587

TABLE II: Average reward and average time per run for 100 runs with 5 entities using QMDP.

As you can see in tables I and II, the average reward decreases as the grid world grows. The drone has a harder time navigating/exploring the larger environment. The drone also performs worse if there are more entities added to the grid world. It seems to act on incorrect beliefs more often. Plus there can be more detectors and benign object to interfere with the mission.

B. Uncertain Positions and Identities

With uncertain positions and identities, The drone starts with no knowledge of the entities. Only by observing an entity within its field of view, can the drone determine where an entity is and what type it might be.

With this additional uncertainty, we observed the drone wandering around more–as is expected. The solver had a harder time solving the problem with more uncertainty, and more frequently produced a far from optimal policy. A typical example of the behavior in this set up can be seen in the "wander" gif in the presentation.

Grid Size	Average Reward	Standard Deviation	Average Time [s]
5x5	0.883	0.748	141.5
7x7	0.559	0.672	211.6

TABLE III: Average reward and average time per run for 100 runs with 3 entities using POMCP.

Comparing Tables I and III, the average reward decreases, and the average time increases by a factor of 10,000. This can be understood as the effect of the increased size of the state space due to the states needing to account for all possible permutations of the set of x-y coordinates for each entity. In addition, the large state space necessitates the use of an online solver such as POMCP as opposed to QMDP or SARSOP, so the policies found may not be completely optimal.

C. Varying POMCP Parameters

In order to determine what parameters work best with our problem and POMCP, the average rewards over a range of tuning parameters were investigated.

Figure 4 shows the affect of increasing tree queries on reward. The reward does increase slightly and has a decrease in variance, but this came with a heavy penalty in terms of time as shown in Figure 4.

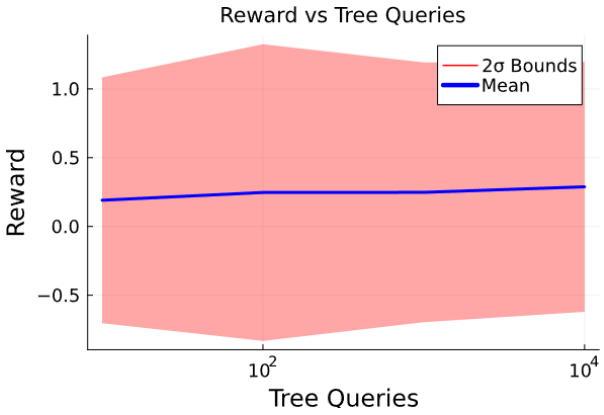

Fig. 3: Reward vs Tree Queries

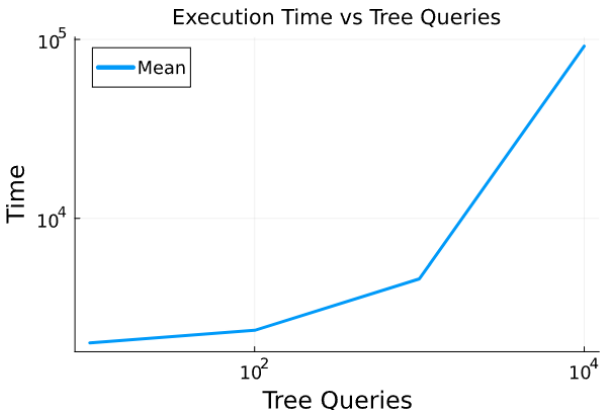

Fig. 4: Run Time [ms] vs Tree Queries

In addition to these two plots, we also investigated how exploration constant affected POMCP's performance as shown in Figure 5. Increasing the time constant did increase the average returned reward, although this also incurred a slight time penalty as the agent would explore more before deciding to exit.

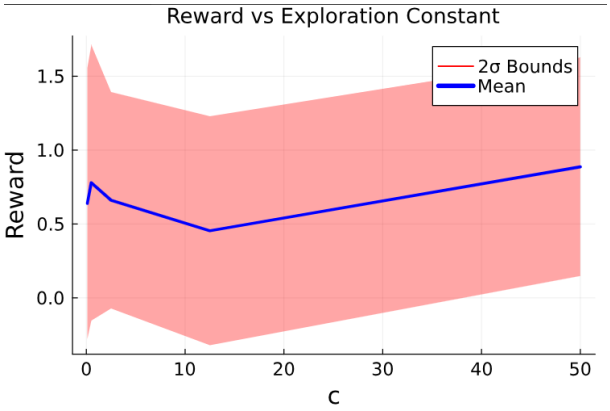

Fig. 5: Average Reward vs Exploration Constant

D. Level 3 Results

An example of a trajectory from the POMDP with memory constraints can be seen in the "Marco Polo" gif. This video shows a quadcopter seeking out entities and taking pictures of both Targets in the environment before exiting.

Results for multiple runs of the limited memory case with varying amounts of entities are shown below in table IV. These rewards are not comparable to the rewards achieved earlier because they were modified to assist the solvers. Unexpectedly, increasing the number of entities in the environment decreased the average reward. When looking at trajectories, when there were more than 3 entities, the drone began to have long stretches where it remained hovering instead of moving towards Targets or exiting the grid world, similar to the behavior shown in the "Sitting" gif. It is unclear why this was happening, but this is almost certainly the reason why the runs with 4 and 5 entities saw a decrease in total average reward because these runs more often ran into the max steps limit.

Number of Entities	Average Reward	Std. Dev.	Average Time [s]
3	1.97	1.99	14.5
4	0.259	0.568	15.3
5	1.18	1.14	12.5

TABLE IV: Average reward and average time per run for 50 runs for multiple photos on a 5×5 grid.

VI. Conclusion

This project successfully demonstrated that searching for multiple targets among a collection of unknown objects can be formulated as a POMDP with both the identities and positions of the entities being unknown.

Although we believe the solvers and our POMDP formulation largely worked as expected, there are still some remaining questions and challenges to address. With discrete spaces, we encountered the curse of dimensionality in some elements like the observation space. We struggled to run grid worlds larger than 7x7 with uncertain positions. If this project were to be applied to a real drone, we would need to find ways

to reduce or work around the large state/observation spaces. We also noticed some unexpected behavior particularly when there were uncertain positions. The drone appeared to become stuck when observing entities and continually hovered even when negative rewards were applied at each time step as a penalty for fuel consumption.

This formulation was based on a simple grid world, but the conclusions could be used as a starting point for real world drone missions. This type of POMDP formulation is applicable to many problems like search and rescue, defence, and agriculture. An application just needs to have some method to classify entity types from observations and model the predicted state to form a belief in order to make a POMDP formulation possible. Future work could focus on the interactions of multiple drones attempting to surveil an environment as a POMG, or the added complexity of moving entities in the environment. In addition, while a grid world is much easier to analyze and solve in a POMDP, more applicable formulations could be created by examining a continuous action and state space for this problem.

VII. Contributions and Release

The authors grant permission for this report to be posted publicly.

A. Team Member Contributions

- Luke
  - Coding
  - Report writing. Especially Level 3 and Solution approach items.
- Collin
  - Coding
  - Problem Formulation
  - Benchmarking
- Amanda
  - Report Writing/Editing
  - Research
  - Debugging

Acknowledgment

The authors would like to thank Professor Sunberg for his guidance on this project and plentiful contributions to the Julia language and POMDP packages.

References

[1] M. Svoreňová, M. Chmelík, K. Leahy, H. F. Eniser, K. Chatterjee, I. Černá, and C. Belta, "Temporal logic motion planning using pomdps with parity objectives: Case study paper," pp. 233–238, Association for Computing Machinery, 4 2015.

[2] R. Z. B. Bravo, A. Leiras, and F. L. C. Oliveira, "The use of uavs in humanitarian relief: An application of pomdp-based methodology for finding victims," *Production and Operations Management*, vol. 28, pp. 421–440, 2 2019.

[3] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," 2010.

[4] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," 1993.

[5] A. Hinas, R. Ragel, J. Roberts, and F. Gonzalez, "A framework for multiple ground target finding and inspection using a multirotor uas," *Sensors (Switzerland)*, vol. 20, 1 2020.

[6] M. Bouton, D. Asmar, and Z. Sunberg, "Dronesurveillance.jl." https://github.com/JuliaPOMDP/DroneSurveillance.jl, 2023.

Appendix

A. Useful Links

- **Presentation with GIFs**: https://docs.google.com/presentation/d/1pi3aLfFjQ-tpgFJh_sZG1QQo2a4Tjuq8/edit?usp=sharing&ouid=118080672912096410467&rtpof=true&sd=true
- **Github Repo**: https://github.com/nilloch/a5264-DroneProject/tree/master