2018 International Conference on Unmanned Aircraft Systems (ICUAS)
Dallas, TX, USA, June 12-15, 2018

# Autonomous UAV Swarm: Behavior Generation and Simulation

Ju Wang[1], Yong Tang[2], Jonathan Kavalen[3], Ahmed F. Abdelzaher[1], Sagar P. Pandit[1]
[1] *Engineering & Computer Science*
*Virginia State University*
*Petersburg, VA*
*jwang@vsu.edu*
[2] *Infoblox Inc,* [3] *University of Florida, Gainesville FL*

***Abstract*—We present a software framework to produce and simulate autonomous behaviors of UAV swarm tasked with search and reconnaissance missions. With proper control and motion planning, multiple-vehicle swarm allows for a more efficient and cost-effective search. The main challenge, however, is on the generation of the desired autonomous behaviors of the UAV swarm, with minimum centralized control and human intervention. The proposed framework allows UAV agents to adopt different behavior models and use a rule-based architecture to select suitable behavior that is best for given situations. In particular, the UAV's search and tracking behavior are treated by two different POMDP decision models to reflect different reward mechanisms. The proposed framework is evaluated in a SITL environment which boasts both realistic low-level flight dynamic and sensor inputs. A two-UAV teamed search and tracking scenario is presented to test the performance of the proposed model against an evasive ground target.**

Keywords: Autonomy, UAV swarm, SITL simulation, evasive target

## I. INTRODUCTION

Autonomy [9], [11] and swarming are centerpieces in future military UAV operations as well as civilian applications involving security, intrusion detection, and SAR (Search and Rescue). The obvious benefit includes accelerated search process and increased success probability. However to operate a UAV swarm efficiently, one needs to address different autonomous behaviors, sometime even with conflicting objectives, to achieve a high level of adaption and human-like cognitive behavior.

The human-like autonomy behavior should allow the UAV swarm to perform tasks as if each UAV is controlled by a human operator and make appropriate decisions. With the diverse set of on-board sensor asset and platform performance, a UAV might be carrying out different tasks such as scouting, close-in inspections, communication relaying, and target pursuing. The UAVs would intelligently take an appropriate role based on the situation. Such high degree of adaption and cooperation in complex scenario calls for innovative software solutions and immersive testing strategies. Ultimately, an intelligent UAV swarm system means fewer human operators, a human-friendly interface, and a large-scale operations.

### A. Search and Track Scenario

We use a 2-UAV search and tracking scenario to examine the desired autonomous behavior among a UAV team. The general goal is for the 2-UAV team to monitor a certain area of interest, called playbox, for the interested target and to track the target once it is identified. We further assume that the detection event is reciprocal, meaning that the target and the UAV can mutually detect each other, with different detection probability, when they are close enough. The target is assumed to have a simple evasive behavior: if it detects being monitored by the UAV, it will execute a probabilistic evasive maneuver to escape. In this example, we noted that autonomous behavior can be interpreted at different levels. For instance, the S&T task could have two distinctive level: (1) Patrol and Search, where the UAV team does not have a high confidence in the target location and thus the objective is to patrol the entire playbox. (2) In a second case, the UAV team has a high confidence in the target location and the main objective becomes tracking the object in a narrowed area. It is clear that the latter scenario requires a more coordinated planning than the former.

In addition to the search-related mission objectives, the UAV must address other objectives during its mission such as survival, fuel checking, communication maintenance and response to directed commands from the base. One approach to address the coexistence of multiple objectives is to somehow assign each objective a numerical value and frame the problem in a unified optimization framework. The problem is thus translated to a version of optimum planning problem such as multi-agent POMDP [1] (assuming reliable communication) or dec-POMDP [2] if the communication channel is not reliable.

We take a different approach where multiple objectives are treated as separate but schedulable tasks with dynamic priorities. For instance, a refueling objective will have a low priority until the fuel becomes critically low. Put in another word, if the fuel level is good, then the refueling objective is not considered in the UAV's reasoning. This is different than the unified approaches where all objectives contribute to an expected reward function during all phase of reasoning.

Our contribution is twofold: (1) We propose a two-level

978-1-5386-1353-5/18/$31.00 ©2018 IEEE

autonomy behavior architecture to model interaction and transition between different behavior. In our framework, each UAV utilizes an agent architecture [10] as autonomy engine, where autonomous behaviors are encoded as part of its reasoning base and selected at a proper time. Our framework allows different autonomous behavior models to co-exist and triggered at a different time. (2) We describe a Search behavior and a Tracking behavior using POMDP framework and evaluate the overall "search and tracking behavior" in simulations. The seperated models might be more close to the way human would operate under multiple objectives. In particular, behaviors are represented in the forms of knowledge/rules that encode different objectives and constraints, such as communication preservation, collision avoidance, and the need to cover new search area. These rules will result in a tactical level behavior such as loitering, circle, survey pattern to serve a local or global search goal. The UAV behavior is driven by new goals generated as new situation data arrives.

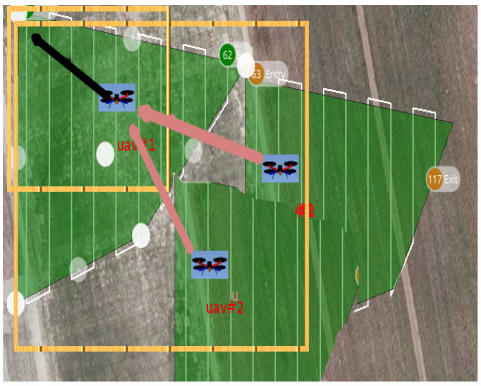

Figure 1. UAV-team coordinated search: a 3-UAV team must maintain unbroken video downlink while searching the target area. The target area is the green area.

To be able to evaluate the swarm autonomous behavior in a realistic scenario, our system is designed to be immersive and based on actual air platform. For this reason, we use a modified PX4 autopilot firmware in a Software-In-The-Loop (SITL) configuration as the low-level component of the UAV agent. The UAV's perception of the world is provided by interacting with Gazebosim whose simulated sensors include IMU, camera, and GPS.

In the rest of the paper, Section 3 describes the proposed software framework and the LTL specifications for behavior selection. Section 4 presents two POMDP models for search and tracking autonomy behavior. Section 5 discusses simulation testbed and section 6 discuss the results of a testing scenario.

## II. Background and Literature Review

Unmanned Aerial Vehicle (UAV) is being used in many military, civilian, and commercial applications. With proper onboard sensor, UAVs offer a practical solution for large-scale search and rescue mission in single UAV or teamed operation [5], [8].

Several AI cognitive frameworks could be used to provide autonomous UAV behavior. One example is the rule-based Soar Cognitive Architecture in emulated combat flight simulation [9], which has been demonstrated to produce human-like autonomous behavior piloting a military fixed-wing jet. Other potential autonomous solution might utilize tools/ideas from symbolic logic system ant colony optimization, or game theory inspired learning system. However, there has not been a widely accepted theory that offers diverse autonomous behaviors for a large swarm.

Another important aspect of agent behavior is its formal specification and verification of useful property such as keep-out-no-fly-zone for a planned temporal flying path. This is often obtained with Linear Temporal Logic (LTL) and its many variants [13], [14]. Starting from a set of atomic propositions, an LTL formula is composed using the standard logical operators and temporal operators such as eventually, next, and $until(U)$ to express temporal events.

There is significant literature in optimum planning policy for the multi-agent system [1], [2] under stochastic world model and reward structure. The partially observed single-agent case is originally discussed by Kaelbling [1] in the POMDP (Partial Observable Markov Decision Process) framework. Recent works seem to focus on dec-POMDP [2] models where agents only have access to local observations and beliefs without full knowledge of the true world state. The complexity of dec-POMDP problems turns out to be doubly exponential of the length of the horizon, which make exact solution intractable for a long planning horizon. A reduced version of dec-POMDP takes a further assumption of inter-agent communication to ease the computing of the joint-believe. In [12], an algorithm to solve multiple objectives is discussed using MDP and methods from formal verification. While dec-POMDP could be used for both search and tracking behavior, its computation requirement is too large to be carried out in realtime on the UAV onboard computer. We thus choose to use POMDP model for local decision making at approperate state.

Another notable area, there are also interests in transition between human-directed behaviors and autonomous behaviors. One such work proposes the concept of Autonomous decision points (ADP) where a human pilot yield decisions to avoid an obstacle or a denied air space [7].

## III. Autonomous Behavior Architecture

The structure of the proposed system is illustrated in Figure 2. A key concept of our agent behavior mechanism is "behavior modules". It is our belief that many autonomous behaviors are fundamentally different, and the overall agent behavior might be better served if we treat them differently with different reasoning models. This is intuitively backed

by the way human acts: our mind functions at a completely different "gear" between watching a movie and writing a report, and none will be served properly if we interweave the two tasks together.

All UAV agents share a common initial working memory including world state and decision policies generated by the central mission controller. The overall behavior hierarchy is based on the principle of "central-aided-planning-distributed- execution". The central part is usually assigned to the ground station, which collects/updates/disseminates the common memory of world state. The common memory will also contain a set of rules that must be observed at each agent as they plan local behavior. For example, a safety rule can be specified in Linear Temporal Logic (LTL) formula for collision avoidance or task prioritizing among agents.

This common memory is periodically disseminated to all UAVs. Once launched and reaching its initial position, each UAV will use its local autonomy engine to evaluate the LTL rules and select behaviors based on locally assessed conditions. Each UAV agent executes an outer-loop that consists of three steps: (1) observing, (2) updating the world state and internal state, (3) and behavior selecting (or persisting).

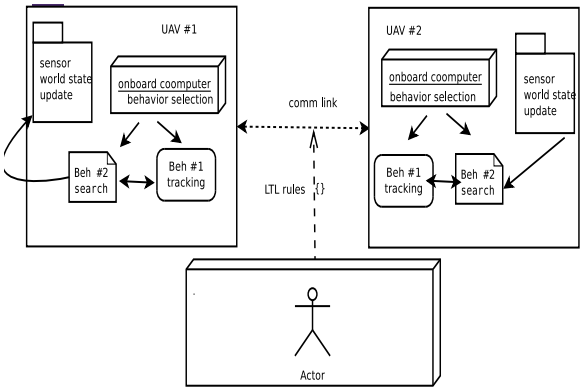


Figure 2. UAV agent architecture. UAV agents communicate with a ground station to update its world state and select appropriate behavior.

### A. High-level rule specification with LTL

We now discuss how high-level behavior can be controlled by specifying LTL rules. We assume that readers are familiar with LTL terminology. For quick reference, we point to the work by Christos [13]. LTL is an extension of logic rule system such that certain property can be evaluated in timeline to determine corresponding behavior. A basic LTL formula is defined over a set of atomic propositions AP and several boolean and temporal operators. LTL formulas are formed according to the following grammar [14]: $\phi ::= true|a|\phi \wedge \phi|\ \not\phi|\bigcirc \phi|\phi \bigcup \phi$ , where $a \in \mathcal{AP}$ and two temporal operators $\bigcirc$ (next), $\bigcup$ (until) allow assertion of sequential temporal events (paths). Other useful operators include $\Box$ (always), and $\diamond$ (eventually), but are not used here.

The system level LTL rules $\phi$ serve three purposes:

1) Provide critical constraints and system level properties for local agents to plan different behaviors include safety, search, tracking, and collision avoidance.
2) Support directed behavior: This type of autonomy allows single-point control of a part of the swarm to be engaged in a specific swarm mode quickly, but without depending on micromanagement from human operators.
3) Serve as a formal mechanism for local agents to switch/schedule behavior. For example, a search mission might start as random best effort. After a positive visual detection event, ground control might want to perform a formation-search where several UAVs are lined in a tight formation, which is more useful to track a moving target. This could be done by transmitting a new LTL rule to the involving agents.

For each agent, we define a set of LTL formulas $\phi_i$, called agent rule, to regulate its behavior. The overall behavior of the swarm is then captured by the joint agent rule $\Psi = (\phi_i)$, i $\in \{1, \ldots, N\}$. The agent can assess if a candidate behavior violated $\phi_i$ based on the expected results (paths) of the behavior. To allow agents to perform emergency or direct behavior, We extend the basic LTL formula to include an optional behavior field $b_r$ which the agent might choose to execute if the proposition is violated.

We point out that unlike other systems, the LTL specification is not stationary and can even be created by a human operator in situ. For instance, a safety rule for low-speed patrol behavior might be *The number of the agent within any* $100\ meter$ *radius airspace must be less than three*. But at tracking behavior, the safety rule might be changed to reduce the number of vehicles to two due to more aggressive in-air maneuvers.

*1) Example Agent Rules:* The set of axiom $\mathcal{AP}$ is defined on top of the world state and agent state. We list some of the state variables and $\mathcal{AP}$ axioms in the table below:

Table I
SUMMARY OF LTL TERMINOLOGY

| | |
|---|---|
| $pos_i$ | most recent location of agent i |
| $fr_i$ | fuel range of agent i |
| $d_i$ | distance from home for agent i |
| $d_s$ | safety distance between agent |
| $goto(z)$ | direct goto waypoint axiom |
| $\alpha_1 \equiv$ $dist(pos_i, pos_j) > d_s$ | agent i and j are in safe distance |
| $\alpha_2(i) \equiv$ $d_i < d_s$ | agent i fuel enough to return home |
| $\alpha_3 \equiv dist(pos_i, pos_j) < 6.d_s$ | agent i and j are nearby |
| behavior modules | $b - collision, b - refuel$ $b - search, b - track$ $b - formation$ |
| $\vdash (bool)$ | recommended behavior if LTL formula violated or enforced |

We can define the following LTL agent rules:

- *Collision policy for agent i*: All other UAVs must be separated by the minimum of safety distance: holds throughout the execution of the system:

$$\forall j \neq i \Box \alpha_1(i,j)$$

$$\vdash (false) b_r \ = \ bcollision$$

- *Fuel policy*: Fuel must be sufficient to return: holds throughout the execution of the system:

$$\forall i \Box \alpha_2(i)$$

$$\vdash (false) b_r \ = \ brefuel$$

- *spread-out search policy*: avoid the area if there is another agent that is closer, until under direct command: holds throughout the execution of the system:

$$\forall z \Box pos_i \in z \rightarrow \neg (pos_j \in z) \bigcup goto(z)$$

- *joint tracking policy*: if there is a nearby agent in tracking mode, joint as a new tracking member : holds throughout the execution of the system:

$$\exists j \alpha_3(i,j) \wedge b_i == btracking$$

$$\vdash (true) b_r = btrack$$

### B. Behavior selection at the local agent

Since our system allows different agent behaviors to be used in different states, each behavior module must be evaluated against the agent rules. The decision for selecting the most suitable behavior is made at the individual agent to achieve a true distributed decision process. The selection process consists of the following steps:

1) check if $\phi_i$ is violated by current agent state. if any of the LTL formulae is violated, the corresponding behavior is pushed to the emergency queue.
2) obtain a list of pending behaviors, referred to as $H$. For each $b_k \in H$, calculate a list of T-step path $p_k^m$ which predicts the expected future states in the next T steps.
3) Evaluate all candidate paths against the agent rules $\phi_i$. A risk score is associated with each candidate behavior based on the number of violation. The risk score is a necessary utility since many autonomous behaviors are non-deterministic, such as the search behavior discussed later.
4) if the emergency queue is not empty, execute the behavior on top of the emergency queue. The behavior in the emergency queue is non-preemptive; else a behavior in the regular queue is selected.

Internally, the agent maintains two separate queues to represent routing objectives and emergency objectives. The emergency objectives are normally empty, until an emergency appears, such as fuel drop below the threshold, motor fail, or imminent collision warning. A simplified goal selection logic is described in algorithm 1. In terms of complexity, the most time-consuming part is in line 11 where the possible path of each behavior module is calculated. With both the search and tracking behavior modeled by POMDP, the number of T-step paths is exponential $O(|S|^{|O|^T})$. Due to space limitation, we defer the treatment of this issue to another publication. Throughout this study, the problem is circumvented by limiting the number of state transition and the time step $T$.

**Algorithm 1** Behavior Selection at agent(i)

1: $Pos_t$: vector of UAV grid locations at time $t$
2: $Br_t$: belief of other UAV behavior
3: $Q_r = \emptyset$: sorted routine queue according to $risk$ $Q_e = \emptyset$: emergency behavior queue
4: $Q_r = \{bsearch\}$: initial behavior
5: **while** true **do**
6: update $Pos$ and $Br$.
7: update check local detection event
8: evaluate all $\phi_i(Pos, Br) in \phi$
9: insert recommended behavior to $Q_e$
10: **for** $b \in Q_r$ **do**
11: update $path(b)$= expected T-step paths
12: update $risk(b, path)$ risk value associated with $path(b)$
13: **if** target detected **then**
14: $new\ b_{track}(targetloc)$
15: push($b_{track}$, $Q_e$)
16: **if** target lost **then**
17: $new\ b_{search}$
18: push($b_search$, $Q_r$)
19: select the top of the $Q_e$ if not empty
20: select the top of the $Q_r$ if $Q_e$ empty

## IV. POMDP Models for Search and Tracking Behavior

As discussed in the introduction, the main reason to split the search and track behaviors is that they are inherently different in a team setting. A team engaged in searching mode would wish to spread out to increase the change of finding the target. A team in tracking mode would wish to perform a more coordinated formation to reduce the probability of losing the target.

Both behaviors can be modeled by POMDP because of the statistical nature of state transition and observation. To avoid confusion of the terminology, the search POMDP model has a prefix $S$ in front of each symbol, while the tracking POMDP start with a prefix $T$. The tracking behavior captures the UAV team strategy when the position of the target is located by one of the UAV team. Due to the mutual detection assumption, a high fidelity detection by UAV also

infers a high probability of the target detecting UAV and triggering the evasive behavior.

We choose to use POMDP to model this behavior since, under this setting, the expected reward is intuitive and easy to define: the transition probability of the target location is highly localized, which provide a good ground for UAV agent belief.

### A. Search behavior model description

Using terminology in [1], the POMDP can be described as a tuple $< SS, SA, ST, SR, S\Omega, SO >$, where

- $SS = SS_1 \times SS_2 \times SS_3$ is a finite set of joint states of the world. $SS_i = \{(i,j)|i \in 1..N, j \in 1,2\}$ represents the location of the two UAV . The target location $SS_3$ has a fixed altitude at the ground station. The location of a UAV is indexed by an integer number from 1 to N. The state of UAV also includes the altitude of the UAV as part of the state. This is necessary since the altitude also affects the probability of detection. Two UAV altitude is allowed: 20 meters and 50 meters.
- $SA = SA_1 \times SA_2$ is a finite set of actions. For each $SA_i$, the action is to move the UAV to one of the nine neighboring $(grid, altitude)$ combination.
- $ST$ defines the state transition probability. The outcome of an action is non-deterministic in that the target might randomly walk and change its location.
- $SR : SS \times SA \rightarrow \Re$ is the reward function. $SR(s,a)$ reflect a joint-detection event:

$$SR(s,a) = \begin{cases} 1 & cocell(a_s^1, a_s^3) \ or \ cocell(a_s^2, a_s^3) \\ 0 & else \end{cases} \quad (1)$$

. $a_s^1$ denote the location of UAV #1 after applying action $a$ on current state $s$. It is also possible to associate the reward function to be proportional to the detection probability. In practice, we found that the search behavior remains insensitive to such change.

It is noteworthy that the joint action set $SA$ grow exponentially as the number of UAV in the swarm. For a 2-UAV team, eighty-one possible outcome must be evaluated.

The reward function $cocell()$ is defined over a state $s \in SS$ to report if a UAV agent is in the same cell as the target. Under the above model, the intuitive search behavior is such that the two UAV randomly search while avoid the cells that are recently visited.

### B. Tracking behavior model description

Similarly, the tracking POMDP can be described as a tuple $< TS, TA, TT, TR, T\Omega, TO >$, where

- $TS = TS_1 \times TS_2 \times TS_3$ is a finite set of joint states of the world similar to $SS$, but with a more smaller grid size. $TS_i = \{(i,j)|i \in 1..9, j \in 1,2\}$ represents the location of the two UAV. The playbox is now $1/9 - th$ of that of the Searching POMDP playbox.
- $TT$ defines the state transition probability. The outcome of an action here introduce the uncertainty by the target's evasive behavior. In particular, the target will try to escape to one of the neighboring locations if it is detected.
- $T\Omega$ is the set of observation. We assume that the UAV team locations are perfectly observable due to the assumption of communication. The location of the target, however, is not fully observable. This is reasonable assumption since the accurate location of the target is subject to camera orientation and aircraft vibration noise during the observation.
- $TR : TS \times TA \rightarrow \Re$ is the reward function for the tracking mode. Here we make $TR(s,a)$ favor a single detection over zero of double detection by:

$$TR(s,a) = \begin{cases} Pr_d(a_s^3, a_s^1), & if cocell(a_s^1) \wedge \not cocell(a_s^2) \\ Pr_d(a_s^3, a_s^2), & if cocell(a_s^2) \wedge \not cocell(a_s^1) \\ \frac{Pr_d(a_s^3,a_s^2)+Pr_d(a_s^3,a_s^1)}{2}, & if cocell(a_s^2) \wedge cocell(a_s^1) \\ 0, & else \end{cases}$$

The probability of detection $Pr_d(a_s^3, a_s^1)$ is none zero if the UAV and the target are in the same cell, but its value is further determined by the altitude of the UAV and the camera model.

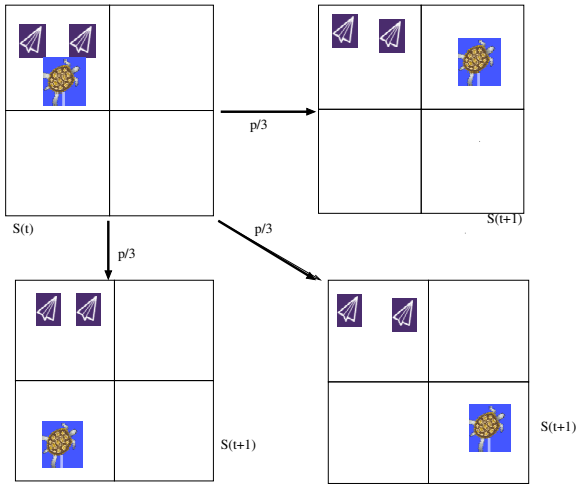


Figure 3. The target transition probability with target evading illustration: the target might evade to the other three cells with equal probability.

## V. Simulation Environment

### A. Swarm SITL architecture

We now describe the simulation environment where the agent behavior is tested. The system consists of three main

components: (1) simulated UAVs, (2) Ground control station (GCS), and (3) gazebosim external simulator.

For each simulated UAV, its software structure is further divided into two layers: the low level is a platform-dependent process which performs flight dynamic related tasks which are essential to keep the UAV afloat. The low layer in our environment is in factor a SITL instance of PX4 autopilot firmware, modified in such a way so it can communicate with the physical simulator as well as high-level layer. Most of the message in and out of the low layer is Mavlink message.

The high layer process of the simulated UAV agent executes the behavior engine as its main thread. The high layer communicates with lower layer process to obtain local sensor observation, such as local GPS reading and altitude. This layer also reads camera data and communicates with the GCS and other UAV agents. In a real deployment, the high layer process would be running on an actual onboard computer. In our simulation environment, the high layer process of the UAV is typically run on a desktop computer.

Each UAV also has a physical instance running in the gazebosim simulator. The physical instance provides simulated physical data to the low layer process, including IMU, GPS location, and camera data. The actuation commands issued from the SITL instance will be faithfully executed to produce results. Due to the significant amount of data (400 Hz IMU data and 30 HZ camera data per UAV instance) in and out of the physical simulator, our current experiments is limited to two UAV instances.

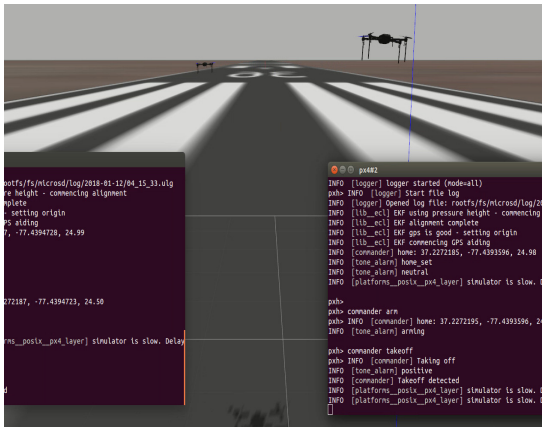

Figure 4. Two simulated UAVs takeoff in gazebosim. The UAV SITL instance controls the flight dynamic and communicates with its behavior agent via Mavlink protocol.

The UAV physical model is a modified 3DRobotics IRIS controlled by a Pixhawk PX4 autopilot. An embedded computer with Ubuntu 14.04 and ROS Hydro provides mission control and communication with a ground station.

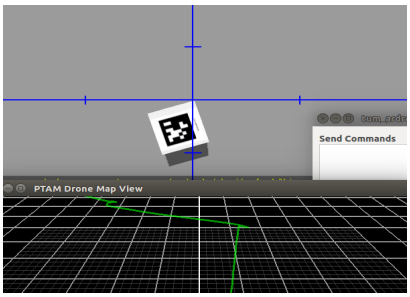


Figure 5. UAV coordinated search bottom camera feed with detected apriltag at $h = 10$ meters.

### B. *Sensor model and detection probability*

To simplify the overall simulation complexity, we choose a simple target detection method based on Apriltagn [3], which is a 2D design pattern that can be detected from a distance without the requirement of heavy computing power. The target is a simulated vehicle with an Apriltag attached on top (Figure 5).

As in the real world image sensor, the simulated camera has a limited field-of-view and resolution. For instance, the GoPro Hero 4 has a vertical FOV of 94.4 and a horizontal FOV of 122.6 degrees. At a height of 20 meters, this gives us an approximate footprint of 44*72 $m^2$. Depending on the lighting of the UAV and the evasive movement of the ground target, the UAV might lose sight of the target. If a high altitude is used, the camera captures more acreage and might results in better evasive tolerance. However, the target will appear smaller and become more difficult to detect under poor lighting condition. The simulated camera provides image resolution of 640 pixels horizontally, and the image height is 480. The detection probability when different tag size is empirically obtained (Figure 6). At standard resolution camera model (640x480) and an altitude of 10 meters, the detection rate is about 90 %. More data on the detection probability is shown in Figure 6

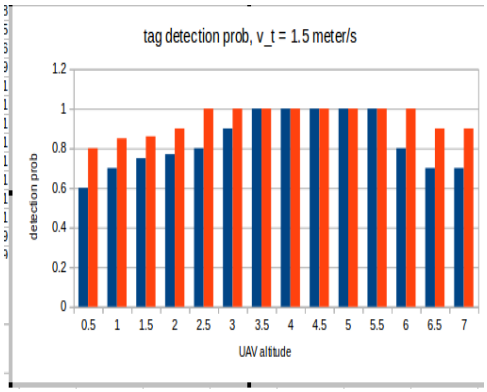


Figure 6. Detection probability of target from different altitude, note that if the UAV altitude is too low, it risks losing track of the target due to the narrow field of view.

## VI. Numerial results

To test the search and tracking behavior of the UAV team, a test case was designed in a playbox where a target is placed at a random location. The simulation is performed in a desktop computer with an eight-core Intel i7 CPU and 16 GB memory. Two UAVs instances are spawned to represent the searching UAVs. Each UAV instance consists of a PX4 SITL thread and a behavior thread. Each UAV instance communicate with the corresponding gazebo UAV via a specified UDP port.

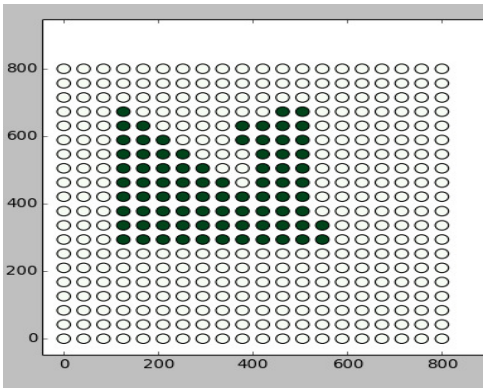


Figure 7. The playbox represents part of the world state and is a shared among all UAVs. The UAV boundary is marked.

In a simulation run, the UAVs take off from the base and head for their initial positions to conduct search behavior. The altitudes of the UAV is initially set at 10 meters.

Figure 8 shows the trajectory of a two-UAV team conduction a search-then-tracking mission. Initially, the target is not detected and both UAVs select search behavior, but each chooses a different area due to the LTL rule $\alpha_3$.

Figure 8(b) shows the trajectory of the UAV teams after the target is detected by one UAV and both UAVs are engaged in tracking behavior, while the target starts its evasive movement. For clarity, the trajectory of the tracked target is not shown. The trajectory shows a clear teaming behavior during the tracking session. In average, the target is detected 80% of the time during the tracking session. The results show that the 2-UAV team maintain a loosely connected team formation during tracking, similar to what human pilot would do.

## VII. CONCLUSIONS

We present a software framework to generate and simulate autonomous behaviors of a UAV team in search and tracking scenario. Our method uses a modified LTL specification for agent behavior selection. We use two separate POMDP models to regulate search behavior and tracking behavior. Our test results show that the proposed framework allows different models to be integrated well and the resulting behavior is human-friendly to interpret.

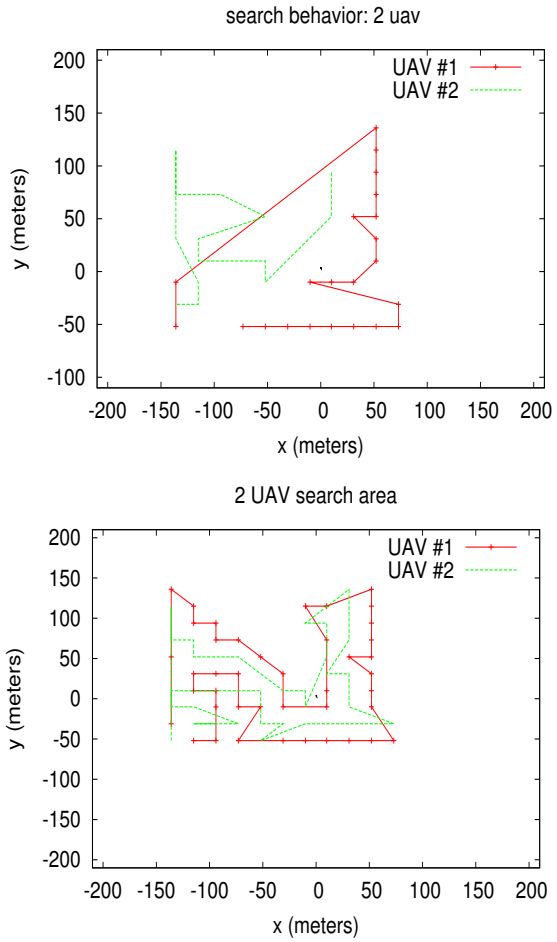


Figure 8. UAV team behavior switching from search mode to tracking mode: (a) initially two UAVs each conduct patrol type of random walk search, (b) after detection, the two UAVs perform a tracking formation

## References

[1] L.P. Kaelbing, et al, Planning and acting in partially observable stochastic domains, Artificial Intelligence, 101 (1998), 99-134.

[2] Frans A. Oliehoek, Christopher Amato, A Concise Introduction to Decentralized POMDPs, Springer, 2016.

[3] Edwin Olson, Johannes Strom, Rob Goeddel, Ryan Morton, Pradeep Ranganathan, and Andrew Richardson, Exploration and mapping with autonomous robot teams, ACM Commun, 56, 3 (March 2013), 62-70.

[4] Barraquand J, Kavraki LE, Latombe JC, Li TY, Motwani R, Raghavan P (1997) A random sampling scheme for path planning. Int. J. of Robotics Research, 16(6):759774.

[5] A. Kleiner and A. Kolling, Guaranteed search with large teams of unmanned aerial vehicles Proc of IEEE ICRA 2013.

[6] J. Wang, et al, Optimizing Ground Vehicle Tracking using Unmanned Aerial Vehicle and Embedded Apriltag Design,

The 2016 International Conference on Computational Science and Computational Intelligence, 2016, Las Vegas.

[7] Chimpalthradi R. Ashokkumar, et al, Data science for decision aiding UAV control, Proc of ICUAV 2017.

[8] N. Dinnbier, et al, Target detection using gaussian mixture models and fourier transforms for UAV maritime search and rescue, Proc of ICUAV 2017.

[9] R.M Jones, J.E. Laird, P.E Nielsen, K.J Coulter., P.G Kenny, and F.V. Koss, Automated intelligent pilots for combat flight simulation, AI Magazine, 20(1), 27-42.

[10] Y. Cao, W. Yu, W. Ren, G. Chen, An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination. IEEE Transactions on Industrial Informatics, 9 (1), pp. 427-438.

[11] P. Haney and J. Derenick, Methodologies for decentralized control of networked autonomous Vehicles, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 2015, pp. 127132.

[12] S. Feyzabadi, S. Carpin, Multi-Objective Planning with Multiple High Level Task Specifications, IEEE ICRA 2016.

[13] K. Christos, Z. Xu and Dimos V. Dimarogonas, Decentralized Motion Planning with Collision Avoidance for a Team of UAVs under High Level Goals, IEEE ICRA 2017.

[14] T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli, and D. Rus. Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications. In Proc. of ICRA, pages 50115018, 2013.