UAV Autonomous Landing on a Moving Platform

Xinyu Ren
Stanford University
hughrxy@stanford.edu

Bilan Yang
Stanford University
bilany@stanford.edu

Chaonan Ye
Stanford University
yec0214@stanford.edu

Abstract

In this work, we aim to find optimal trajectories to achieve an autonomous landing of an Unmanned Aerial Vehicle (UAV) with a camera on a moving ground platform in stochastic environments. We consider both Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) to formulate the landing problem with different assumptions of the environment. We solve the MDP problem with the Monte Carlo Tree Search (MCTS); for the POMDP problem, we use the partially observable Monte Carlo planning with observation widening (POMCPOW) algorithm to handle continuous state, action, and observation spaces. Further, with the MDP model, we analyze the performance of the model for 36 different UAV initial locations and reasons of failure of landings. Also, we compare the performance of the model with the different target moving speeds. With the POMDP model, we simulated the UAV landing scenario and verified our model's capability of tackling the state uncertainty. Finally, we conclude with some observations from both POMDP and MDP models.

1 Introduction

We consider the problem of a UAV landing problem under uncertainty. A UAV is expected to land through an optimal trajectory from some initial locations. Normally, there is a two-step plan done by the UAV: first computing a sequence of actions that will take it landing on the target, and then executing it. For simplifying the question, we assume the UAV will successfully execute every action computed by planning stage even though it's too ideal in the real world.

First, we define this problem based on an MDP model (Section 3)[8] in an ideally observable environment assuming that the camera sensor is able to fully observe the target platform. In other words, states, rewards and transition function are always known. And then, we formulate it as a POMDP problem (Section 4) when the target is only partially observable by the camera because of the field-of-view (FOV) constraint. Accordingly, the state information is only partially known. Considering the computational cost, we choose MCTS[4][6], one of the most successful sampling-based online approaches in recent year, be the solver for the MDP model, and POMCPOW[10], an online algorithm for POMDPs with continuous state, action, and observation spaces, be the solver for the POMDP model. Finally, we conclude with some observations for both types of problem in Section 5.

2 Related Work

As we mentioned early, to reach the moving target, the UAV does two steps, which is motion planning and execution. To solve such problem, many works formulated it as an MDP based problem. For example, [3] describes an MDP-based planning method that uses a hierarchic representation of the robot's state space to solve both planning and execution problems. [1] presents an approach

Preprint. Work in progress.

to improve the traditional MPD methods by non-random initialization of the algorithms with a significant reduction of calculations at each iteration.

Besides, more and more people utilize a POMDP formulation to solve the motion planning problem in a stochastic environment whose state is partially observed. [11] uses a Monte Carlo algorithm for learning to act in a POMDP model with real-valued state and action spaces. [9] focuses on online approaches that alleviate the computational complexity by computing good local policies at each decision step to solve POMDP. More recently, [12] presents a new approach to motion planning under sensing and motion uncertainty by computing a locally optimal solution to a continuous POMDP problem.

To alleviate the computational cost, we utilize MCTS[6] to be the solver for the MDP problem. MCTS [2] is used to find optimal decisions in a given domain by taking random samples in the action space and building a search tree according to the rewards. Although it saves time and storage, tuning the MCTS solver is not an easy task. We have to carefully choose a proper balance between exploration and exploitation in the selection step of MCTS[5]. At the other siede, even though many offline methods have been developed to solve small and moderately sized POMDPs[7], we choose POMCPOW, a new online method, to handle the POMDP with continuous state, action, and observation spaces[10].

3 The MDP Landing Problem

In this section, we introduce the UAV landing problem represented as MDP=MDP(S, A, T, R), *i.e.* we set up clear transition and reward functions (no model uncertainty). Also, we assume that both the UAV and the target coordinates have been known without state uncertainty. This section first explains the MDP formulation by describing four essential elements: state, action, transition, and reward. The state, action, and transition functions are built according to the real UAV dynamic model. Besides, several rewards are designed to realize different control purposes. Then applying the online solver MCTS, we solve the MDP instance for various initial conditions. Finally, we qualitatively discuss the rationality of results based on the problem formulation.

3.1 MDP Formulation

A UAV landing to a moving target should choose a series of sequential actions considering relative coordinates between the current UAV and the target. This control task could be represented as an MDP network as shown in Fig.1-1. Notice that the target position is dependent only on the initial setting but not controlling actions, while we select different UAV actions to adjust UAV space coordinates for a reasonable landing trajectory. However, UAV and target positions jointly (typically relative position) decide the reward for each time step. We will discuss specific details of the MDP problem in the following subsections.

3.1.1 UAV and Target Dynamic: state, action and generative transition

Our world is simplified as a 3D Cartesian coordinate so that we can conveniently represent any physical value as a three components vector. In this world, obviously, the UAV state vector is a three-element vector (x_u, y_u, z_u) plus a heading scalar θ_u in xy-plane (tilts and flips are not considered). And the state of a target is a two-element vector (x_t, y_t) representing its location.

In addition, physically we are capable to control the velocity (both speed and orientation) of the UAV to achieve the next possible state. Therefore, we utilize a three-element vector (horizontal speed, angular speed, vertical speed) $= (v_{xy}, \omega, v_z)$ as an action. It is necessary to clarify that the representation of the action vector is based on the cylindrical coordinate. This setting is to make the action vector more reasonable because in reality, we access to control the heading and xy-plane speed other than the x-speed and y-speed separately. However, it won't affect the formulation of the problem in Cartesian coordinate with a naive transformation.

We assume that for each time step (dt = 1), the target movement, parameterized by v_{tx}, v_{ty} and σ, is uniformly linear with the addition of the Gaussian white noise. On the other hand, the UAV takes a

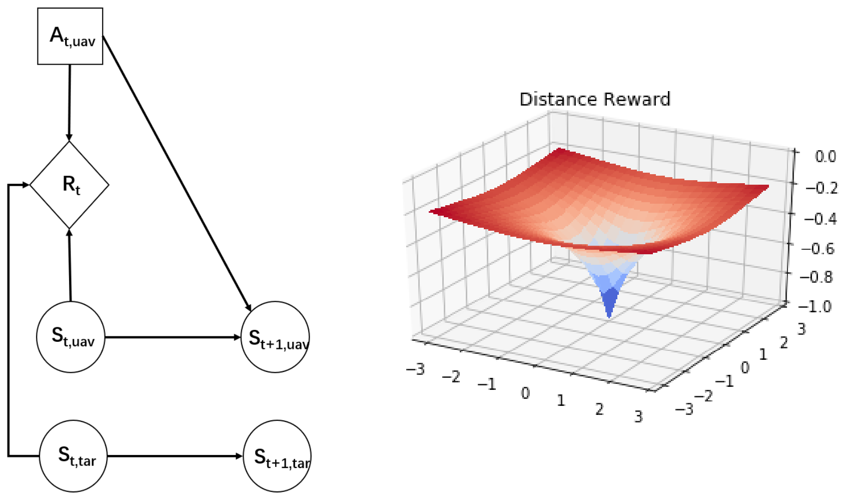

Figure 1: Left: MDP Network Representation of UAV Landing Problem. Right: Distance Reward as a function of x and y

uniformly linear motion with current actions as velocity, so the transition $T(s', s, a)$ is a generative model shown as below:

$$\begin{aligned}
x_u &= x_u + v_{xy}\cos\theta_u \\
y_u &= y_u + v_{xy}\sin\theta_u \\
z_u &= z_u + v_z \\
&\theta_u + \omega \\
x_t &= x_t + v_{tx} + noise \\
y_t &= y_t + v_{ty} + noise
\end{aligned}$$

In the algorithm, the state is continuous around all space, but the action is discrete since MCTS requires a potential action set when building the tree. The specific value of action is: $v_{xy} = -0.5 : 0.05 : 0.5$, $v_z = -1.0 : 0.05 : -0.05$ and $\omega = -\pi/3 : \pi/6 : \pi/3$. Also, it is worth to mention that the non-positive v_z would be meaningful when formulating rewards.

3.1.2 Landing Setting: reward and two-step landing

The physical motion decides the UAV dynamic process, yet it is also necessary to figure out what purpose is expected to achieve, which could be implemented by setting several rewards under different conditions. In our case, action reward, out-scene reward, landing reward as well as distance reward lead UAV to land on the moving target. Action reward is set as -0.5 with discount factor 0.95 so as to penalize the case when the trajectory takes redundant walks. Moreover, negative out-scene reward is assigned to the out-of-boundary trajectory (i.e. $0 < x, y, z < 10$). At the same time, the large landing reward (1000) encourages the UAV to land closely to the target (the distance in xy-plane is less than landing radius $r = 1$). Also, distance rewards, representing as $1/(1+r) = 1/(1+\sqrt{(x_t - x_u)^2 + (y_t - y_u)^2})$, induce the UAV to stay directly above the target. Additionally, we use 1+r, instead of directly using r, to avoid the singularity at r=0. The reward as a function of x and y is plotted in Fig.1-1 (the reward is multiplied by -1 to indicate the landing process).

Based on the reward as described above, we are able to find the optimal action of a UAV always following the target but without landing. The reason for this is that, in such way, a UAV always obtain a high distance reward rather than a one-off landing reward. To fix the problem, we purpose the two-step landing by dividing the landing interval at $z = 1$. Within the boundary, we expect the UAV track down to the target from an its initial starting point based on the distance reward. Then

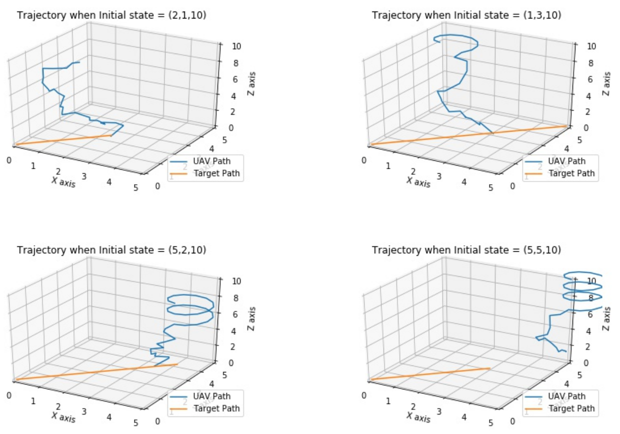

Figure 2: MDP Landing Trajectory with different initial locations

we turn off the distance reward when the UAV is out of the boundary. After that, the UAV will only evaluate landing reward and finally get to the target.

3.2 Results and Discussion

After building up the problem, a MCTS solver with iterations=800, depth=80 and explore constant=15 is utilized under various conditions. Here we discuss two landing configurations. For each case, we initialize the UAV at $z = 10$ for providing enough vertical space to take actions. A typical landing solution is shown in Fig.2-1. The UAV goes through a zigzag trajectory and meets the target at about $(2.3, 2.3)$. The following discussion exhibits the UAV landing solutions with different initial parameters.

3.2.1 Initial positions

With different initial positions, the trajectory exhibits new configurations and outcomes. One distinguished characteristic in Fig.2 is the circular path of the UAV. In Fig.2-2, the UAV rotates and almost forms a complete circle followed by a succesful landing. In Fig.2-3, the UAV rotates more and wait until the target get closer. In Fig.2-4, the UAV awaits much longer but fail to land. Notice that the circular motion occurs when the relative distance between the UAV and the target platform is large. There are two potential reasons associated with this phenomenon. First, the distance reward may not be strong enough to induce the UAV landing. Shown in Fig.1-2, when r gets larger, the valley tends to become a flattened region and the reward becomes too small to significantly influence actions. Second, the depth in MCTS isn't large enough to allow the algorithm to find out target positions. Actually, We observe that with an increased depth, the rotating-landing phenomenon happens earlier. These two possible reasons may jointly create the circular motion.

3.2.2 Target Velocity and Noise

Besides trying different initial positions, we also tune the parameters of the target movement in Fig.3. Referring to the successful landing trajectory as shown in Fig.3-1, it seems that the UAV cannot follow the target when the target speed increases. Also, if the distance between the UAV and the target is too far, again we would observe the circular motion. Moreover, when we increase noise to be std = 0.3, the UAV would deviate to the landing region even though the UAV attempts to follow the target at the beginning.

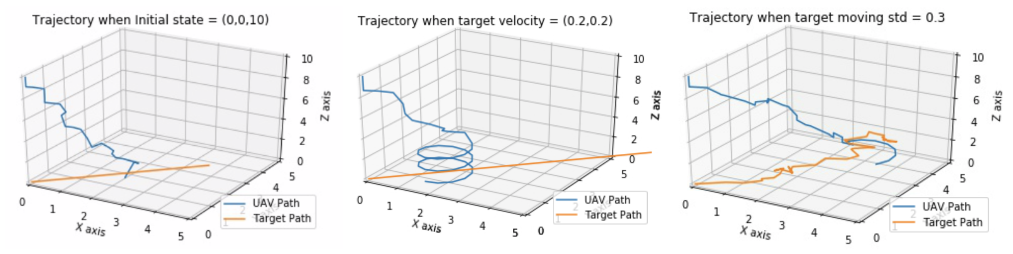

Figure 3: MDP Landing Trajectory with different target velocity and noise

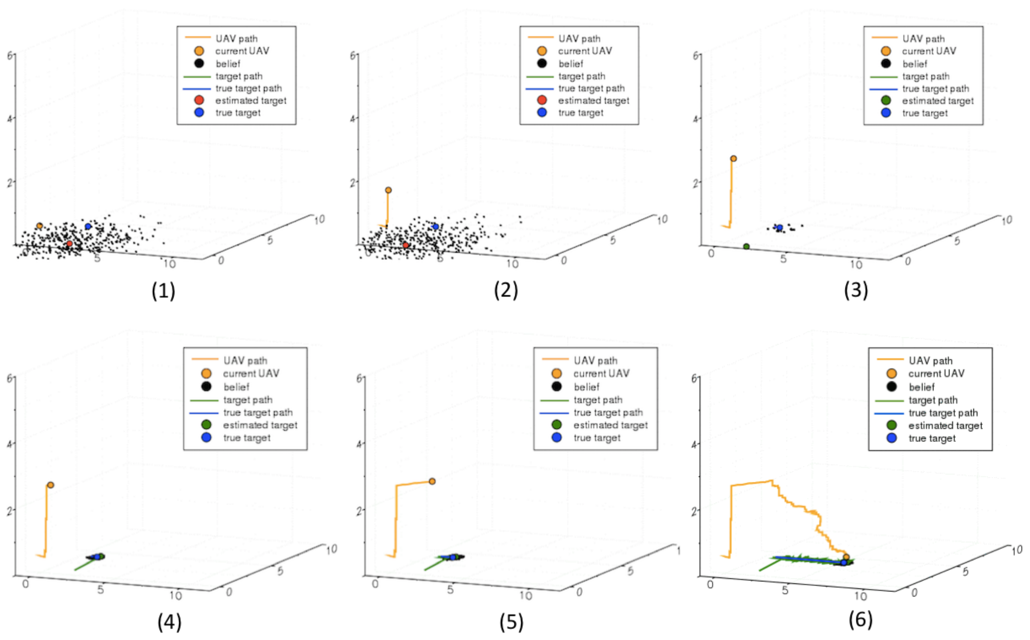

Figure 4: POMDP landing simulation snapshots with the UAV path in orange, the estimated target in green, the true target in blue, and the target position belief in black (x, y, z axes are in meters)

4 The POMDP Landing Problem

Instead of assuming the full knowledge of states as what is done in the MDP model, a POMDP model is constructed to handle the case when state uncertainty of a target platform is present. In this new scenario, the UAV state is assumed to be fully observed and the uncertainty only lies on the platform's state, which is estimated through intermittent camera measurements. Besides the necessary components described in a MDP model, a POMDP model takes observations belief states into account (i.e. POMDP=POMMDP(S, A, T, R, O, B)).

4.1 POMDP Formulation

The state, action spaces and the state transition are similar to what are defined for the MDP model. The state space includes the UAV pose in x, y, z coordinates, the UAV heading angle, and the target pose in x, y coordinates. Since we are controlling a UAV to land on a linearly moving ground platform whose dynamic model is known, the action space is only concerned with UAV actions, which are described by a forwarding speed, the speed in the z-direction, and a angle to head forward horizontally. The transition of the UAV state follows a dynamic model described by its state and action spaces.

Besides, four penalties are defined for the POMDP problem: the "out of scene" penalty (-100), the "action" penalty (-10), the "undetected" penalty (-800), and the "distance" penalty (-300*distance).

(1) The "out of scene" penalty sets constraints to discourage the UAV to fly below the plane level of the target height or above a certain height, like that of a ceiling;

(2) the "action" penalty encourages fewer steps of actions;

(3) the "undetected" penalty gives an extra burden when the UAV does not receive any measurement from the camera regarding the platform;

(4) the negative "distance" reward pushes the UAV to fly closer to the platform. All rewards are set to be negative values, namely, the penalties, so that landing with a zero reward will naturally be seen as the best option in a search tree algorithm.

The observation space is defined as a Boolean type variable indicating if any measurement of the platform is available and a 2-element vector describing the measured position of the platform. The relative position of the UAV and the platform determines whether or not the platform is inside of the FOV of the camera. A measurement is obtained only when the platform is within the camera FOV. In this scenario, a large "undetected" penalty pushes the UAV to develop "active sensing" behaviors to see the platform in the FOV. The belief of the platform is maintained as a Gaussian distribution and only gets updated when a measurement is received through a particle filter. In other words, when no measurement is observed, the platform state will be predicted by its known dynamic model but no updates.

A POMCPOW solver is used to handle this POMDP problem taking its capability for continuous states. A depth of 1000 and a tree query number of 1000 are set to render a robust solution. A terminal condition is reached when the UAV and the platform are close enough to each other.

4.2 Simulation and Results

A simulation[1] of the UAV attempting to chase the "target" and land on it is shown as snapshots in Fig.4. The motion of a UAV is simplified to have zero pitch and roll so that the camera FOV can be easily simulated as a square shape. The FOV angle is set to be 40 degrees. Initially, the confidence about the platform state is very low as presented by the black dots. At the same time, the estimated target shown as a red dot is randomly sampled from the belief distribution of the target platform, which is relatively far from the true target position, shown in blue. With a large penalty of not being able to "see" the platform, the UAV is forced to fly higher in order to cover the platform within its camera FOV. As soon as a measurement is obtained, the estimated target, now shown in green, quickly converges to the true target position while the belief space shrinks. After this point, the UAV never loses the view of the platform until landing on it. Therefore, it's shown that the estimated target dot moves with the true target dot with some oscillations inherently from the sensor and measurement noises.

5 Conclusion

In this project, we came up with detailed MDP and POMDP formulations to handle the problem of a UAV landing on a moving target. With the MDP model, the UAV achieves successful landings from 28 initial locations out of a total of 36 locations(i.e. $x \in [0,5], y \in [0,5], z = 10$). The success rate is around 77.78%. We observe that when the UAV starts from initial positions which are far from the target, the distance rewards are too small to be used for computing efficient actions. Therefore, the UAV hovers instead of taking efficient actions to chase the target as it does with large distance rewards. Besides initial UAV positions, target velocity and noise are two more important factors which would cause the failure of the UAV landing. If a target moves too fast or with a large noise (i.e large std), the UAV will not be able to land on the target as well. For the POMDP model, the UAV starts off from a predefined location and moves upward until the camera observes the moving target. Then the UAV starts to chase the target and achieves landing finally.

[1]https://github.com/biy001/UAV-autonomous-landing

Contributions

Xinyu Ren worked on MDP problem formulation, typically design dynamic model and landing reward, simulation as well as preliminary parameter tuning. Also written relevant paper part (S3) .

Bilan Yang took a full responsibility over the POMDP model, including developing, implementing, parameter tuning, simulating, visualizing the model as well as writing related part (S4).

C.Y. worked on MDP model optimization, tuning and visualizing the MDP model, and animation for testing models. Besides, written abstract, introduction, related work as well as conclusion and proofread the report.

References

[1] Nouara Achour and Karim Braikia. An mdp-based approach oriented optimal policy for path planning. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pages 205–210. IEEE, 2010.

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[3] Julien Burlet, Olivier Aycard, and Thierry Fraichard. Robust motion planning using markov decision processes and quadtree decomposition. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2820–2825. IEEE, 2004.

[4] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *AIIDE*, 2008.

[5] Guillaume Maurice Jean-Bernard Chaslot Chaslot. *Monte-carlo tree search*. PhD thesis, Maastricht University, 2010.

[6] M.J. Kochenderfer. *Decision Making Under Uncertainty: Theory and Application*. MIT Lincoln Laboratory series. MIT Press, 2015.

[7] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008.

[8] George E Monahan. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[9] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-Draa. Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[10] Zachary Sunberg and Mykel Kochenderfer. Pomcpow: An online algorithm for pomdps with continuous state, action, and observation spaces. *arXiv preprint arXiv:1709.06196*, 2017.

[11] Sebastian Thrun. Monte carlo pomdps. In *Advances in neural information processing systems*, pages 1064–1070, 2000.

[12] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.