Learning Swing-free Trajectories for UAVs with a Suspended Load

Aleksandra Faust[1], Ivana Palunko[2], Patricio Cruz[2], Rafael Fierro[2], and Lydia Tapia[1]

***Abstract*— Attaining autonomous flight is an important task in aerial robotics. Often flight trajectories are not only subject to unknown system dynamics, but also to specific task constraints. We are interested in producing a trajectory for an aerial robot with a suspended load that delivers the load to a destination in a swing-free fashion. This paper presents a motion planning framework for generating trajectories with minimal residual oscillations (swing-free) for rotorcraft carrying a suspended load. We rely on a finite-sampling, batch reinforcement learning algorithm to train the system for a particular load. We find the criteria that allow the trained agent to be transferred to a variety of models, state and action spaces and produce a number of different trajectories. Through a combination of simulations and experiments, we demonstrate that the inferred policy is robust to noise and to the unmodeled dynamics of the system. The contributions of this work are 1) applying reinforcement learning to solve the problem of finding a swing-free trajectory for a rotorcraft, 2) designing a problem-specific feature vector for value function approximation, 3) giving sufficient conditions that need to be met to allow successful learning transfer to different models, state and action spaces, and 4) verification of the resulting trajectories in simulation and to autonomously control quadrotors.**

I. Introduction

Unmanned aerial vehicles (UAVs) play an increasing role in a wide number of missions such as remote sensing, transportation, and search and rescue missions. Often, a critical part of a UAV's role is to carry loads vital to the mission. For example, cargoes may consist of food and supply delivery in disaster struck areas, patient transport, or spacecraft landing. Planning motions for a UAV with a load is complex because load swing is difficult to control. However, it is necessary for the safety and success of the mission.

Helicopters and quadrotors are ideal candidates for autonomous cargo delivery tasks because they are highly maneuverable, holonomic vehicles with the abilities of vertical takeoff and landing, and single-point hover. However, they are inherently unstable systems with complicated, nonlinear dynamics. Furthermore, the added suspended load changes the dynamics of the system.

Motions with minimal residual oscillations have applications in construction and manufacturing domains as well. They are desired for cranes on construction sites and loading docks [4], or for industrial robots carrying parts through the plants [18]. Swing-free trajectories for these systems are needed for safety concerns for the payload and the environment. Further, by not having to wait for the oscillation to naturally subside, swing-free trajectories improve the overall throughput of the system and increase the manufacturing capacity of the plant.

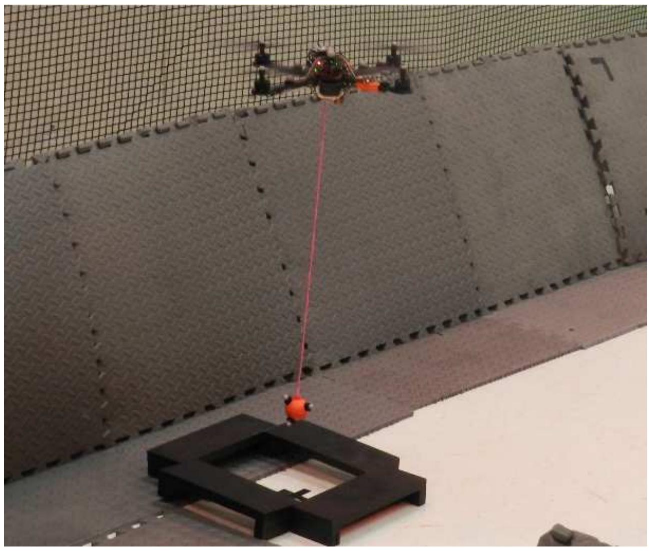


Fig. 1. Quadrotor with a suspended load

Our goal is to find a fast trajectory with minimal residual oscillations (swing-free) for a rotor craft aerial robot carrying a suspended load as described in [13]. In addition, swing control during the flight is desired. We assume that we know the goal state of the vehicle, and the initial state can be arbitrary. Furthermore, we assume that we have a *black box* simulator (or a generative model) available, but we make no assumptions about the dynamics of the systems while designing the algorithm.

We apply a machine learning approach to obtain a swing-free trajectory. We learn the task using an approximate value iteration (AVI) reinforcement learning algorithm. The value function is parametrized with problem-specific feature vectors. The learning and trajectory generation are separated in two distinct phases. In the first phase, we learn the value function approximation for a particular load. Once the value function is learned, we can use it to generate any number of trajectories. These trajectories can have different starting and ending positions and use different (but compatible) models (see Figure 2). We find the sufficient criteria to allow the transfer of the learned, inferred policy to a variety of situations. We demonstrate that the approach produces a swing-free trajectory to the desired state regardless of the starting position, robust to noise.

To verify our approach, we learn a value function approximation for swing-free flight using a generic holonomic

[1]Aleksandra Faust and Lydia Tapia are with the Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, {afaust, tapia}@cs.unm.edu

[2] Ivana Palunko, Patricio J. Cruz, and Rafael Fierro are with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, 87131-0001, {ipalunko, pcruzec, rfierro}@ece.unm.edu

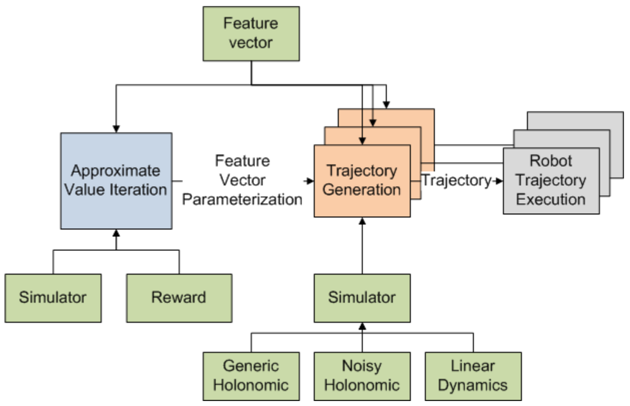


Fig. 2. Trajectory generation block diagram. The system learns problem-specific feature vector parametrization of the value function. It queries a simulator, calculates feature vectors and receives a reward for a state. Once learned, the value function approximation is passed to the trajectory generation to generate number of different trajectories. The module uses the same feature vectors, but can rely on different simulators to find the best action in any given state. The produced trajectory is sent to a robot. The green blocks are external to the learning algorithm and considered to be unknown.

model of the aerial vehicle with a suspended load as a simulator. Then we generate trajectories using two models: the same holonomic model used to learn parameters, and a noisy holonomic model. We demonstrate that the trajectories are feasible by using them for autonomous control of a Hummingbird quadrotor shown in Figure 1 to fly a single and multi-waypoint flight in a cluttered environment.

The contributions of this work are 1) applying reinforcement learning to solve the problem of finding a swing-free trajectory for a rotorcraft, 2) designing a rotorcraft UAV problem-specific feature vector for value function approximation, 3) giving sufficient conditions that need to be met to allow successful learning transfer to different models, state and action spaces, and 4) verification of the resulting trajectories in simulation and to autonomously control quadrotors.

II. Related Work

1) Quadrotor Trajectory Tracking: Schoellig et al. in [14] use an expectation-maximization learning algorithm to achieve quadrotor trajectory tracking. They start with a target trajectory and a simple linear model. Lupashin et al. [10] apply policy gradient descent techniques to perform aggressive quadrotor multi-flips that improve over repeated iterations. They improve upon it in [9] by segmenting the trajectory into keyframes and learning the parameters for each segment separately.

2) Quadrotor Swing-free Trajectory Creation: Palunko et al. successfully applied dynamic programming to solve swing-free trajectories for quadrotors [13] and [12]. However, dynamic programming requires that the dynamics of the system are known ahead of time, and is sensitive to the accuracy of the model, and the start and goal states. A machine learning approach doesn't require the white box approach to system's dynamics, and learning doesn't need to be repeated when the start state changes. Further, the reinforcement learning approach is more suitable for compensating for the accumulated error resulting from model approximation. Lastly, while dynamic programming requires pre-calculating each trajectory, the approach presented here allows us to learn the problem once, and generate any number of different trajectories with different starting positions using same value function approximation.

3) Swing-free Trajectories in Manufacturing: Swing-free trajectories have been studied outside of the UAV domain. They are important in industrial robotics with applications such as cranes in construction sites and for cargo loading in ports [4], [18]. Residual oscillation reduction has applications in manufacturing environments where parts need to be transported in a limited space. Zameroski et al. [19] applied dynamic programming to reduce residual vibrations of a freely suspended payload.

4) Reinforcement and Transfer Learning: To accomplish swing-free trajectories for rotorcraft with a suspended load, we rely on approximate value iteration [5], [7], [16] with a specifically designed feature vector for value function approximation. Taylor and Stone [17] propose value function transfer between the tasks in different state and action spaces using behavior transfer function to transfer the value function to the new domain. In this work, we transfer the learned value function to tasks with state and action space supersets and changed dynamics, and find sufficient characteristics of the target tasks for the learning transfer to occur successfully. We directly transfer the value function, and perform no further learning. Sherstov and Stone in [15] examine action transfer between the tasks, learning the optimal policy and transferring only the most relevant actions for the optimal policy. We take the opposite approach. To save computational time, we learn sub-optimal policy on a subset of actions, and transfer it to the expanded action space to produce a more refined plan.

McMahan et al. [11] suggested learning a partial policy for fixed start and goal states. Such a partial policy manages state space complexity by focusing on states are that more likely to be encountered. We are interested in finding swing-free trajectories from different start states, but we do have a single goal state. Thus, all trajectories will pass near the goal state, and we learn the partial policy only in the vicinity of the goal state. Then, we apply it for any start state.

III. Methods

A. Reinforcement Learning for Swing-Free Trajectories

The approximate value iteration algorithm produces an approximate solution to a Markov Decision Process in continuous state spaces with a discrete action set. We approximate the value function with a linearly parametrized feature vector. It is in an expectation-maximization (EM) algorithm which relies on a sampling of the state space transitions, an estimation of the feature vector parameters, and a linear regression to find the parameters that minimize the least square error.

AVI does not directly depend on the time step size. Similarly, the algorithm is agnostic to the time it takes to reach the goal state and to any ordering there might be in the state space chain. It randomly explores the state space and learns a function that assigns a scalar representing a quality of a state. Its running time is $O(n \cdot iterations \cdot \|actions\|)$, where

n is number of samples, $iterations$ number of iterations to perform, and $actions$ is the size of the discretized action space.

In our implementation, the state space is a 10-dimensional vector s of the vehicle's position $p = (x, y, z)$, linear velocity $v = (\dot{x}, \dot{y}, \dot{z})^T$, load displacement angles $\eta_L = (\phi_L, \theta_L)^T$ and their respective angular speeds $\dot{\eta_L} = (\dot{\phi_L}, \dot{\theta_L})^T$, where L is the length of the suspension cable (see Figure 3).

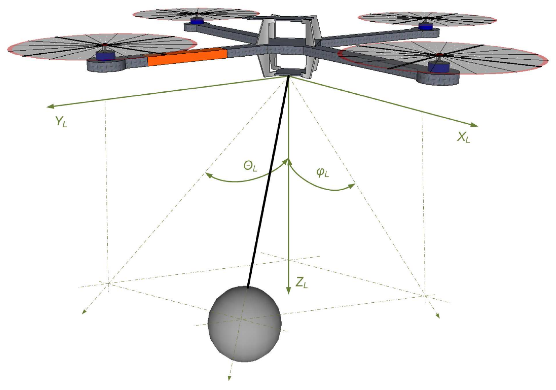


Fig. 3. Load displacement angles for a quadrotor carrying a suspended load.

The samples are uniformly, randomly drawn from a hypercube centred in the goal state at equilibrium. The action space is a linear acceleration vector $a = [\ddot{x} \quad \ddot{y} \quad \ddot{z}]^T$ discretized using equidistant steps centered around zero acceleration.

The value function V is approximated with a linear combination of the feature vector $F(s)$. The feature vector chosen for this problem consists of four basis functions: squares of position, velocity magnitude, load distance and load velocity magnitude relative to the goal state as shown in (1):

$$\begin{aligned} V_n(s) &= \psi_n^T * F(s) \\ F(s) &= [\|p\|^2 \quad \|(v)\|^2 \quad \|\eta\|^2 \quad \|\dot{\eta}\|^2]^T \end{aligned} \tag{1}$$

where $\psi \in \mathbb{R}^4$.

The reward function penalizes the distance from the goal state, and the size of the load swing. It also penalizes the negative z coordinate to provide a bounding box and enforce that the vehicle must stay above the ground. Lastly, the agent is rewarded when it reaches equilibrium. The reward function $R(s) = c^T r(s)$ is a linear combination of basis rewards $r(s) = [r_1(s) \quad r_2(s) \quad r_3(s)]^T$, weighted with vector $c = [c_1 \quad c_2 \quad c_3]^T$, where:

$$r_1(s) = -\|p\|^2$$

$$r_2(s) = \begin{cases} a_1 & \|F(s)\| < \epsilon \\ -\|\eta\|^2 & \text{otherwise} \end{cases}$$

$$r_3(s) = \begin{cases} -a_2 & z < 0 \\ 0 & z \geq 0 \end{cases}$$

To obtain the state transition function samples, we rely on a simplified model of the quadrotor-load system, where the quadrotor is represented by a holonomic model of a UAV widely used in the literature. Equations (2) and (3) describe the simulator. $g' = [0 \quad 0 \quad g]^T$ is gravity force, l is the length of the sling, and τ is the length of the time step.

$$\begin{aligned} v_{n+1} &= v_n + \tau a; \quad p_{n+1} = p_n + \tau v_n + 0.5\tau^2 a \\ \dot{\eta}_{n+1} &= \dot{\eta}_n + \tau\ddot{\eta}; \quad \eta_{n+1} = \eta_n + \tau\dot{\eta}_n + 0.5\tau^2\ddot{\eta} \end{aligned} \tag{2}$$

where

$$\ddot{\eta} = \begin{bmatrix} \sin\theta_n \sin\phi_n & -\cos\phi_n & \cos\theta_n \sin\phi_n l^{-1} \\ -\cos\theta_n \cos\phi_n & 0 & \cos\phi_n \sin\theta_n l^{-1} \end{bmatrix} (a - g) \tag{3}$$

B. Trajectory Generation

An approximated value function induces a greedy policy that is used to generate the trajectory and control the vehicle. It is determined by $a = argmax_a(\psi^T F(P(state, a)))$, where P is the state transition function. The algorithm starts with the initial state. Then it finds an action that produces the highest return using an approximated value function. That action is used to transition to the next state. The algorithm stops a when the goal is reached or when the trajectory exceeds a maximum number of steps. Trajectory generation does not refine the policy with the new information. However, it still can adapt and find its way to the goal state even in the presence of noise, as we will see later. The trajectory generation running time is $O(max\_steps \cdot \|actions\|)$.

It is important to note that both AVI and the trajectory generation are not suitable to be executed on real hardware, and are strictly simulation algorithms. In the case of AVI, this is because the random sampling is infeasible on hardware. One could adapt the algorithm not to do a random sample, but rather to observe an actual flight if possible, similarly to how Abbeel et al. approached learning initial helicopter dynamics [3].

C. Analysis

The value function approximation does not necessarily need to be numerically close to the true value function. The Proposition III.1 gives sufficient conditions that the value function approximation, action state space and system dynamics need to meet to guarantee a plan that leads to the goal state.

Proposition III.1. *Let $s_0 \in S$ be a desired goal state in the planning problem described by MDP (S, A, P, R). If a function $V : S \to \mathbb{R}$ has a unique maximum in $s_0 \in S$, and action space A is such that $\forall s \in S \setminus \{s_0\}, \exists a \in A$ such that $V(\pi_A(s)) > \epsilon + V(s)$, for some $\epsilon > 0$, then for an arbitrary start state $s \in S$, greedy policy with respect to V leads to the goal state s_0. In other words, $\forall s \in S, \exists n, \pi_A^n(s) = s_0$.*

See Appendix for proof.

Proposition A.1 shows that all ψ_i need to be negative for the value function V described in (1) to have a unique maximum. As we will see in the IV-B, the empirical results show that is the case. These observations lead to several practical properties of the induced greedy policy that we will verify empirically:

1) The induced greedy policy is robust to some noise: as long as there is a transition to a state with a higher value, an action could be taken and the goal will be attained, although not optimally. Section IV-B presents the empirical evidence for this property.

2) The policy is agnostic to the simulator used: The simulator defines the transition function and along with the action space defines the set of reachable states. Thus, as long as the conditions of Proposition III.1 are met, we can switch the simulators we use. This means that we can train on a simple simulator and generate a trajectory on a more sophisticated model that would predict the system better.

3) Learning on the domain subset: As we will show experimentally in Sections IV-B and IV-C, we can learn the model on a small subset of the state space around the goal state, and the resulting policy will work on the whole domain where the criteria above hold, i.e., where the value function doesn't have other maxima. This property makes the method a good choice for a local planner.

4) Changing action space: Lastly, the action space between learning and the trajectory generation can change, and the algorithm will still produce a trajectory to the goal state. For example, to save computational time, we can learn on the smaller, more coarse discretization of the action space to obtain the value function parameters, and generate a trajectory on a more refined action space which produces a smoother trajectory. We will demonstrate this property during the multi-waypoint flight experiment.

Since we are using an approximation to represent a value function and obtain an estimate iteratively, the question of algorithm convergence is twofold. First, the parameters that determine the value function must converge to a fixed point. Second, the fixed point of the approximator must be close to the true value function.

Convergence of the algorithm is not guaranteed in the general case. Convergence is guaranteed if the value function is a contraction [6]. In our case, the approximator function is not a contraction. Thus, we will show empirically that the approximator parameters stabilize. To show that the policy derived from a stabilized approximator is sound, we will examine the resulting trajectory. The trajectory needs to be swing-free at the arrival at the goal state, and be suitable for the system.

IV. Results

In this section we verify the convergence of the proposed algorithm as well as its effectiveness in simulation and experiment. Section IV-A assesses the approximate value iteration convergence. Section IV-B shows the results of trajectory generation in simulation for the expanded state and action space. Lastly, Section IV-C presents results of experiments with the quadrotor in expanded state and action space. The experiments assess the discrepancy between the simulation swing predictions and the actual swing encountered during the flight, and make a comparison between a cubic trajectory (trajectory where position is a 3^{rd} order polynomial function of time) and our method.

TABLE I
Approximate value iteration algorithm hyperparameters.

Parameter	3D Configuration	2D Configuration
γ	0.9	0.9
Min action	(-3, -3, -3)	(-3, -3, 0)
Max action	(3, 3, 3)	(3, 3, 0)
Action step	0.5	0.05
Min sampling space	$p = (-1, -1, -1)$, $v = (-3, -3, -3)$ $\eta = (-10°, -10°)$, $\dot{\eta} = (-10, -10)$	
MAX sampling space	$p = (1, 1, 1)$, $v = (3, 3, 3)$ $\eta = (10°, 10°)$, $\dot{\eta} = (10, 10)$	
Sampling	Linear	Constant (200)
Simulator	Holonomic	
Frequency	50Hz	
Number of iterations	1000	800
Number of trials	100	40
Reward function	$c_1 = 10000$, $c2 = 750$, $c_3 = 1$ $a_1 = 14$, $a_2 = 10000$, $\epsilon = 0.05$	

A. Value Function Approximation Learning Results

We run AVI in two configurations: 2D and 3D (see table IV-A). Both configurations use the same discount parameter $\gamma < 1$ to ensure that the value function is finite. The configurations also share the simulator, described in (2) and (3).

The 3D configuration trains the agent with a coarse three-dimensional action vector. Each direction of the linear acceleration is discretized in 13 steps, resulting in 13^3 total actions. In this phase of the algorithm we are shaping the value function, and this level of coarseness is sufficient.

Farahmand et al. in [8] showed that AVI's approximation error decays exponentially with the number of iterations, and that gradually increasing the sampling with iterations yields less error as the number of iterations increases. Thus, we increase sampling linearly with the number of iterations in the 3D configuration.

To assess the stability of the approximate value iteration, we ran that AVI 100 times, for 1,000 iterations in the 3D configuration. Figure 4 shows the trend of the norm of value parameter vector ψ with respect to L_2 norm. We can see that the $\|\psi\|$ stabilizes after about 200 iterations with the mean of $3.6117e + 05$. The empirical results show that the algorithm is stable and produces a consistent policy over different trials. A mean value of $\psi = (-86, 290, -350, 350, -1, 430, -1, 160)^T$, which means that according to A.1, the value function approximation has a global maximum.

Figure 5 depicts trajectories with the start state in (-2, -2, 1) over 100 trials. Although there are slight variations in duration, all the trajectories are similar in shape and are consistent, giving us confidence that the AVI converges to the optimal value. The load initially lags behind as the vehicle accelerates, but then stabilizes to end in a minimal swing. We can also see that the swing is controlled throughout the trajectory, maintaining the swing under $10°$ for the duration of the entire flight.

The 2D configuration uses a finer discretization of the action space, although only in the x and y directions. There are 121 actions in each direction, totalling to 121^2 actions in the discretized space. We will use this configuration in the experiments on the quadrotor. This configuration uses a fixed sampling methodology. Results in [8] show that the approximation error stabilizes to a roughly constant level after the parameters stabilize.

B. Simulation Results

We access the quality and robustness of a trained agent in simulation by generating trajectories from different distances for two different simulators. The first simulator is a generic holonomic aerial vehicle with suspended load simulator, the same simulator we used in the learning phase. The second simulator is a noisy holonomic aerial vehicle simulator, which adds up to 5% uniform noise to the predicted state. Its intent is to simulate the inaccuracies and uncertainties of the real hardware.

We compare the performance of our learned generated trajectories with model-based dynamic programming (DP) and cubic trajectories. The cubic and DP trajectories are generated as described in [13] using the the dynamics model in (2) and (3) and are of the same duration as corresponding learned trajectories.

The agent is trained in 3D configuration (see Table IV-A). For trajectory generation, we use a fine-grain discretized 3D action space $A = (-3 : 0.05 : 3)^3$. This action space is ten times per dimension finer and contains 121^3 different actions. The trajectories were generated at 50Hz with a maximum duration of 15 seconds. All the trajectories were generated and averaged over 100 trials.

To assess how well a policy adapts to different starting positions, we choose two different fixed positions, (-2,-2, 1) and (-20,-20,15), and two variable positions. The variable positions are randomly drawn from between 4 and 5 meters, and within 1 meter from the goal state. The last position measures how well the agent performs within the sampling box. The rest of the positions are well outside of the sampling space used for the policy generation, and assess how well the method works for trajectories outside of the sampling bounds with an extended state space.

Table IV-B presents the averaged results with their standard deviations. We measure the end state and the time when the agent reaches the goal, the percentage of trajectories that reach the goal state within 15 seconds, and the maximum swing experienced among all 100 trails. With the exception of the noisy holonomic simulator at the starting position (-20,-20,15), all experiments complete the trajectory within 4 cm of the goal, with a swing of less than $0.6°$. The trajectories using the noisy simulator from a distance of 32 meters (-20,-20,15) don't reach within 5 cm because 11% of the trajectories exceed the 15-second time limit before the agent reaches its destination. However, we still see that the swing is controlled and minimal at the destination approach even in that case.

The results show that trajectories generated under noisy conditions take a bit longer to reach the goal state, and the standard deviation associated with the results is a bit larger. This is expected, given the random nature of the noise. However, all of the noisy trajectories reach the goal with about the same accuracy as the non-noisy trajectories. This finding matches our prediction from Section III.

The maximum angle of the load during its entire trajectory for all 100 trials inversely depends on the distance from the initial state to the goal state. For short trajectories within the sampling box, the swing always remains within $4°$, while for the very long trajectories it could go up to $46°$. As seen in Figure 5, the peak angle is reached at the beginning of the trajectory during the initial acceleration, and as the trajectory proceeds, the swing reduces. This makes sense, given that the agent is minimizing the combination of the swing and

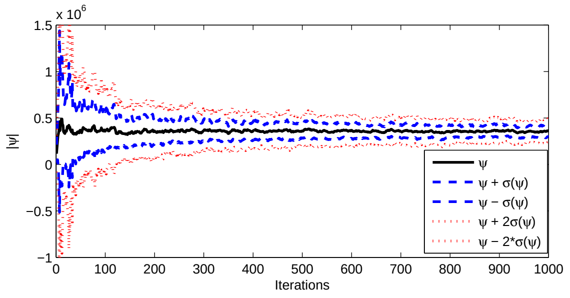
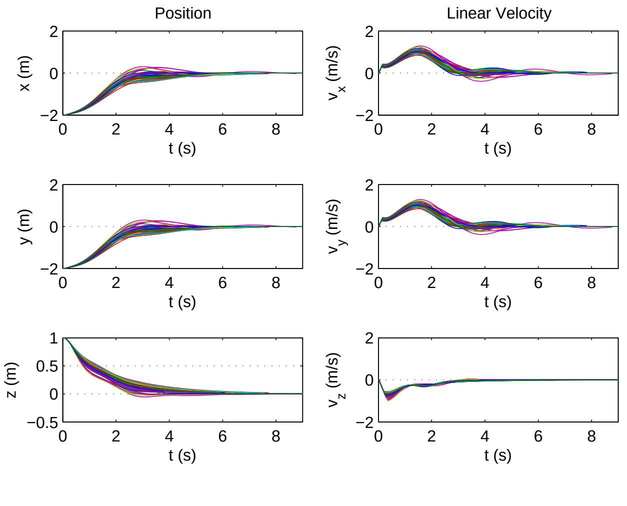
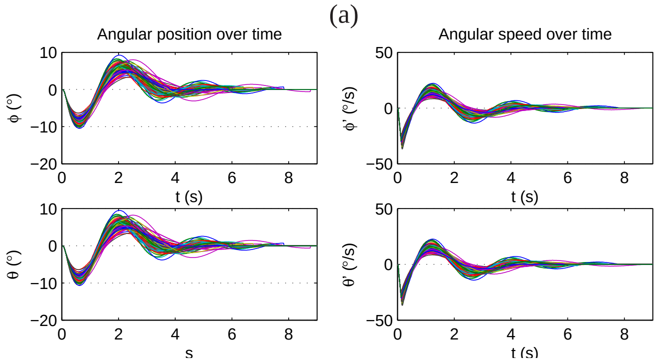


Fig. 4. Convergence of feature parameter vector ψ's norm over 1000 iterations. The results are averaged over 100 trials. One and two standard deviations are shown. After initial learning phase, ψ stabilizes to a constant value.



(a)

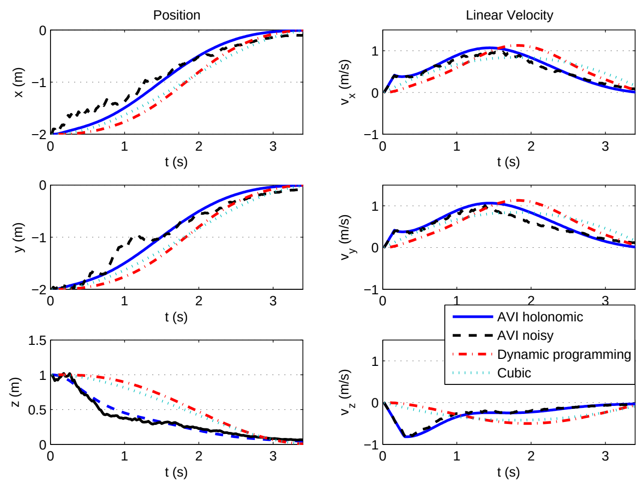


(b)

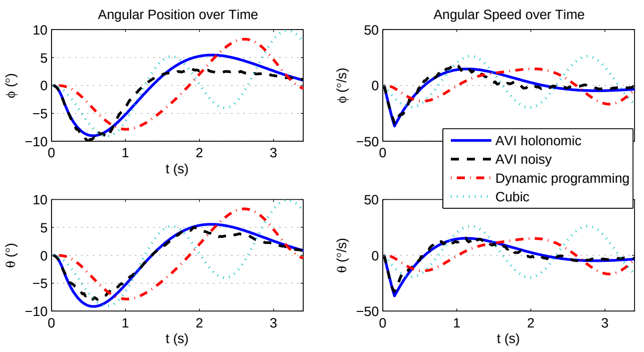
Fig. 5. Trajectories starting at (-2, -2, 1) for each of the 100 trials of the (a) vehicle and (b) its load using 3D configuration for training and holonomic simulator with fine-grain action space for trajectory generation.

distance. When very far away from the goal, the agent will move quickly towards the goal state and produce increased swing. Once the agent is closer to the goal state, the swing component becomes dominant in the value function, and the swing reduces.

Figure 6 shows the comparison of the trajectories with the same starting position (-2, -2, 1) and same ψ parameter, generated using the models above (AVI trajectories) compared to cubic and DP trajectories. First, we see that the AVI trajectories share a similar velocity profile (Figure 6 (a)) with two velocity peaks, both occurring in the first half of the flight. Velocities in DP and cubic trajectories have a single maximum in the second half of the trajectory. The resulting swing predictions (Figure 6 (b)) show that in the last 0.3 seconds of the trajectory, the cubic trajectory a exhibits swing of 10°, while the DP trajectory ends with a swing of less than 5°. Our trajectories are within 2° in the same time period.



(a)



(b)

Fig. 6. Trajectories of the (a) vehicle and (b) its load where the training was performed in 3D configuration and the trajectories were generated using generic and noisy holonomic simulators compared to the cubic and dynamic programming trajectories of the same duration.

C. Experimental Results

1) Setup: The experiments were performed using the MARHES multi-aerial vehicle testbed. This testbed and its real-time controller are described in detail in [12]. We first trained an agent in 2D configuration (see Table IV-A). Once the agent was trained, we generated trajectories for two experiments: flight with a single waypoint, and with a multi-waypoint flight.

To generate trajectories, we used a fine-grain discretized 3D action space $A = (-3 : 0.05 : 3)^3$. The trajectories were generated at 50Hz using the generic holonomic aerial vehicle carrying a suspended load simulator, the same simulator that was used in the learning phase.

These trajectories were sent to the quadrotor with a suspended load weighing 45 grams on a 62 cm-long suspension cable. The vehicle and load positioning was tracked by the Vicon Positioning System [2].

2) Single-waypoint experiment: In flight with a single waypoint, the quadrotor flew from (-1,-1,1) to (1,1,1). Figure 9 compares the vehicle and load trajectories for a learned trajectory as flown and in simulation, with cubic and DP trajectories of the same length and duration. The vehicle trajectories in Figure 9 (a) suggest a difference in the velocity profile, with the learned trajectory producing a slightly steeper acceleration between 1 and 2.5 seconds. The learned trajectory also contains a 10 cm vertical move up toward the end of the flight.

Comparison with Simulation: Looking at the load trajectories in Figure 9 (b), we notice the reduced swing, especially in the second half of the load's ϕ coordinate. The trajectory in simulation never exceeds 10°, and the actual flown trajectory reaches the maximum at 12°. Both learned load trajectories follow the same profile with two distinct peaks around 0.5 seconds and 2.2 seconds into the flight, followed by a rapid swing control and reduction to under 5°. The actual flown trajectory naturally contains more oscillations that the simulator didn't model for. Despite that, the limits, boundaries, and the profile of the load trajectory are close between the simulation and the flown trajectory. This verifies the validity of the simulation results: the load trajectory predictions in the simulator are reasonably accurate.

Comparison with Cubic: Comparing the flown learned trajectory with a cubic trajectory, we see a different swing profile. The cubic load trajectory has higher oscillation, four peaks within 3.5 seconds of flight, compared to three peaks for the learned trajectory. The maximum peak of the cubic trajectory is 14° at the beginning of the flight. The most notable difference happens after the destination is reached during the hover (after 3.5 seconds in Figure 9 (b)). In this part of the trajectory, the cubic trajectory shows a load swing of $5-12^\circ$, while the learned trajectory controls the swing to under 4°.

Comparison with DP: Figure 9 (b) shows that load for the trajectory learned with reinforcement learning stays within the load trajectory generated using dynamic programming at all times: during the flight (the first 3.4 seconds) and the residual oscillation after the flight.

3) Multi-waypoint experiment: In the second set of experiments, the same agent was used to generate multiple trajectories to perform a multi-waypoint flight and demonstrate the ability to perform a more complex flight pattern in a cluttered environment based on a single learning. The flight consists of

TABLE II

SUMMARY OF TRAJECTORY RESULTS FOR DIFFERENT STARTING POSITION AVERAGED OVER 100 TRIALS: PERCENT COMPLETED TRAJECTORIES WITHIN 15 SECONDS, TIME TO REACH THE GOAL, FINAL DISTANCE TO GOAL, FINAL SWING, AND MAXIMUM SWING.

State		Goal reached	t (s)		$\parallel p \parallel (m)$		$\parallel \eta \parallel (^\circ)$		$max \parallel \eta \parallel (^\circ)$	
Location	Simulator	(%)	μ	σ	μ	σ	μ	σ	μ	σ
(-2,-2,1)	Generic Holonomic	100	6.13	0.82	0.03	0.01	0.54	0.28	12.19	1.16
	Noisy Holonomic	100	6.39	0.98	0.04	0.01	0.55	0.30	12.66	1.89
(-20,-20,15)	Generic Holonomic	99	10.94	1.15	0.04	0.01	0.49	0.33	46.28	3.90
	Noisy Holonomic	89	12.04	1.91	0.08	0.22	0.47	0.45	44.39	7.22
((4,5),(4,5),(4,5))	Generic Holonomic	100	7.89	0.87	0.04	0.01	0.36	0.31	26.51	2.84
	Noisy Holonomic	100	7.96	1.11	0.04	0.01	0.44	0.29	27.70	3.94
((-1,1),(-1,1),(-1,1))	Generic Holonomic	100	4.55	0.89	0.04	0.01	0.33	0.30	3.36	1.39
	Noisy Holonomic	100	4.55	1.03	0.04	0.01	0.38	0.29	3.46	1.52

nine segments, covering different altitudes, see Figure 7. The clutter placed in the environment (Figure 8) deflects the rotor wind and affects the load swing. The high-resolution flight video is available at [1]. Note that the trajectories generated for this experiment used value approximator parameters that were learned on a 2D action space, in the xy plane, and produced a viable trajectory that changes altitude.

The experiment was performed three times and the resulting quadrotor and load trajectories are depicted in Figure 10. During the first 10 seconds of the flight, the quadrotor hovers and the swing is within 5°. At the very beginning of the flight (seconds 10 to 20), the swing is maximal, staying within 15°. In the last phase of the flight, the swing reduces to within 10°, and at the very end of the trajectory (seconds 40 to 45) it reduces to the nominal swing within 5° although the aircraft is still moving.

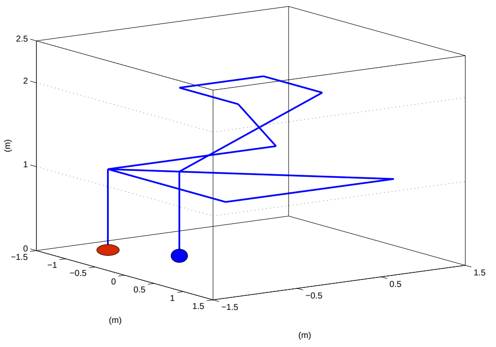


Fig. 7. Trajectory of the multi-waypoint flight.

V. CONCLUSIONS

In this work, we presented a motion planning framework for producing trajectories with minimal residual oscillations for a rotorcraft UAV with a freely suspended load. The framework relies on reinforcement learning to learn the problem characteristics for a particular load. We found conditions that if met allow the learned agent to be applied to produce a wide variety of trajectories. We discussed the learning convergence, assessed the produced motion plans in simulation, and their robustness to noise. Lastly, we implemented the proposed algorithm on a quadrotor type UAV in order to demonstrate its feasibility and to assess the accuracy of the simulation results.

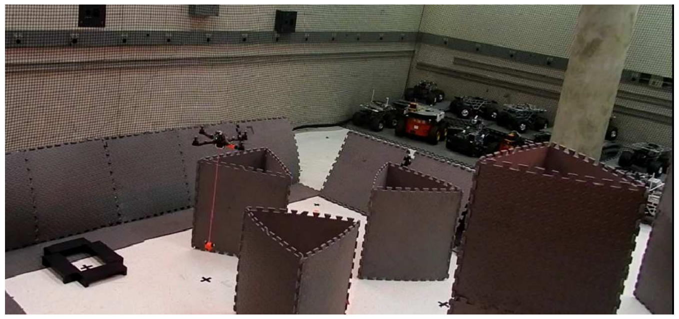


Fig. 8. Quadrotor completing the multi-waypoint flight in the cluttered environment.

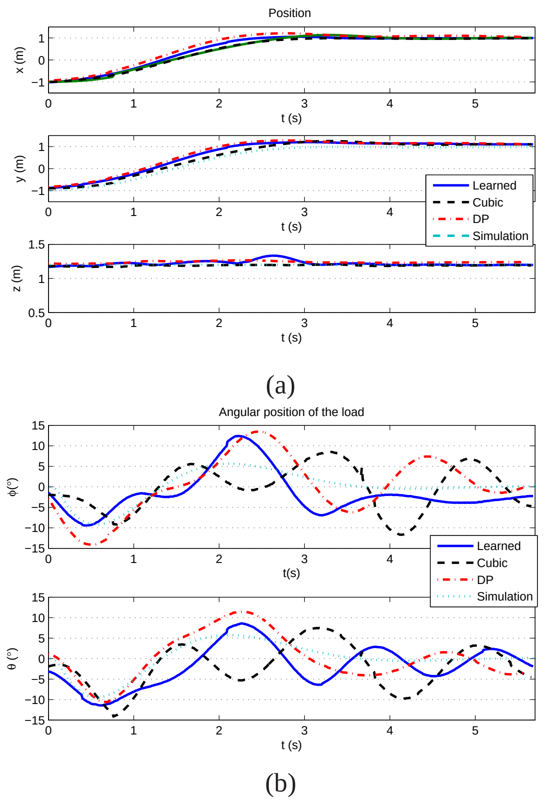


Fig. 9. Quadrotor (a) and load (b) trajectories as flown, created through learning compared to cubic, dynamic programming, and simulated trajectories.

Acknowledgements

This work was in part supported by NSF grant ECCS #1027775 and the Department of Energy URPR Grant #DE-FG52-04NA25590. Tapia is supported in part by the National Institutes of Health (NIH) Grant P20RR018754 to the Center for Evolutionary and Theoretical Immunology.

References

[1] Adaptive motion planning research group - adaptive quadrotor control project. http://cs.unm.edu/amprg/Research/Quadrotor/.

[2] Vicon Motion Systems Limited. http://www.vicon.com/.

[3] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *International Journal of Robotics Research (IJRR)*, 29, Nov. 2010.

[4] M. Agostini, G. Parker, H. Schaub, K. Groom, and I. Robinett, R.D. Generating swing-suppressed maneuvers for crane systems with rate saturation. *Control Systems Technology, IEEE Transactions on*, 11(4):471 – 481, July 2003.

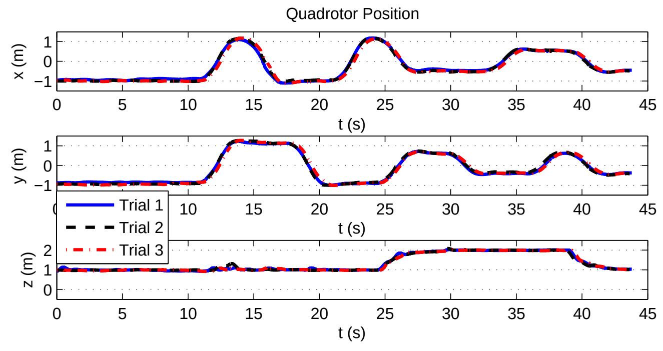
[5] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[6] L. Busoniu, R. Babuska, B. D. Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2010.

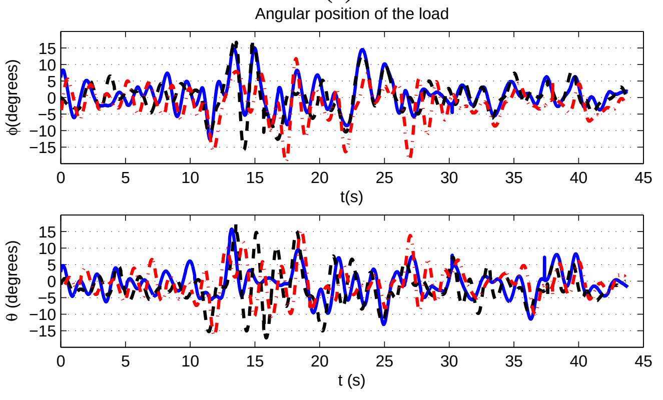
[7] D. Ernst, M. Glavic, P. Geurts, and L. Wehenkel. Approximate value iteration in the reinforcement learning context. application to electrical power system control, 2005. Download from: http://www.bepress.com/ijeeps/vol3/iss1/art1066.

[8] A. M. Farahmand, R. Munos, and C. Szepesvári. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, 2010.

[9] S. Lupashin and R. DAndrea. Adaptive open-loop aerobatic maneuvers for quadrocopters. In *World Congress*, volume 18, pages 2600–2606, 2011.



(a)



(b)

Fig. 10. Quadrotor (a) and load (b) trajectories as flown, over three trials in the multi-waypoint test.

[10] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *IEEE International Conference on Robotics and Automation*, pages 1642–1648, May 2010.

[11] H. B. Mcmahan, M. Likhachev, and G. J. Gordon. Bounded real-time dynamic programming: Rtdp with monotone upper bounds and performance guarantees. In *In ICML05*, pages 569–576, 2005.

[12] I. Palunko, P. Cruz, and R. Fierro. Agile load transportation : Safe and efficient load manipulation with aerial robots. *Robotics Automation Magazine, IEEE*, 19(3):69 –79, sept. 2012.

[13] I. Palunko, R. Fierro, and P. Cruz. Trajectory generation for swing-free maneuvers of a quadrotor with suspended payload: A dynamic programming approach. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2691 –2697, May 2012.

[14] A. Schoellig, F. Mueller, and R. DAndrea. Optimization-based iterative learning for precise quadrocopter trajectory tracking. *Autonomous Robots*, 33:103–127, 2012.

[15] A. A. Sherstov and P. Stone. Improving action selection in MDP's via knowledge transfer. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, July 2005.

[16] R. Sutton and A. Barto. *A Reinforcement Learning: an Introduction*. MIT Press, MIT, 1998.

[17] M. E. Taylor and P. Stone. Behavior transfer for value-function-based reinforcement learning. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. P. Singh, and M. Wooldridge, editors, *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 53–59, New York, NY, July 2005. ACM Press.

[18] J. Yang and S. Shen. Novel approach for adaptive tracking control of a 3-d overhead crane system. *Journal of Intelligent and Robotic Systems*, 62:59–80, 2011. 10.1007/s10846-010-9440-9.

[19] D. Zameroski, G. Starr, J. Wood, and R. Lumia. Rapid swing-free transport of nonlinear payloads using dynamic programming. *ASME Journal of Dynamic Systems, Measurement, and Control*, 130(4), July 2008.

Appendix

Proof of Proposition III.1

Proof. For simplicity, let's denote with $V_n = V(\pi_A^n(s))$, the value of the n^{th} step when applying the greedy policy π_A with respect to the action space A.

Let $V_0 = V(s_0)$, be the single maximum of function V. Then since V_0 is finite there $\exists k \in N$ s.t. $V_1 + (k-1)\epsilon \le V_0 < V_1 + k\epsilon$, where $V_1 = V(s)$ is the value of the initial state.

Notice that because $V(\pi_A(s)) > \epsilon + V(s) \Rightarrow V_n > (n-1)\epsilon + V_1$. Thus, for an arbitrary step n: $\|V_0 - V_n\| \le \|V_0 - (n-1)\epsilon - V_1\| \le \|k\epsilon - (n-1)\epsilon\| = \|(k-n+1)\epsilon\|$.

For $n_0 = k+1 \Rightarrow V_0 = V_{n_0}$.

□

Proposition A.1. V has a single maximum - *Value function approximation $V(s) = \psi^T * F(s)$, $F(s) = (\|p\|^2, \|(v)\|^2, \|\eta\|^2, \|\dot{\eta}\|^2)^T$, has a single extrema if all components of the vector ψ are of the same sign. If all components of ψ are negative, the extrema is maximum.*

Proof. By solving $\frac{dV}{ds} = 0$,and ensuring that, $\frac{d^2V}{ds^2}$ is negative definite. $\frac{dV}{ds} = 2\begin{bmatrix}\psi_1 p & \psi_2 v & \psi_3 \eta & \psi_4 \dot{\eta}\end{bmatrix}^T$. Thus,

$$\frac{dV}{ds} = 0 \Leftrightarrow \|\psi\| \neq 0 \wedge s_0 = [0,0,0,0,0,0,0,0,0,0]^T. \tag{A.4}$$

$\frac{d^2V}{ds^2} = 2\begin{bmatrix}\psi_1 & 0 & 0 & 0\\ 0 & \psi_2 & 0 & 0\\ 0 & 0 & \psi_3 & 0\\ 0 & 0 & 0 & \psi_4\end{bmatrix}$. s_0 is a unique extrema iff $\psi_i \neq 0$, and $\frac{d^2V}{ds^2}$ is a negative definite matrix only if all $\psi_i \le 0$. Learning in both Configurations described in IV-A, resulted in negative psi vector for all 100 trials.

□