Reinforcement Learning for Balancing a Flying Inverted Pendulum*

Rafael Figueroa[1], Aleksandra Faust[2], Patricio Cruz[1], Lydia Tapia[2], and Rafael Fierro[1]
[1]*Department of Electrical and Computer Engineering*, [2]*Department of Computer Science*
University of New Mexico
Albuquerque, NM 87131, United States
[1]{*rfigueroa, pcruzec, rfierro*}*@unm.edu*, [2]{*afaust, tapia*}*@cs.unm.edu*

***Abstract*— The problem of balancing an inverted pendulum on an unmanned aerial vehicle (UAV) has been achieved using linear and nonlinear control approaches. However, to the best of our knowledge, this problem has not been solved using learning methods. On the other hand, the classical inverted pendulum is a common benchmark problem to evaluate learning techniques. In this paper we demonstrate a novel solution to the inverted pendulum problem extended to UAVs, specifically quadrotors. This complex system is underactuated and sensitive to small acceleration changes of the quadrotor. The solution is provided by reinforcement learning (RL), a platform commonly applied to solve nonlinear control problems. We generate a control policy to balance the pendulum using Continuous Action Fitted Value Iteration (CAFVI) [1] which is a RL algorithm for high-dimensional input-spaces. This technique combines learning of both state and state-action value functions in an approximate value iteration setting with continuous inputs. Simulations verify the performance of the generated control policy for varying initial conditions. The results show the control policy is computationally fast enough to be appropriate of real-time control.**

***Index Terms*— Aerial robotics, quadrotor control, inverted pendulum, approximate value iteration, reinforcement learning.**

I. Introduction

A pendulum on top of a quadrotor, Fig. 1, requires controls to balance the pendulum and to stabilize the aerial vehicle. Solving this complex problem offers insight into advanced control strategies that can be considered for similar aerial manipulation tasks. The flying inverted pendulum was first introduced in [2] where it was solved designing linear controllers for stabilization. However, the results indicated that a learning approach could improve the system performance [2].

Reinforcement learning (RL) has grown as an effective framework for control applications in recent years, due to its ability to learn from available data rather than fully-understood system models. For example, RL has been used to solve the classical inverted pendulum control problem [3], [4]. Also, iterative learning has improved quadrotor flips [5].

We previously employed RL for suspended load delivery with a quadrotor [7], [8], [1]. Combining motion planning with reinforcement learning, we generated collision-free trajectories with minimal residual oscillations [9].

Our method for balancing a flying inverted pendulum is based on a RL technique, Continuous Action Fitted Value Iteration (CAFVI) developed by Faust *et al.*, [1]. CAFVI was tested both in simulation and in experiments for the aerial suspended cargo task. CAFVI is *sample-efficient*, needing only a constant number of samples at every time-step to produce a control input, and is *consistent*, meaning that the resulting control input does not depend on the available samples. For these characteristics we selected CAFVI as our learning algorithm that learns the problem's value function. Once the value function is learned, *convex sum policy* [1] is used to generate control input for the system starting at different initial conditions.

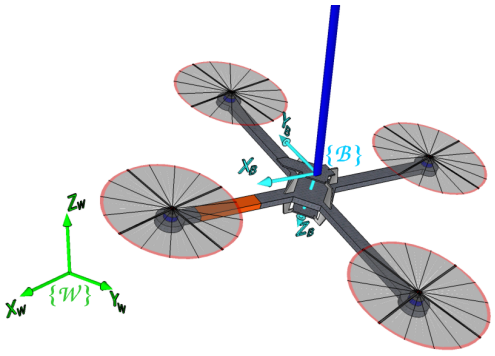


Fig. 1. The inertial frame $\{\mathcal{W}\}$ and the body-fixed frame $\{\mathcal{B}\}$ for a quadrotor balancing a pendulum.

The novel approach presented here decomposes the flying inverted pendulum task into two subtasks, *Initial Balancing* that places pendulum very close to upright position, and *Balanced Hover* that slows down the quadrotor to a hover while maintaining the upright inverted pendulum position. We learn both tasks with CAFVI, and generate trajectories with the *convex sum policy* also developed by Faust *et al.*, [1]. The joint controller, *Flying Inverted Pendulum*, sequentially combines the two subtasks to solve the flying inverted pendulum. It produces a control action that drives the system from an arbitrary initial inverted pendulum displacement to a stable state where the aerial vehicle hovers maintaining the inverted pendulum minimally displaced from the upright position.

*This work is partially supported by NM Space Grant to A. Faust, NIH Grant P20RR018754 to the Center for Evolutionary and Theoretical Immunology.

II. PRELIMINARIES

This section gives a necessary background on task learning with CAFVI algorithm and control action generation given an initial condition. The content of this section is a summary of [1].

A. Reinforcement Learning Paradigm

Consider the discrete-time nonlinear system

$$\mathbf{x}_{k+1} = \mathscr{F}(\mathbf{x}_k, \mathbf{u}_k), \tag{1}$$

where k is the discrete time, $\mathbf{x} \in X$ represents the state vector, $\mathbf{u} \in U$ denotes the control action and $\mathscr{F} : X \times U \to X$ is the system function. Also, a reward function is defined $\rho : X \times U \to \mathbb{R}$ to evaluate the immediate effect of the action $\mathbf{u}_k$. The tuple $\{X, U, \mathscr{F}, \rho, \}$ constitutes a Markov Decision Process (MDP) [3], [10], [4].

The control policy for the system is defined by a function $h : X \to U$ such that $\mathbf{u}_k = h(\mathbf{x}_k)$. A solution to MDP is an *optimal policy* h^*, which maximizes *the return*, *i.e.*, the accumulated reward. The return depends on the policy because the accumulated reward reachable from the given state is determined by the policy that the system follows. Two types of value functions exist for a given policy h, a state value function $V : X \to \mathbb{R}$ (V-function), and a state-action value function $Q : X \times U \to \mathbb{R}$ (Q-function). By definition,

$$Q(\mathbf{x}_k, \mathbf{u}_k) = V(\mathscr{F}(\mathbf{x}_k, \mathbf{u}_k)) = V(\mathbf{x}_{k+1}). \tag{2}$$

A value function, a V-function or a Q-function, represents the accumulated reward obtained by the controller in a long run.

A solution to a MDP is to find an optimal policy which maximizes the value function, *i.e.*, maximizes the cumulative reward over the course of interaction. When all the components of the MDP are known, dynamic programming methods can produce an optimal policy [3].

Value iteration is a major class of RL techniques where the optimal value function is iteratively found and used to compute an optimal policy [4]. Required tabular representation of the value function is impractical or infeasible when the state X and action U spaces are large or continuous. In these cases, the V-function has to be approximated yielding approximate value iteration (AVI) methods [3]. For instance, a V-function can be approximated as

$$V(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{F}(\mathbf{x}), \tag{3}$$

where $\boldsymbol{\theta}$ is a parameter vector and $\mathbf{F}(\mathbf{x})$ is a vector of features [3]. Offline variants of AVI methods have learning and policy generation split in two distinct phases. During the *learning phase* the AVI algorithm learns the feature vector parametization. The method randomly samples the state space in simulation and comes up with the updates for the parameterization vector typically in the expectation-maximization (EM) manner [3]. In the *policy generation phase*, the learning stops and the system enters a control-feedback loop.

B. Continuous Action Fitted Value Iteration (CAFVI)

CAFVI is an offline approximate value iteration method for tasks that balance several opposing constraints on the system [1]. It is appropriate for high-dimensional input spaces for control-affine nonlinear systems. When (1) is a control-affine nonlinear system, it can be written as

$$\mathbf{x}_{k+1} = \mathscr{F}(\mathbf{x}_k, \mathbf{u}_k) = f(\mathbf{x}_k) + g(\mathbf{x}_k)\mathbf{u}_k. \tag{4}$$

In CAFVI, the value function is approximated using (3) considering that $\mathbf{F}(\mathbf{x})$ is a vector of quadratic feature state functions (QFSFs) defined by the user. For example, a QFSF can be $\|\mathbf{x}\|^2$. CAFVI works in continuous state and input spaces. It employs both value functions, a V-function and a Q-function. It learns the parameterization for the V-function, but uses a Q-function based policy, so it learns local Q-function for a fixed state.

C. Control Action Generation with Convex Sum Policy

Given an initial condition, the parameterization vector, the feature vectors, and the black-box system simulator, the control action generation process uses *convex sum policy* [1] to determine the next control input. A controller based on the *convex sum policy* transitions the system to the completion state without any assumptions of the system dynamics. For each dimension, the actions are selected from the appropriate axis and the Q-function is sampled. Then, the samples are fitted with univariate polynomials. The value of the maximum of the polynomial on the segment is found. The resulting action is a convex linear combinations of the orthogonal action vectors that maximizes each of its corresponding interpolations. This action is applied to the system, and the resulting state is observed. The convex sum policy is fast because it requires constant number of Q-function samples. Further the resulting action does not depend on the samples available and the policy gives consistent results. For these reasons, the convex sum policy is a good candidate for the flying inverted pendulum problem.

III. METHODOLOGY

A. Quadrotor-Pendulum Model

The model presented here is used for simulations and for next state evaluation during the convex policy.

We assume that the the pendulum mass is small compared to the quadrotor mass, so the reactive forces of the pendulum on the quadrotor are negligible. [2].

The world coordinate frame $\{\mathcal{W}\}$ and the body-fixed frame $\{\mathcal{B}\}$ are shown in Fig. 1. $\{\mathcal{B}\}$ is attached to the center of mass of the quadrotor. The rigid body equations of motion of the quadrotor are [11]

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \mathbf{R} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \tag{5}$$

$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\Omega}_{\times}, \tag{6}$$

$$\mathbf{I}\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times \mathbf{I}\boldsymbol{\Omega} + \boldsymbol{\tau}, \tag{7}$$

where

$m \in \mathbb{R}$	is the mass of the quadrotor,
$[x\ y\ z]^T \in \mathbb{R}^3$	is the position of the quadrotor center of mass respect to $\{\mathcal{W}\}$,
$g \in \mathbb{R}$	is the constant gravitational acceleration,
$\mathbf{R} \in SO(3)$	is the rotational matrix of $\{\mathcal{B}\}$ with respect to $\{\mathcal{W}\}$,
$T \in \mathbb{R}$	is the total upward thrust,
$\boldsymbol{\Omega} \in \mathbb{R}^3$	is the angular velocity of $\{\mathcal{B}\}$ with respect to $\{\mathcal{W}\}$,
$\mathbf{I} \in \mathbb{R}^{3\times3}$	is the constant inertia matrix expressed in $\{\mathcal{B}\}$,
$\boldsymbol{\tau} \in \mathbb{R}^3$	is the moment applied to the quadrotor by the aerodynamics of the rotors.

The notation $\boldsymbol{\Omega}_\times$ denotes the skew-symmetric matrix. Also, we denote as $\mathbf{r}$ the position of the quadrotor center of mass, *i.e.*, $\mathbf{r} = [x\ y\ z]^T$.

The pendulum can be considered as a point mass that is rigidly attached to the quadrotor center of mass [2]. Also, we assume that the z-position of the quadrotor is fixed. Let a and b be respectively the x-position and y-position of the pendulum center of mass relative to its base expressed respect to $\{\mathcal{W}\}$. We denote as $\mathbf{p}$ the position of the pendulum center of mass, *i.e.*, $\mathbf{p} = [a\ b]^T$. The relative z-position of the pendulum center of mass is given by

$$c = \sqrt{L^2 - a^2 - b^2}, \tag{8}$$

where L is the length from the origin of $\{\mathcal{B}\}$ to the center of mass of the pendulum.

The kinetic energy $\mathcal{K}$ and the potential energy $\mathcal{U}$ of the pendulum are

$$\mathcal{K} = \frac{m_p}{2}\left[(\dot{x}+\dot{a})^2 + \left(\dot{y}+\dot{b}\right)^2 + \left(\frac{a\dot{a}+b\dot{b}}{c}\right)^2\right],$$
$$\mathcal{U} = m_p g c,$$

where m_p is the mass of the pendulum. The nonlinear dynamics of the pendulum can be derived from the Lagrangian $\mathcal{L} = \mathcal{K} - \mathcal{U}$ applying conventional Lagrangian mechanics. Then, we obtain

$$\ddot{a} = \frac{ab\,\varrho}{L^2c^4} - \frac{\left(L^2-a^2\right)\zeta}{L^2c^4}, \tag{9}$$
$$\ddot{b} = \frac{ab\,\zeta}{L^2c^4} - \frac{\left(L^2-a^2\right)\varrho}{L^2c^4}, \tag{10}$$

where

$$\varrho = b\left(a\dot{a}+b\dot{b}\right)^2 + c^2 b\left(\dot{a}^2+\dot{b}^2\right) - gc^3 b + c^4\ddot{y}$$
$$\zeta = a\left(a\dot{a}+b\dot{b}\right)^2 + c^2 a\left(\dot{a}^2+\dot{b}^2\right) - gc^3 a + c^4\ddot{x}$$

Since we assume that the quadrotor maintains its altitude, just the accelerations in x and y drive the motion of the pendulum through (9) and (10). Because the quadrotor is differentially flat [12], [13], a baseline controller for the quadrotor that has as inputs the desired translational acceleration $\ddot{\mathbf{r}}_d = [\ddot{x}_d\ \ddot{y}_d\ \ddot{z}_d]^T$, and the desired yaw angular acceleration $\ddot{\psi}_d$ can be implemented in a similar fashion as in [14]. This control scheme is shown in Fig. 2. For the pendulum block, we use a linear model obtained applying linearization of the pendulum equations of motion (9) and (10). The implementation of the baseline controller is without assuming the presence of the pendulum. Thus, we learn to balance the inverted pendulum using CAFVI.

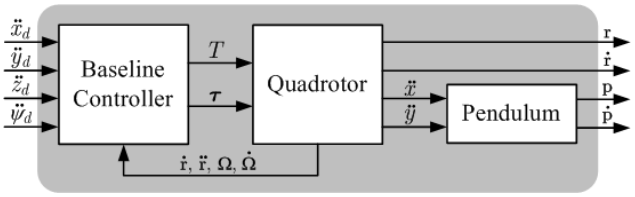


Fig. 2. Block diagram of the reference model.

B. Problem Statement

From our assumption that the quadrotor maintains its height and from the pendulum dynamics (9) and (10), the control inputs $\ddot{z}_d$ and $\ddot{\psi}_d$ of the system shown in Fig. 2 do not help to balance the pendulum. Therefore, we consider that the quadrotor keeps its initial altitude and heading at all times. This condition can be achieved setting up $\ddot{z}_d = \ddot{\psi}_d = 0$. Thus, the control action for balancing the pendulum is

$$\mathbf{u} = \begin{bmatrix}\ddot{x}_d & \ddot{y}_d\end{bmatrix}^T. \tag{11}$$

Given an initial displacement of the pendulum center of mass $\mathbf{p}_0 = [a_0\ b_0]^T$, our goal is to find an optimal control input $\mathbf{u}$ such that the pendulum is balanced on top of the quadrotor, *i.e.*, $\mathbf{p}$ and $\dot{\mathbf{p}}$ tends to zero, and the quadrotor reaches a hovering state, *i.e.*, $\dot{\mathbf{r}}$ also tends to zero. Thus, the *flying inverted pendulum task* is completed when the system comes to rest. This is when $\mathbf{p} = \dot{\mathbf{p}} = \dot{\mathbf{r}} = \mathbf{0}$. From the practical perspective, we consider the task done when the state norm is sufficiently small.

We decompose the *flying inverted pendulum task* into two subtasks, *initial balance* and *hover balanced tasks*. *Initial balance task* raises the pendulum to an upright position without regard to the quadrotor state. The task is completed when

$$\|\mathbf{p}\| < \epsilon_{p_B},\ \|\dot{\mathbf{p}}\| < \epsilon_{\dot{p}_B}, \tag{12}$$

for small positive constants ϵ_{p_B} and $\epsilon_{\dot{p}_B}$. The subscript B denotes that just the balancing task is under consideration. The second subtask, *hover balanced task* assumes that *initial balance* was completed and the initial conditions satisfy (12). The task requires quadrotor to reduce speed to hover while it maintains the minimal inverted pendulum displacement. It is completed when

$$\|\mathbf{p}\| < \epsilon_{p_{BH}},\ \|\dot{\mathbf{p}}\| < \epsilon_{\dot{p}_{BH}}, \|\dot{\mathbf{r}}\| < \epsilon_{\dot{r}_{BH}}, \tag{13}$$

for small positive constants $\epsilon_{p_{BH}}$, $\epsilon_{\dot{p}_{BH}}$, and $\epsilon_{\dot{r}_{BH}}$. The subscript BH denotes that balancing and hovering are under

consideration. Note that the completion criteria for the *hover balanced* task is the same as for *flying inverted pendulum* task although the latter has less restrictive initial conditions.

Before proceeding, we need to check that the system is control-affine in order to apply the CAFVI algorithm. The combination of the baseline controller and the quadrotor blocks in Fig. 2 gives as a result a double integrator system because of the differential flatness property of the quadrotor [12]. This result together with the equations of motion for the pendulum, (9) and (10), shows clearly that the system is nonlinear control-affine. Because in simulation the discrete-time transitions of the system illustrated in Fig. 2 are obtained by numerical integration of the continuous time dynamics and we set up $\ddot{z}_d = \ddot{\psi}_d = 0$, the baseline controller-quadrotor-pendulum system can be seen as the one-block system presented in Fig. 3 with the complete state of the system denoted as $\mathbf{\Gamma} = [\mathbf{r}\ \dot{\mathbf{r}}\ \mathbf{p}\ \dot{\mathbf{p}}]^T$.

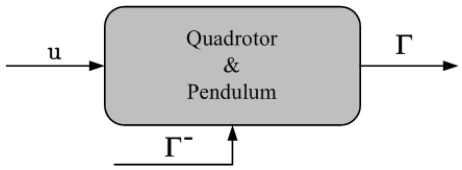


Fig. 3. Concise generative model used for simulation purposes and for next state evaluation during the convex policy, the system state $\mathbf{\Gamma} = [\mathbf{r}\ \dot{\mathbf{r}}\ \mathbf{p}\ \dot{\mathbf{p}}]^T$ and the control action $\mathbf{u} = [\ddot{x}_d\ \ddot{y}_d]^T$. $\mathbf{\Gamma}^-$ denotes the previous system state.

C. Initial Balance

To run Algorithm CAFVI, we need to give a discount factor γ, and input dimensionality d_U. We select $\gamma \in (0, 1)$ and in our case $d_U = 2$. Also, an input to this algorithm is the vector of QFSFs $\mathbf{F}(\mathbf{\Gamma})$. Because one of our goals is to balance the pendulum, we define

$$\mathbf{F}_B(\mathbf{\Gamma}) = [\|\mathbf{p}\|^2\ \|\dot{\mathbf{p}}\|^2]^T. \tag{14}$$

The reward function is defined as

$$\rho_B = \begin{cases} K_B & \text{if } cond_B = true \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

with $cond_B = \|\mathbf{p}\| < \delta_{B_p} \wedge \|\dot{\mathbf{p}}\| < \delta_{B_{\dot{p}}}$, and $K_B, \delta_{B_p}, \delta_{B_{\dot{p}}}$ are positive constant values. We run Algorithm CAFVI until a $\boldsymbol{\theta}$ that accomplishes the task of balancing the pendulum is generated. We denote this result as $\boldsymbol{\theta}_B$.

D. Balanced Hover

Next, we consider *balanced hover* task. Thus, we define

$$\mathbf{F}_{BH}(\mathbf{\Gamma}) = [\|\mathbf{p}\|^2\ \|\dot{\mathbf{p}}\|^2\ \|\dot{\mathbf{r}}\|^2]^T. \tag{16}$$

The reward is

$$\rho_{BH} = \rho_{BH,1} + \rho_{BH,2}, \tag{17}$$

where

$$\rho_{BH,1} = -K_{BH,1}\|\mathbf{p}\| \tag{18}$$

and

$$\rho_{BH,2} = \begin{cases} K_{BH,2} & \text{if } cond_{BH} = true \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

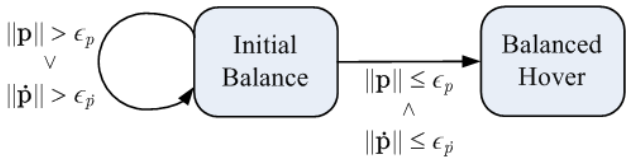
with $cond_{BH} = \|\mathbf{p}\| < \delta_{BH_p} \wedge \|\dot{\mathbf{p}}\| < \delta_{BH_{\dot{p}}} \wedge \|\dot{\mathbf{r}}\| < \delta_{BH_{\dot{r}}}$, and $K_{BH,1}, K_{BH,2}, \delta_{BH_p}, \delta_{BH_{\dot{p}}}, \delta_{BH_{\dot{r}}}$ are positive constant values. Under these conditions, we run Algorithm CAFVI until we obtain a $\boldsymbol{\theta}$ that fulfills the balancing and the hovering tasks. We denote this result as $\boldsymbol{\theta}_{BH}$.

E. Flying Inverted Pendulum

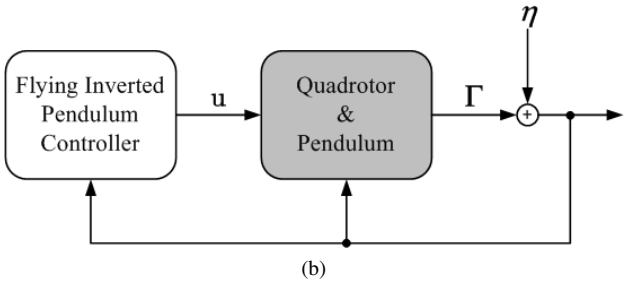
Algorithm 1 depicts learning flying inverted pendulum task. We generate $\boldsymbol{\theta}_B$ and $\boldsymbol{\theta}_{BH}$ as explained in Sections III-C and III-D. We learn using Monte Carlo simulation for a fixed number of trials. Upon generating family of policies, the fittest one is selected for the inverted pendulum controller. The fittest policy reaches the completion state fastest.

Initial balancing policy provides a satisfactory answer for a high initial displacement when no consideration is given to the condition that $\dot{\mathbf{r}}$ reaches zero. When the initial displacement is small, the requirement of $\dot{\mathbf{r}} \to \mathbf{0}$ can be added. For this reason, we used both $\boldsymbol{\theta}_B$ and $\boldsymbol{\theta}_{BH}$ to implement the *flying inverted pendulum* controller. The finite state machine in Fig. 4a shows the transition map and the transition states of our controller.

Algorithm 2 summarizes the proposed control technique. The controller is initially configured for the *Initial Balance* task. It selects a control input according to the convex sum policy explained in Section II-C. When the completion condition for *Initial Balance* task is met, the controller switches to the parameters for *Balanced Hover* task. On the other hand, if the controller state is set to 'Balanced Hover', the controller invokes convex sum policy with the parameters for the *Balanced Hover* task. The output is the state of the controller and the control input.



(a)



(b)

Fig. 4. (a) Finite state machine for the flying inverted pendulum. (b) Scheme of the configuration for simulation tests, $\boldsymbol{\eta}$ is a noise vector.

Algorithm 1 Learning to fly inverted pendulum

Input: γ, vectors of QFSFs $\mathbf{F}_B(\mathbf{\Gamma})$, $\mathbf{F}_{BH}(\mathbf{\Gamma})$, ρ_B, ρ_{BH}, dimension of the action space d_U (in our case $d_U = 2$), number of Monte Carlo simulations mc

▷ Learning Initial balance with Monte Carlo simulation
for $i = 1, ..., mc$ **do**
  $\boldsymbol{\theta}_{B,i} \leftarrow$ CAFVI$(\mathbf{F}_B(\mathbf{\Gamma}), \rho_B, \gamma, d_U)$
end for
$\boldsymbol{\theta}_{B,i} \leftarrow$select fittest $\boldsymbol{\theta}_{B,i}$
▷ Learning Balanced Hover with Monte Carlo simulation
for $i = 1, ..., mc$ **do**
  $\boldsymbol{\theta}_{BH,i} \leftarrow$ CAFVI$(\mathbf{F}_{BH}(\mathbf{\Gamma}), \rho_{BH}, \gamma, d_U)$
end for
$\boldsymbol{\theta}_{BH,i} \leftarrow$select fittest $\boldsymbol{\theta}_{BH,i}$

Output: parameter vectors $\boldsymbol{\theta}_B, \boldsymbol{\theta}_{BH}$,

Algorithm 2 Flying inverted pendulum controller

Input: parameter vectors $\boldsymbol{\theta}_B$, $\boldsymbol{\theta}_{BH}$, vectors of QFSFs $\mathbf{F}_B(\mathbf{\Gamma})$, $\mathbf{F}_{BH}(\mathbf{\Gamma})$, small positive constants ϵ_p, $\epsilon_{\dot{p}}$, dimension of the action space d_U (in our case $d_U = 2$), controller state

measure initial state $\mathbf{\Gamma}_0$
if controller state is 'Initial Balance' **then** ▷ Initial balance
  $\hat{\mathbf{u}} \leftarrow$ convexSumPolicy$(\mathbf{F}_B(\mathbf{\Gamma}), \boldsymbol{\theta}_B, \mathbf{\Gamma}_0)$
  if $\|\mathbf{p}_0\| \leq \epsilon_p \wedge \|\dot{\mathbf{p}}_0\| \leq \epsilon_{\dot{p}}$ **then**
    ▷ switch controller state
    controller state ← 'Balanced Hover'
  end if
else ▷ Balanced Hover
  $\hat{\mathbf{u}} \leftarrow$ convexSumPolicy$(\mathbf{F}_{BH}(\mathbf{\Gamma}), \boldsymbol{\theta}_{BH}, \mathbf{\Gamma}_0)$
end if

Output: $\mathbf{u} = \hat{\mathbf{u}}$, controller state

IV. Results

This section evaluates the flying inverted pendulum controller. We examine each of the subtasks, *Initial Balance* and *Balanced Hover* separately, as well as the joint *Flying Inverted Pendulum* controller from Algorithm 2. The evaluation goals are to determine the computational speed, region of attraction, and noise tolerance.

All simulations are performed in Matlab 2012a running Windows 7 with Pentium Dual-Core CPU and 4Gbs of RAM. The configuration scheme is presented in Fig. 4b. $\boldsymbol{\eta}$ is a noise vector, so we can add a percentage of noise and test the performance of the controller. The parameters for the simulations are: $L = 0.5$ m, $m_p = 0.1$ kg, $K_B = 10^6$, $K_{BH,1} = 100$, $K_{BH,2} = 10^6$, $\delta_{B_p} = 0.01$ m, $\delta_{B_{\dot{p}}} = 0.01$ m/s, $\delta_{BH_p} = 0.05$ m, $\delta_{BH_{\dot{p}}} = 0.05$ m/s and $\delta_{BH_{\dot{r}}} = 0.5$ m/s. The terminating condition for *Initial Balance* controller is $\|\mathbf{p}\| \leq 0.01$ m and $\|\dot{\mathbf{p}}\| \leq 0.01$ m/s, while the terminating condition for the *Balanced Hover* and *Flying Inverted Pendulum* polices is $\|\mathbf{p}\| \leq 0.05$ m, $\|\dot{\mathbf{p}}\| \leq 0.05$ m/s, and $\|\dot{\mathbf{r}}\| \leq 0.5$ m/s.

Algorithm 1 gives as results

$$\boldsymbol{\theta}_B = [-86.6809 \quad -0.3345]^T,$$
$$\boldsymbol{\theta}_{BH} = [-1.6692 \quad -0.0069 \quad 0.0007]^T \times 10^6$$

Both approximated V-functions, calculated with (3), are much more sensitive to changes in pendulum's position than changes in velocity. To visualize this characteristic, Fig. 5 presents the V-function for the *Initial Balance* task projected onto a and $\dot{a}$ axes. The similar preference is seen in the *Balanced Hover* with $\boldsymbol{\theta}_{BH}$. Here, only when the position and velocity of the inverted pendulum are close to the upright position, the controlled reduces the quadrotor's speed.

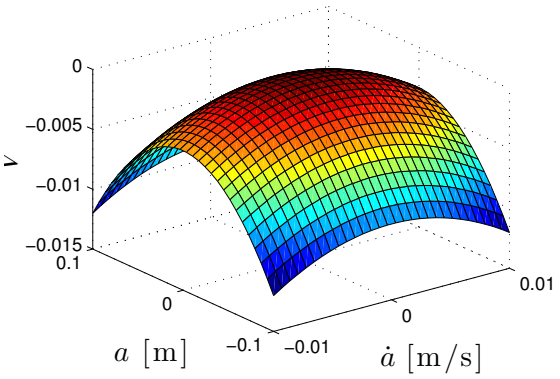


Fig. 5. Approximate V-function associated with θ_B projected onto a and $\dot{a}$.

To examine the policies' characteristic, we randomly selected 100 initial system state conditions $\mathbf{\Gamma}$ for each policy, *Initial Balance* with no noise, and with 5% of randomly added noise, *Balanced Hover* with no noise, and with 2% of noise, *Flying Inverted Pendulum* with no noise, and with 5% of noise. The ranges for initial conditions per policy are presented in Table I.

TABLE I
Range for $\dot{\mathbf{r}}$ and $\mathbf{p}$ to generate simulation initial condition samples

Controller	$\dot{\mathbf{r}} = [\dot{x}\ \dot{y}]^T$	$\mathbf{p} = [a\ b]^T$
Initial Balance	$\dot{x} = \dot{y} = 0$ [m/s]	$a, b \in [-0.2, 0.2]$ [m]
Balanced Hover	$\dot{x}, \dot{y} \in [-5, 5]$ [m/s]	$a, b \in [-0.01, 0.01]$ [m]
Flying Inv. Pend.	$\dot{x} = \dot{y} = 0$ [m/s]	$a, b \in [-0.2, 0.2]$ [m]

For all policies, and all the initial conditions, simulations resulted in reaching the task terminal conditions. That means that the initial conditions in Table I are within the policies regions of attraction.

Table II shows the performance of the individual controller states *Initial Balance*, *Balanced Hover*, and for the complete *Flying Inverted Pendulum* controller. In this table, t_f is the

time to achieve the task terminal conditions measured in system time, t_c is the computation time for a full trajectory, $\mathbf{p}_f$ is the final displacement of the pendulum, $\dot{\mathbf{p}}_f$ is the final velocity of the pendulum, and $\dot{\mathbf{r}}_f$ is the final velocity of the quadrotor. We note that the system time t_f is an order of magnitude higher than the computational time to calculate the next action, rendering our method real-time capable.

Initial Balance policy brings the inverted pendulum to upright position efficiently for initial displacement of up to 34°, but the quadrotor's velocity remains constant at the completion, see Table II. This is expected due to the policy design. *Balanced hover*, on the other hand, reduces quadrotor speed to zero, while maintaining the upright pole position and its small velocities. The downside is that it is capable in doing so only for a small initial pole displacement. Finally, the *Flying Inverted Pendulum* policy handles large initial pole displacements, and brings the system without noise to the the task terminal conditions in a maximum time of 5.38 s.

Lastly, the Table II shows that all three policies are capable of handling some level of random noise. This is important because it shows that the controllers have some tolerance to random disturbances and unmodeled system dynamics without impacting their performance significantly.

Fig. 6, 7, and 8 show the trajectories obtained with each of the three policies. Pendulum positions and velocity, and quadrotor speed are depicted. Fig. 6 depicts the results considering only the *Initial Balance* task starting at $\boldsymbol{p} = [0.2 \ -0.2]^T$ m. This figure shows that Initial Balance policy brings the pole to the upright position with minimal velocity, Fig. 6a and 6b respectively, while the quadrotor velocity becomes constant, Fig. 6c. The quadrotor's residual velocity is considered in the two following policies, see Fig. 7c and 8c. Fig. 8 shows the results obtained using the policy *Flying Inverted Pendulum*. The policy *Initial Balancing* starts by default and reduces the position and velocity of the pendulum, once its terminal conditions are achieved, the *Balanced Hover* policy reduces the quadrotor velocity rapidly to zero while returning the pendulum to the upright position with minimal velocity. Fig. 9 shows the results obtained using the policy *Flying Inverted Pendulum* while 5% of random noise is being added.

V. Conclusion

In this paper, we proposed a novel solution for the flying inverted pendulum problem, *i.e.*, balancing a pendulum on top of a quadrotor. Our method is based on a reinforcement learning algorithm for approximate value iteration which works for multi-dimensional nonlinear control-affine systems. Establishing that the pendulum-quadrotor system is control-affine, we applied this algorithm to generate two approximate near-optimal state value functions: one for the task of balancing the pendulum and the other for the task of balancing the pendulum plus reaching a closed hovering state. Then, we created a Inverted Flying Pendulum controller that computes a control action using both value functions. We evaluated the proposed controller in simulation considering a noisy system. The controller handled a large initial displacement of the pendulum driving the system close to the stable state given by small pendulum displacement and closed-to-zero velocity for the aerial vehicle and the pendulum.

References

[1] A. Faust, P. Ruymgaart, M. Salman, R. Fierro, and L. Tapia, "Continuous action reinforcement learning for underactuated dynamical system control," *Adaptive Motion Planning Research Group Technical Report TR13-002*, 2013, under submission. [Online]. Available: https://cs.unm.edu/amprg/Publications/afaust-TR13-002.pdf

[2] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 763–770.

[3] L. Buşoniu, R. Babuška, B. D. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2010.

[4] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[5] 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 (ICRA)*, 2010, pp. 1642–1648.

[6] F. Mueller, A. Schöellig, and R. D'Andrea, "Iterative learning of feed-forward corrections for high-performance tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3276–3281.

[7] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Learning swing-free trajectories for UAVs with a suspended load," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4902–4909.

[8] I. Palunko, A. Faust, P. Cruz, L. Tapia, and R. Fierro, "A reinforcement learning approach towards autonomous suspended load manipulation using aerial robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 4896–4901.

[9] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Automated aerial suspended cargo delivery through reinforcement learning," *Adaptive Motion Planning Research Group Technical Report TR13-001*, 2013, under submission. [Online]. Available: https://cs.unm.edu/amprg/Publications/afaustTR13-001.pdf

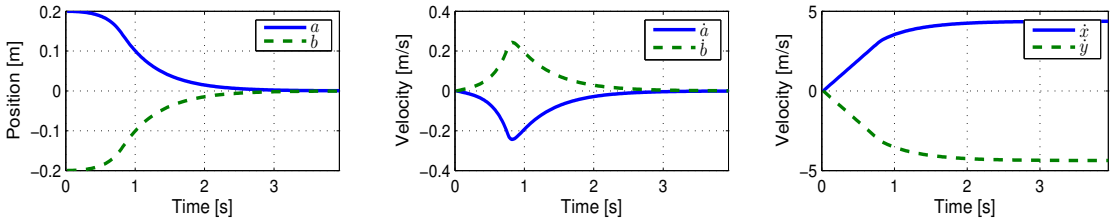
[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. I and Vol. II*, 3rd ed. Athena Scientific, 2005 (Vol. I) and 2012 (Vol. II).

[11] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[12] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

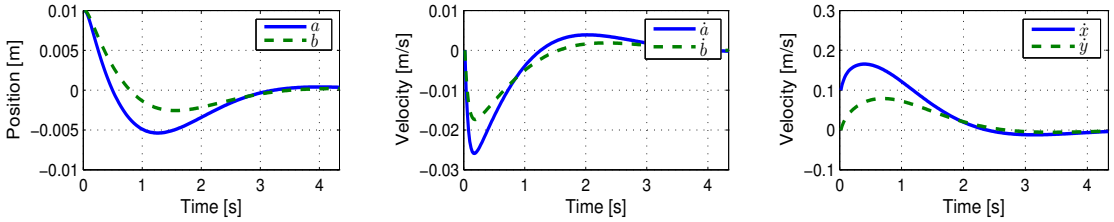
[13] A. Chamseddine, Y. Zhang, C. Rabbath, C. Join, and D. Theilliol, "Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2832–2848, 2012.

[14] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *IEEE Robotics Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.



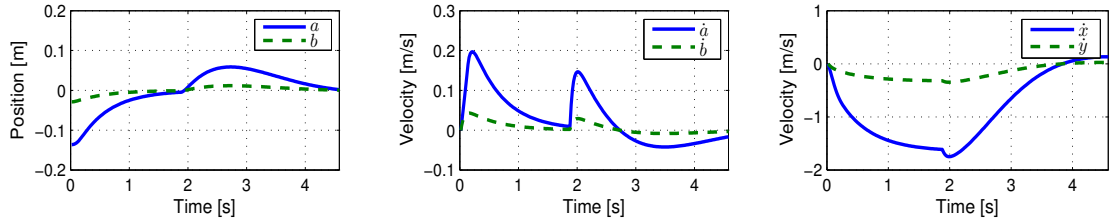
(a) Pendulum position (b) Pendulum velocity (c) Quadrotor velocity

Fig. 6. Trajectory created with *Initial Balance* policy.



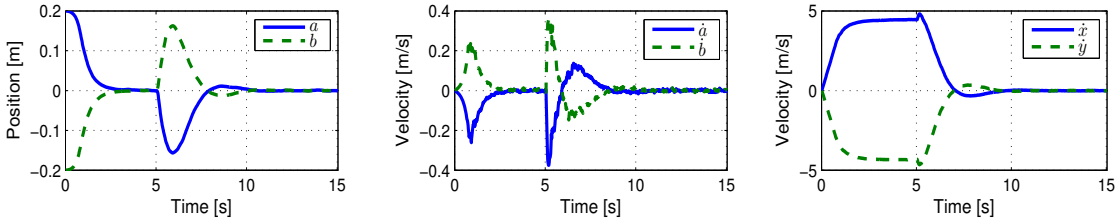
(a) Pendulum position (b) Pendulum velocity (c) Quadrotor velocity

Fig. 7. Trajectory created with *Balanced Hover* policy.



(a) Pendulum position (b) Pendulum velocity (c) Quadrotor velocity

Fig. 8. Trajectory created with *Flying Inverted Pendulum* policy.



(a) Pendulum position (b) Pendulum velocity (c) Quadrotor velocity

Fig. 9. Trajectory created with *Flying Inverted Pendulum* policy with 5% of noise.

TABLE II

SIMULATION RESULTS FOR 100 RANDOM INITIAL CONDITIONS FOR THE THREE CONTROLLERS WITH AND WITHOUT NOISE.

	Noise	t_f [s]			t_c [s]			$\|\mathbf{p}_f\|$ [mm]			$\|\dot{\mathbf{p}}_f\|$ [mm/s]			$\|\dot{\mathbf{r}}_f\|$ [mm/s]		
	%	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max
Initial	0	0.67	0.04	1.32	0.03	0.00	0.06	54	26	98	82	15	96	1622	25	4121
Balancing	5	0.68	0.04	1.54	0.03	0.00	0.09	53	27	99	85	15	105	1593	24	5545
Balanced	0	5.55	5.48	5.60	0.19	0.18	0.22	0	0	0	2	1	4	24	10	40
Hover	2	2.86	0.22	5.16	0.10	0.01	0.17	2	1	2	32	7	49	293	33	495
Flying	0	4.50	1.06	5.38	0.17	0.05	0.28	2	1	2	20	7	42	165	35	438
Inv. Pend.	5	4.51	0.76	5.50	0.17	0.04	0.32	2	1	2	21	4	46	173	25	446