A Reinforcement Learning Approach Towards Autonomous Suspended Load Manipulation Using Aerial Robots

Ivana Palunko[1], Aleksandra Faust[2], Patricio Cruz[3], Lydia Tapia[2], and Rafael Fierro[3]

***Abstract*— In this paper, we present a problem where a suspended load, carried by a rotorcraft aerial robot, performs trajectory tracking. We want to accomplish this by specifying the reference trajectory for the suspended load only. The aerial robot needs to discover/learn its own trajectory which ensures that the suspended load tracks the reference trajectory. As a solution, we propose a method based on least-square policy iteration (LSPI) which is a type of reinforcement learning algorithm. The proposed method is verified through simulation and experiments.**

Keywords Aerial robotics, aerial load transportation, motion planning and control, machine learning, quadrotor control, trajectory tracking, reinforcement learning

I. Introduction

Unmanned aerial vehicles (UAVs) are flexible platforms with an increasing role in assisting humans in tasks that are deemed too risky or in some cases impractical for humans. Examples of such missions are search and rescue, environmental monitoring, surveillance, and cooperative manipulation. The agility and speed of rotorcraft UAVs (RUAVs) allow them not only to take off and land in a very limited area, but also to carry and manipulate payloads more precisely and efficiently. Therefore, they can be well suited for use in urban environments, canyons, caves, and other cluttered environments for example in assistive roboticscarrying and/or transporting a load.

RUAV existing applications vary from basic hovering [1] and trajectory tracking [2], to formation control [3], surveillance [4], aggressive maneuvering [5], aerobatic flips [6] and aerial manipulators [7]. Trajectory tracking for helicopters based on reinforcement learning resulted in autonomous aggressive helicopter flights. This work was summarized by Abbeel et al. [8]. This line of research relies on apprenticeship learning to achieve aggressive autonomous maneuvers such as as flips, rolls, loops, chaos, tic-tocs, and auto-rotation landings. An expert pilot demonstrates the target trajectory and the cost function is learned from the target trajectory. Parameters for an initial linear model are learned from observing an expert flying non-aggressive maneuvers. Then, the trajectory is broken down into small segments, so that each one can be learned separately using a linear dynamics model.

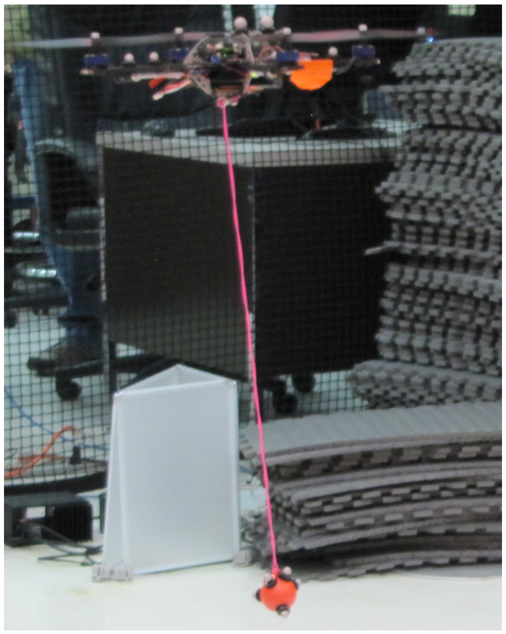


Fig. 1. Hummingbird quadrotor with a suspended load at MARHES Lab - University of New Mexico.

Quadrotors, a type of RUAV, are inherently unstable systems with highly nonlinear dynamics. The task of load carrying puts additional strain on the controller. The reason for this is because the suspended load alters quadrotor dynamics with forces and torques transferred through the suspension cable. Furthermore, maneuvering with a suspended load can be a hazardous task, especially in cluttered and hardly accessible environments. Reducing the residual oscillations of the suspended load is one of the methods to achieve better control over the system. Another method is to control the trajectory of the suspended load directly. In this paper we focus on the problem where a suspended load, carried by a rotorcraft aerial robot, performs trajectory tracking. The solution to this problem has a broad spectrum of applications ranging from motion planning and aerial robotics to control of cranes and robotic manipulators [9], [10].

In order to solve this problem we propose a method based on least square policy iteration (LSPI), that is a type of a reinforcement learning algorithm. One of the main advantages of this method is that it is model free, meaning that in the

This work was supported by NSF grant ECCS #1027775 and the Department of Energy URPR Grant #DE-FG52-04NA25590. Ivana Palunko is supported by the European Community's Seventh Framework Programme under grant No. 285939 (ACROSS). Lydia Tapia is supported in part by the National Institutes of Health (NIH) Grant P20RR018754 to the Center for Evolutionary and Theoretical Immunology.

[1] Ivana Palunko, is with the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, ivana.palunko@fer.hr

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

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

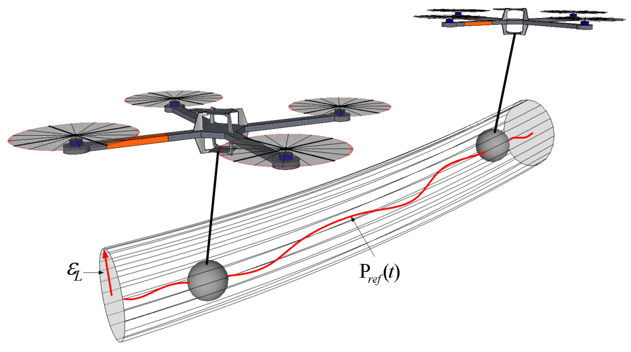


Fig. 2. Load trajectory tracking.

design process we did not use any explicit knowledge about the system model. This makes the method robust to model uncertainties and noise, as it is demonstrated in our results. Another advantage is that it can be implemented as an online learning algorithm which brings us a step further towards fully autonomous UAV transportation system. Online least square policy iteration has been successfully implemented in order to solve benchmark control problems of balancing an inverted pendulum and balancing and riding a bicycle [11], [12]. The method shows promising convergence properties [13] and algorithmic stability [12].

The rest of the paper is organized as follows: in Section II, we formally state the problem, Section III discusses the algorithm and methodology in detail including algorithm convergence, Section IV shows the simulation and experimental results implemented on a quadrotor with suspended load, while Section V presents the closing remarks.

II. Problem Statement

In our previous work [14] we used a dynamic programming approach in order to generate trajectories which ensured swing-free maneuvers of a suspended load carried by a quadrotor in an open-loop fashion. By open-loop we mean that there was no feedback involved in order to control the swing of the load, just the profile of the trajectory was shaped based on the model of the system. Since this approach was a model-based method, the experimental results showed sensitivity with respect to model uncertainties. In this paper we move a step further, as we focus on trajectory tracking of a suspended load as depicted in Figure 2 and defined as:

Definition 1: Given a reference trajectory $\mathcal{P}_{ref}(t) = [x_{Lref}(t) \quad y_{Lref}(t) \quad z_{Lref}(t)]^T$, the position of the suspended load $P_L(t) = [x_L(t) \quad y_L(t) \quad z_L(t)]^T$ and a small positive constant $\epsilon_L > 0$, we say that the suspended load performs trajectory tracking if $|e_L(t)| \leq \epsilon_L$ where $e_L(t) = [x_{Lref}(t) - x_L(t) \quad y_{Lref}(t) - y_L(t) \quad z_{Lref}(t) - z_L(t)]^T$. We want to achieve this goal by specifying the reference trajectory for the suspended load only. The quadrotor needs to learn its own trajectory and at the same time to ensure that the suspended load tracks the reference trajectory. Therefore we are faced with an optimal control problem again. Since the quadrotor should learn it's trajectory from the interaction between a learning algorithm and the aerial load manipulation system, we choose least-square policy iteration (LSPI), a reinforcement learning algorithm, as the basis of our approach. The implementation is depicted in Figure 3. The inputs to the LSPI algorithm are the vector of the tracking error and the vector of the suspended load displacement angles. There variables are obtained by the sensors like an on-board camera or an indoor positioning system. The output from the LSPI algorithm are three control signals which are sent to the quadrotor. These signals are additional control inputs to the quadrotor control system. Since they are bounded, which is defined by the algorithm [11], the quadrotor will remain stable.

III. Suspended Load Trajectory Tracking Using Least Square Policy Iteration

Model-free reinforcement learning (RL) algorithms can be divided into three categories:

- value iteration algorithms that iteratively construct optimal value function from which the a greedy policy is derived,
- policy iteration algorithms which construct value functions that are then used to construct new, improved policies,
- policy search algorithms that use optimization techniques to directly search for an optimal policy.

Offline RL algorithms use data collected in advance while online RL algorithms learn a solution by interacting with the process. Online RL algorithms must balance the need to collect the informative data (exploration) with the need to control the process well (exploitation).

Policy iteration (PI) algorithms iteratively evaluate and improve control policies. Least-squares techniques for policy evaluation are sample-efficient and have relaxed convergence requirements.

The quadrotor with a suspended load can be represented as a deterministic Markov Decision Process (MDP) (X, U, f, ρ) where X is the state space of the process, U is the action space of the controller, $f : X \times U \to X$ is the transition function of the process

$$x_{k+1} = f(x_k, u_k)$$

and $\rho : X \times U \to \mathbb{R}$ is the reward function

$$r_{k+1} = \rho(x_k, u_k)$$

The controller chooses actions according to its policy $h : X \to U$

$$u_k = h(x_k).$$

The goal is to find an optimal policy that maximizes the return from any initial state x_0. The return R is a cumulative aggregation of rewards

$$R^h(x_0) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$

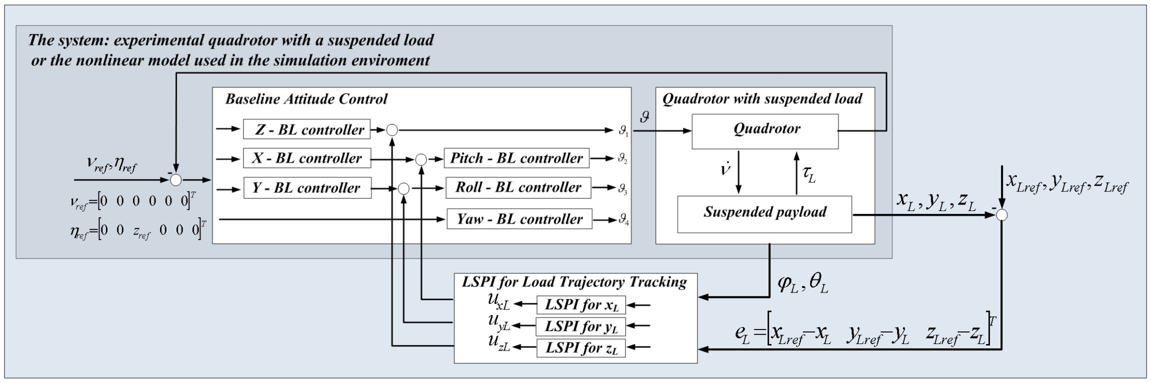

The system: experimental quadrotor with a suspended load or the nonlinear model used in the simulation enviroment
Baseline Attitude Control
Z - BL controller
X - BL controller
Y - BL controller
Pitch - BL controller
Roll - BL controller
Yaw - BL controller
ϑ_1
ϑ_2
ϑ_3
ϑ_4
ϑ
Quadrotor with suspended load
Quadrotor
Suspended payload
$\dot{\nu}$
τ_L
ν_{ref}, η_{ref}
$\nu_{ref} = [0\ 0\ 0\ 0\ 0\ 0]^T$
$\eta_{ref} = [0\ 0\ z_{ref}\ 0\ 0\ 0]^T$
x_L, y_L, z_L
$x_{Lref}, y_{Lref}, z_{Lref}$
φ_L, θ_L
LSPI for Load Trajectory Tracking
u_{xL}
u_{yL}
u_{zL}
LSPI for x_L
LSPI for y_L
LSPI for z_L
$e_L = [x_{Lref} - x_L \quad y_{Lref} - y_L \quad z_{Lref} - z_L]^T$

Fig. 3. Block scheme for load trajectory tracking using LSPI.

where $\gamma \in [0,\ 1]$ is the discount factor. State-action value function (Q-function) $Q^h : X \times U \to \mathbb{R}$ of a policy h is

$$Q^h(x,u) = \rho(x,u) + \gamma R^h(f(x,u))$$

The optimal Q-function is

$$Q^*(x,u) = \max_h Q^h(x,u)$$

The optimal policy (greedy policy in Q^*)

$$h(x) \in arg \max_u Q(x,u)$$

Policy iteration evaluates policies by iteratively constructing an estimate of state action value function. This estimate is then used to construct a new, improved policy. The policy improvement step is done at every iteration l by minimizing the least square error in the Bellman equation [15]

$$Q^{h_l}(x,u) = \rho(x,u) + \gamma Q^{h_l}(f(x,u), h_l(f(x,u)))$$

for Q^{h_l} of the current policy h_l. The new policy is a greedy policy with respect to the state action value function:

$$h_{l+1}(x) \in arg \max_u Q^{h_l}(x,u).$$

The sequence of Q-functions produced by policy iteration converges asymptotically to Q^* as $l \to \infty$. In continuous spaces, policy evaluation cannot be solved exactly, and the value function has to be approximated. Linearly parametrized Q-function approximator $\hat{Q}$ consists of n basis function (BFs) $\phi_1, \ldots, \phi_n : X \times U \to \mathbb{R}$ and n dimensional parameter vector θ

$$\hat{Q} = \sum_{l=1}^{n} \phi_l(x,u)\theta_l = \phi^T(x,u)\theta$$

where $\phi(x,u) = [\phi_1(x,u), \ldots, \phi_n(x,u)]^T$. Control action u is a scalar which is bounded to an interval $U = [u_L \quad u_H]$.

In this paper we use state-dependent basis functions and orthogonal polynomials of the action variable which separates approximation over the state space from approximation over the action space. Orthogonal polynomials are chosen because it is simple to solve the maximization problem over action variables and orthogonality ensure better conditioned regression problem at the policy improvement steps. As approximators over the state-action space we use Chebyshev polynomials of the first kind which are defined as

$$\begin{aligned} \psi_0(\bar{\xi}) &= 1, \\ \psi_1(\bar{\xi}) &= \bar{\xi}, \\ \psi_{j+1}(\bar{\xi}) &= 2\bar{\xi}\psi_j(\bar{\xi}) - \psi_{j-1}(\bar{\xi}), \end{aligned} \tag{1}$$

where

$$\bar{\xi} = -1 + 2\frac{\xi - \xi_L}{\xi_H - \xi_L}. \tag{2}$$

and ξ is any variable for which we build the Chebyshev polynomial. These polynomials are defined on the interval $[-1 \quad 1]$. The approximator over the state-action space is given as

$$\phi(x,u) = \Phi(u) \otimes \Psi(x), \tag{3}$$

where $\Phi(u) \in \mathbb{R}^{N\times 1}$ is the vector of Chebyshev polynomials in action space, $\Psi(x) \in \mathbb{R}^{M\times 1}$ is the vector of Chebyshev polynomials in the state space, N and M are dimensions of Chebyshev polynomials and $\otimes$ is the Kronecker product. Chebyshev polynomials are good approximators because they show uniform convergence as approximators which is shown in [13] and [16]. The online least square policy iteration algorithm is described by algorithm (1). The difference between an offline and an online variation of LSPI is that the online algorithm interacts with the system directly at every iteration (line 10). Since we do not want the algorithm to get stuck in local minima, exploration and exploitations actions have to be balanced (line 9). To be able to converge quickly, the algorithm needs to exploit the interaction with the environment. On the other hand, in order to find the global minimum, it has to explore the actions space with probability ε_k. As the time passes, the algorithm is relying more on the current policy than on exploration, so ε_k decays exponentially as the number of steps increases. Instead of waiting until many samples are passed, the online LSPI improves the policy by solving for the Q-function parameters every K_θ iterations using the current values of the Γ, Λ and z. When $K_\theta = 1$ then the policy is improved at every iteration step and is called fully optimistic. Usually K_θ should be a number greater than zero but not too large. In our case $K_\theta = 7$ which we picked based on the algorithm performance. Changing this parameter the performance of the algorithm

Algorithm 1 Online LSPI with ε - greedy exploration

```
1: Input: discount factor $\gamma$,
2: BFs $\phi_1, \dots \phi_n : X \times U \to \mathbb{R}$,
3: policy improvement interval $K_\theta$, exploration $\{\varepsilon_k\}_{k=0}^{\infty}$,
4: a small constant $\beta_\Gamma > 0$
5: $l \leftarrow 0$, initialize policy $h_0$,
6: $\Gamma_0 \leftarrow \beta_\Gamma I_{n\times n}$, $\Lambda_0 \leftarrow 0$, $z_0 \leftarrow 0$
7: measure initial state $x_0$
8: for every time step   $k = 0, 1, 2, \dots$ do
9:     $u_k \leftarrow \begin{cases} h_l(x_k): \text{ with prob.} 1-\varepsilon_k \text{ (exploit)} \\ \text{uni. rand. ac. in U}: \text{ with prob.} \varepsilon_k \text{ (explore)} \end{cases}$
10:    apply $u_k$, measure next state $x_{k+1}$ and reward $r_{k+1}$
11:    $\Gamma_{k+1} \leftarrow \Gamma_k + \phi(x_k, u_k)\phi^T(x_k, u_k)$
12:    $\Lambda_{k+1} \leftarrow \Lambda_k + \phi(x_k, u_k)\phi^T(x_{k+1}, h_l(x_{k+1}))$
13:    $z_{k+1} \leftarrow z_k + \phi(x_k, u_k) r_{k+1}$
14:    if $k = (l+1)K_\theta$ then
15:        $\theta_l \leftarrow$ solve $\frac{1}{k+1}\Gamma_{k+1}\theta_l = \frac{1}{k+1}\Lambda_{k+1}\theta_l + \frac{1}{k+1}z_{k+1}$
16:        $h_{l+1}(x) \leftarrow \arg\max_u \phi^T(x, u)\theta_l, \forall x$
17:        $l \leftarrow l + 1$
18:    end if
19: end for
```

changes. The reward function vector $r_k \in \mathbb{R}^{3\times 1}$ at step k is defined as

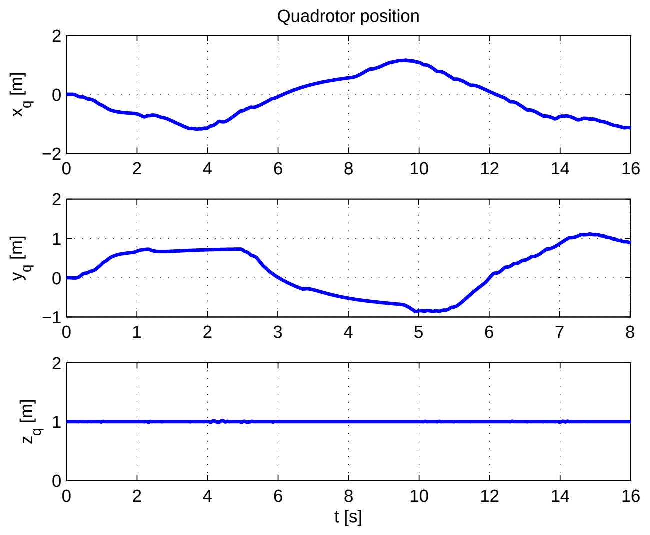
$$r_k = -c_u\,|u_{Lk}| - e_{Lk}^T c_e e_{Lk} - \delta_{Lk}^T c_a \delta_{Lk}, \tag{4}$$

where $u_{Lk} \in \mathbb{R}^{3\times 1}$ is the action vector at step k, $e_{Lk} \in \mathbb{R}^{3\times 1}$ is the vector of the load tracking error at step k and $\delta_{Lk} \in \mathbb{R}^{3\times 1}$ is the load displacement angle vector, $c_u \in \mathbb{R}^{3\times 1}$ is a vector with positive entries, $c_e \in \mathbb{R}^{3\times 3}$ and $c_a \in \mathbb{R}^{3\times 3}$ are diagonal positive definite matrices. With this reward function we penalize the tracking error, load swing angles and actions. Since in LSPI the action and the reward are scalar, in order to implement a 2D or a 3D trajectory tracking, we develop three distinct functions that run three LSPI algorithms in parallel providing us with three control actions for the quadrotor u_{Lx}, u_{Ly} and u_{Lz}, as depicted in Figure 3. Control actions are bounded to the interval $[u_{lo}\; u_{hi}]$.

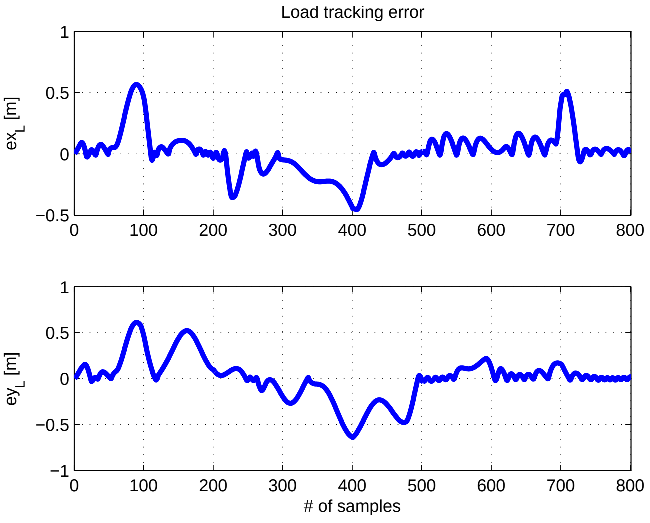
[13, Theorem 4.2 (*Mean convergence of an API algorithm*), Corollary 4.1 (*Mean convergence of RLSAPI*)] and [17, Theorem 7.1 (*Mean convergence of LS/RLSAPI with exact Chebyshev polynomials*)] ensure mean convergence of the approximate policy iteration algorithm with Chebyshev approximators. This guarantees that the solution of the optimal control problem solved using Chebyshev based LSPI generates a sequence of policies that asymptotically converges to the optimal performance in the mean.

IV. Simulation and Experimental Results

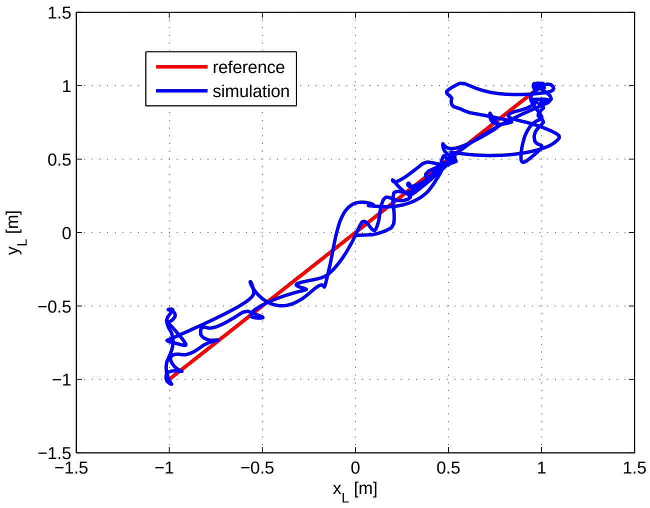
The algorithm described in the previous section is an online LSPI. This means that during every simulation step, one iteration of the algorithm is performed while interacting with the system. Therefore, the algorithm provides an action at every simulation step, based on the previous outcome of the action or based on exploration. We use the nonlinear model of the quadrotor and the suspended load described



(a) Quadrotor position.



(b) Load tracking error.



(c) Load displacement in 2D.

Fig. 4. Simulation results for online LSPI used for suspended load tracking the straight line.

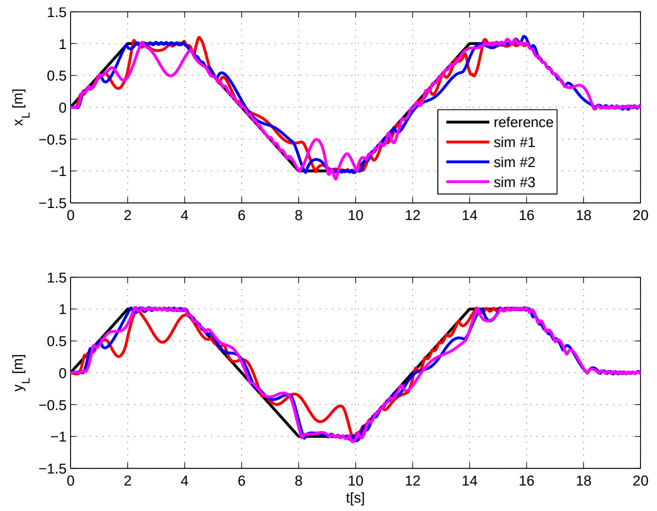


Fig. 5. Simulation results for online LSPI used for suspended load tracking the straight line multiple simulation trials.

in [14] for building the simulation environment. The model provides feedback to the LSI algorithm in form of the load tracking error e_{Lk} and the load displacement angle θ_{Lk} and ϕ_{Lk} for the LSPI algorithm as depicted in Figure 3. The algorithm and simulations were written and executed in Matlab and Simulink 2011, on a Windows machine. In order to check the execution time of the proposed algorithm we ran the algorithm for 10000 steps which took 0.562 seconds. Therefore, the average time for execution of one iteration step k of the LSPI algorithm is $5.62 \cdot 10^{-5}$ seconds. This shows that the algorithm is suitable for real-time implementation. In the algorithm described in the previous section, we use three LSPI algorithms in parallel. In order to be able to execute these algorithms in parallel, we used the Parallel Computing Toolbox from Matlab.

For our first simulation, we have a load tracking a straight line in the xy plane going back and forth. Red lines in Figure 4(c) show the targeted trajectory.We tracked the performance of the algorithm for 800 steps and these results are presented in Figure 4 in blue lines. Figure 4(a) shows the quadrotor trajectory resulting from the learning. Figure 4(c) presents the load trajectory and displacement from the targeted trajectory. Figure 4(b) depicts load tracking error that the agent is trying to minimize. The error is bounded which shows stability of the algorithm over 800 iteration steps. To additionally confirm the convergence and stability of the LSPI algorithm we refer to the Figure 5 which shows simulation results for 3 simulation trials. We can see that the load starts tracking the reference trajectory very fast in each of the simulation trials. Furthermore, we can see that the load closely follows the reference trajectory after 15 seconds, not diverging more than few centimeters from the reference trajectory in all three trials. Figure 6 shows simulation results of the performance of the algorithm with respect to noisy measurements of the displacement angle Φ_L and tracking error e_{xL}. Additionally, at time t = 20s there is a change in the length of the suspension cable of the load. We can see that the algorithm is robust and that the load tracks

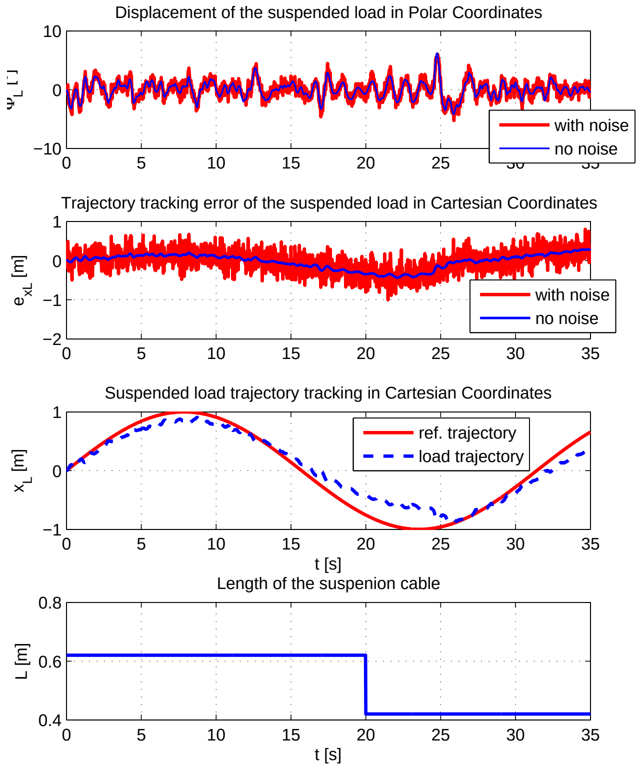


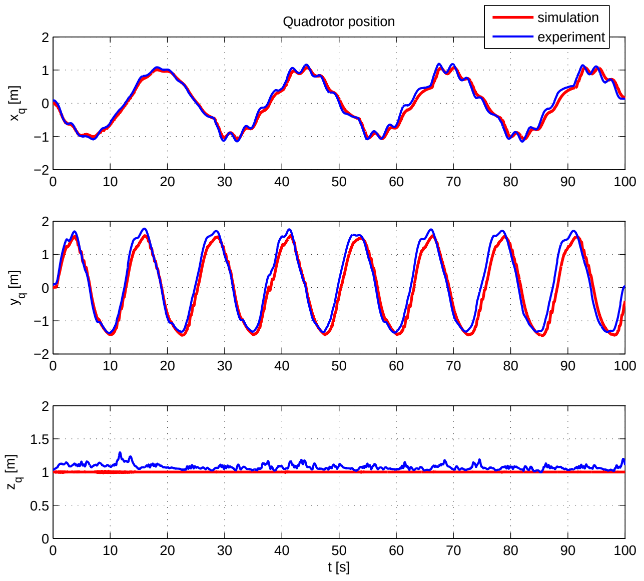
Fig. 6. Simulation results for online LSPI. It shows the algorithm performance with respect to noisy measurements and to variable length of the suspension cable.

the reference trajectory. To verify that the simulation results are sound and that the produced LSPI trajectories are valid, we performed experiments using an AscTec Hummingbird quadrotor (see Figure 1). The experimental results were performed in MARHES lab at University of New Mexico. The weight of the load is 45g and the length of the suspension cable is 62cm. More detailed description of the experimental testbed can be found in [18]. We generated trajectories for the quadrotor by learning in simulation, and tested them on a real quadrotor with suspended load. We measured trajectories of the quadrotor and the load as well as the load displacement angles using VICON indoor motion tracking system. We compare the experimental results with simulation predictions. To show that with LSPI algorithm the system is able to perform more complex trajectories, we chose a Lissajeou curve as a second trajectory to track. Figure 7(a) compares the actual quadrotor trajectory as flown (in blue) with the trajectory predicted by simulation (in red). The differences between them are negligible, never exceeding more than 15 cm. Figure 7(b) compares the load trajectory recorded in the experiment (in blue) with the simulation predicted trajectory (in red) and target trajectory (in black).

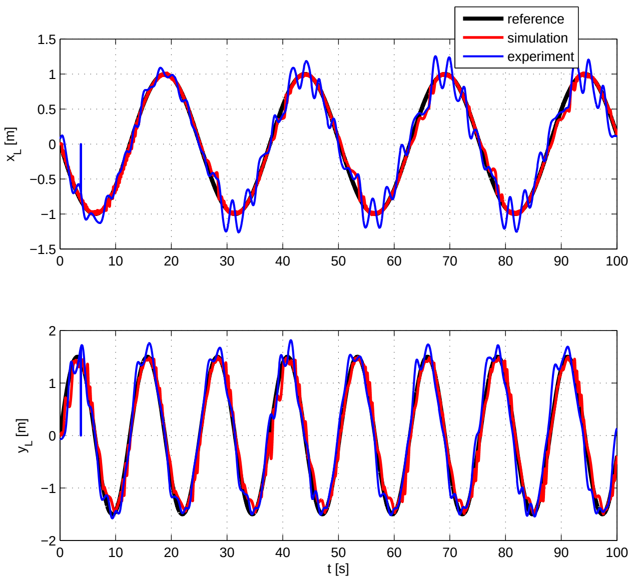
Videos of simulations and experiments can be found on [19].

V. Conclusions

Tracking and controlling the position of a suspended load can be critical to mission success. In this paper we present a model-free approach to solving this problem involving a reinforcement learning algorithm. This method converges



(a) Quadrotor position.



(b) Suspended load position.

Fig. 7. Experimental results using the trajectory generated in simulation using LSPI.

quickly to learn the policy function that minimizes the tracking error of the load with respect to the reference trajectory. We show results in simulation of this policy on a variety of trajectories from a simple straight line to a more complex Lissajeou curve. In both cases, the error is bounded to centimeters. In order to further validate the simulation, experiments were run on a experimental testbed. In these experiments, we see that the load is able to follow a Lissajeou curve trajectory closely.

The proposed method is designed so it can be easily implemented on an off-the-shelf UAV system with minimal prerequisites regarding extra sensors, computational power or additional knowledge of the system's model.

The reinforcement learning methods presented are flexible and can be used both offline and online for trajectory tracking. This flexibility makes it highly appropriate for quadrotor policy learning.

References

[1] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation*, vol. 5, Barcelona, Spain, April 2004, pp. 4393 – 4398.

[2] G. Hoffman, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, April 2008, pp. 1–14.

[3] A. Schöllig, F. Augugliaro, and R. D'Andrea, "A platform for dance performances with multiple quadrocopters," in *IEEE International Conference on Intelligent Robots and Systems*, 2010, pp. 1–8.

[4] M. Valenti, D. Dale, J. How, and D. Pucci de Farias, "Mission health management for 24/7 persistent surveillance operations," in *AIAA Conference on Guidance, Navigation and Control*, Hilton Head, SC, August 2007, pp. 1–18.

[5] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *International Symposium on Experimental Robotics*, New Delhi, India, December 2010.

[6] 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*, May 2010, pp. 1642–1648.

[7] M. Orsag, C. Korpela, M. Pekala, and P. Oh, "Stability in aerial manipulation," in *Proc. of IEEE American Control Conference (ACC)*, 2013, To appear.

[8] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, 2010.

[9] 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*, vol. 130, no. 4, July 2008.

[10] G. Starr, J. Wood, and R. Lumia, "Rapid transport of suspended payloads," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 1394 – 1399.

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

[12] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, December 2003.

[13] J. Ma and W. B. Powell, "A convergent recursive least squares policy iteration algorithm for multi-dimensional markov decision process with continuous state and action spaces," in *IEEE Conference on Approximate Dynamic Programming and Reinforcement Learning*, March 2009.

[14] I. Palunko, R. Fierro, and P. Cruz, "Trajectory generation for swing-free maneuvers of a quadrotor with supended payload: A dynamic programming approach," in *IEEE International Conference on Robotics and Automation*, St. Paul, MN, USA, May 2012.

[15] R. E. Bellman, *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[16] W. B. Powell and J. Ma, "A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multi-dimensional continuous applications," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 336 – 352, 2011.

[17] J. Ma and W. B. Powell, "Convergence analysis of on-policy lspi for multi-dimensional continuous state and action space mdps and extension with orthogonal polynomial approximation," *working paper, Dept. Operations Res. Financial Eng., Princeton Univ., Princeton*, 2010.

[18] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: Safe and efficient load manipulation with aerial robots," *IEEE Robotics and Automation Magazine*, vol. 19, no. 9, pp. 69–80, September 2012.

[19] (2012, September) MARHES - simulation and experimetal videos. [Online]. Available: http://marhes.ece.unm.edu/index.php/Ipalunko:Home