Controller Design for Quadrotor UAVs using Reinforcement Learning

Haitham Bou-Ammar, Holger Voos, Wolfgang Ertel

University of Applied Sciences Ravensburg-Weingarten, Mobile Robotics Lab,
88241 Weingarten, Germany, Email: {bouammah, voos, ertel}@hs-weingarten.de

Abstract— **Quadrotor UAVs are one of the most preferred type of small unmanned aerial vehicles because of the very simple mechanical construction and propulsion principle. However, the nonlinear dynamic behavior requires a rather advanced stabilizing control of these vehicles. One possible approach that relaxes the difficult task of nonlinear control design is the application of a learning algorithm that allows the training of suitable control actions. Here we apply reinforcement learning as one form of unsupervised learning. In this paper, we first propose a nonlinear autopilot for quadrotor UAVs based on feedback linearization. This controller is then compared to an autopilot which has been learned by reinforcement learning using fitted value iteration with regard to design effort and performance. First simulation and experimental results underline the outcome of this comparison.**

1. Introduction

Unmanned aerial vehicles (UAVs) already have a wide area of possible applications. While large outdoor UAVs are already in use for military or commercial purposes, indoor flight of small UAVs is still a challenging area of application from a scientific perspective. Indoor flight requires a suitable type of vehicle as well as suitable control, navigation and collision avoidance algorithms. Concerning the vehicle type, helicopter-like vehicles are among the most promising candidates with respect to size, weight, maneuverability and the ability for slow and even hovering flight. One special helicopter-like vehicle with the additional advantage of a simple construction and rotor mechanics is the quadrotor. The quadrotor is a system with four propellers in a cross configuration, see Fig. 1 for a sketch of a quadrotor UAV. While the front and the rear motor rotate clockwise, the left and the right motor rotate counter-clockwise which nearly cancels gyroscopic effects and aerodynamic torques in trimmed flight. One additional advantage of the quadrotor compared to a conventional helicopter is the simplified rotor mechanics. By varying the speed of the single motors, the lift force can be changed and vertical and/or lateral motion can be generated. However, in spite of the four actuators, the quadrotor is a dynamically unstable nonlinear system that has to be stabilized by a suitable control system.

In this paper, we address the problem of a precise and fast stabilization of the quadrotor UAV since the fulfillment of this task is a precondition for further implementation of other functionalities. Concerning controller design for small quadrotor UAVs, some solutions are already proposed in the literature, see e.g. [1], [2], [3], [4] and [5] to mention only a few. Many of the proposed control systems are based on a linearized model and conventional PID- or state space control while other approaches apply sliding-mode [1], H_∞ or SDRE (state dependent Riccati equations) control [4], [5]. Recently, a new nonlinear control algorithm has been proposed by one of the authors which is based upon a decomposition of the overall controller into a nested structure of velocity and attitude control, see [6]. The controller has the advantage of an easy implementation and proven stability while taking the nonlinearities of the dynamics directly into account.

However, these proposed algorithms are either based on simplification and linearization leading to limited generalization with regard to the full state space or the algorithms are rather complex and require deep knowledge in nonlinear control and accurate modelling. An interesting alternative method to control a system is to learn the suitable control action, either using supervised or unsupervised learning. Since supervised learning comprises the training of already existing control actions (e.g. generated by a human operator), unsupervised learning seems to be more promising to solve more complex control problems as they arise in robotics or UAV control. In this contribution we are applying reinforcement learning (see e.g. [7]) where a simple reward function judges any generated control action. Reinforcement learning for quadrotor UAVs has already been investigated in [8], however we are investigating value iteration in this work for faster convergence of the learning process and also apply a full nonlinear model for the learning process.

The nonlinear controller as derived in [6] is first shortly introduced, followed by a short introduction in the reinforcement learning algorithm applied in this work. Finally, some first experimental as well as simulation results are presented in order to compare the two approaches.

2. Dynamic Model of the Quadrotor

The general dynamic model of a quadrotor UAV has been presented in a number of papers, see e.g. [1], [3], [4], [5] or [6], and therefore will not be discussed here in all details again. We consider an inertial frame and a body fixed frame whose origin is in the center of mass of the quadrotor, see Fig. 1. The attitude of the quadrotor is given by the roll, pitch and yaw angle, forming the vector $\boldsymbol{\Omega}^T = (\phi, \theta, \psi)$, while the position of the vehicle in the inertial frame is given by the position vector $\boldsymbol{r}^T = (x, y, z)$. The dynamic model of the quadrotor can be derived by applying the laws of conservation of momentum and angular momentum, taking the applied forces and torques into account (see [6]). The thrust force generated by rotor $i, i = 1, 2, 3, 4$ is $F_i = b \cdot \omega_i^2$ whith the thrust factor b and the rotor speed ω_i, and the law

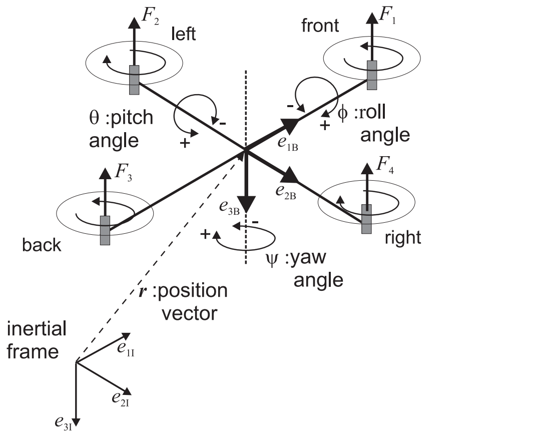

Fig. 1. Configuration, inertial and body fixed frame of the quadrotor.

of conservation of momentum yields

$$\ddot{\boldsymbol{r}} = g \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} - \boldsymbol{R}(\boldsymbol{\Omega}) \cdot b/m \sum_{i=1}^{4} \omega_i^2 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{1}$$

Herein, $\boldsymbol{R}(\boldsymbol{\Omega})$ is a suitable rotation matrix. With the inertia matrix $\boldsymbol{I}$ (a pure diagonal matrix with the inertias I_x, I_y and I_z on the main diagonal), the rotor inertia J_R, the vector $\boldsymbol{M}$ of the torque applied to the vehicle's body and the vector $\boldsymbol{M}_G$ of the gyroscopic torques of the rotors, the law of conservation of angular momentum yields:

$$\boldsymbol{I}\ddot{\boldsymbol{\Omega}} = -\left(\dot{\boldsymbol{\Omega}} \times \boldsymbol{I}\dot{\boldsymbol{\Omega}}\right) - \boldsymbol{M}_G + \boldsymbol{M} \tag{2}$$

The vector $\boldsymbol{M}$ is defined as (see Fig. 1)

$$\boldsymbol{M} = \begin{pmatrix} Lb(\omega_2^2 - \omega_4^2) \\ Lb(\omega_1^2 - \omega_3^2) \\ d(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{pmatrix} \tag{3}$$

with the drag factor d and the length L of the lever. The gyroscopic torques caused by rotations of the vehicle with rotating rotors are

$$\boldsymbol{M}_G = I_R(\dot{\boldsymbol{\Omega}} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}) \cdot (\omega_1 - \omega_2 + \omega_3 - \omega_4) \tag{4}$$

The four rotational velocities ω_i of the rotors are the real input variables of the vehicle, but for a simplification of the model, the following substitute input variables are defined:

$$\begin{aligned} u_1 &= b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ u_2 &= b(\omega_2^2 - \omega_4^2) \\ u_3 &= b(\omega_1^2 - \omega_3^2) \\ u_4 &= d(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \end{aligned} \tag{5}$$

Defining $\boldsymbol{u}^T = (u_1, u_2, u_3, u_4)$ and $(\omega_1 - \omega_2 + \omega_3 - \omega_4) = g(\boldsymbol{u})$ and introducing the vector of state variables $\boldsymbol{x}^T = (\dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi})$, evaluation of (1) until (5) yields the following state variable model:

$$\dot{\boldsymbol{x}} = \begin{pmatrix} -(\cos x_4 \sin x_5 \cos x_6 + \sin x_4 \sin x_6) \cdot u_1/m \\ -(\cos x_4 \sin x_5 \sin x_6 - \sin x_4 \cos x_6) \cdot u_1/m \\ g - (\cos x_4 \cos x_5) \cdot u_1/m \\ x_7 \\ x_8 \\ x_9 \\ x_8 x_9 I_1 - \frac{I_R}{I_x} x_8 g(\boldsymbol{u}) + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{I_R}{I_y} x_7 g(\boldsymbol{u}) + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{pmatrix} \tag{6}$$

Herein, we use the abbreviations $I_1 = (I_y - I_z)/I_x$, $I_2 = (I_z - I_x)/I_y$ and $I_3 = (I_x - I_y)/I_z$. It becomes obvious that the state variable model can be decomposed into one subset of differential equations (DEQs) that describe the dynamics of the attitude using the last six equations of (6), and one subset that describes the translation of the UAV using the first three equations of (6).

3. Nonlinear Controller Design

The task of the vehicle controller is the stabilization of a desired velocity vector which is calculated by the higher-level mission controller. The decomposed structure of the state variable model (6) already suggests a nested structure for vehicle control. In order to achieve and maintain a desired velocity vector, first the necessary attitude of the UAV has to be stabilized. Therefore, we propose a decomposition of the vehicle control system in an outer-loop velocity control and an inner-loop attitude control system. In this structure, the inner attitude control loop has to be much faster than the outer loop and stabilizes the desired angles $\boldsymbol{\Omega}_d^T = (\phi_d, \theta_d, \psi_d) = (x_{4,d}, x_{5,d}, x_{6,d})$ that are commanded by the outer loop. First we consider the inner attitude control loop, then we derive the outer-loop controller to stabilize a desired velocity vector.

A. Attitude Control System

For the design of the attitude control system we consider the last six DEQs of (6) as the relevant submodel. Herein, the last three DEQs describing x_7, x_8, x_9 are nonlinear and depend on the input variables u_2, u_3, u_4, while x_4, x_5, x_6 are obtained from the former state variables by pure integration leading to three simple linear DEQs in (6). The control strategy now is as follows: we first apply a nonlinear feedback linearization to the last three DEQs in order to transfer them into linear and decoupled DEQs. Together with the set of the remaining linear DEQs we finally obtain three independent linear systems which can be stabilized via linear feedback.

If we first neglect the gyroscopic terms (since the rotor inertias are comparatively small) we obtain the simplified DEQs for x_7, x_8, x_9 as

$$\begin{pmatrix} \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \end{pmatrix} = \begin{pmatrix} x_8 x_9 I_1 + \frac{L}{I_x} u_2 \\ x_7 x_9 I_2 + \frac{L}{I_y} u_3 \\ x_7 x_8 I_3 + \frac{1}{I_z} u_4 \end{pmatrix} \tag{7}$$

Now we apply a feedback linearization in order to obtain a linear system:

$$\begin{aligned} u_2 &= f_2(x_7, x_8, x_9) + u_2^* \\ u_3 &= f_3(x_7, x_8, x_9) + u_3^* \\ u_4 &= f_4(x_7, x_8, x_9) + u_4^* \end{aligned} \quad (8)$$

with the new input variables u_2^*, u_3^*, u_4^*. It can be shown that

$$\begin{aligned} f_2(x_7, x_8, x_9) &= \frac{I_x}{L}\left(K_2 x_7 - x_8 x_9 I_1\right) \\ f_3(x_7, x_8, x_9) &= \frac{I_y}{L}\left(K_3 x_8 - x_7 x_9 I_2\right) \\ f_4(x_7, x_8, x_9) &= I_z\left(K_4 x_9 - x_7 x_8 I_3\right) \end{aligned} \quad (9)$$

with the so far undetermined constant parameters K_2, K_3, K_4 transfer (7) into a set of linear and decoupled DEQs. It has been proven in [6] using a suitable Lyapunov function that this feedback is stable for $K_2, K_3, K_4 < 0$ even if the gyroscopic terms from (6) are considered again. Since $\dot{x}_4 = x_7$, $\dot{x}_5 = x_8$, $\dot{x}_6 = x_9$ we finally obtain linear decoupled DEQs for x_4, x_5, x_6, respectively, see e.g. x_4:

$$\ddot{x}_4 = K_2 \dot{x}_4 + L/I_x u_2^* \quad (10)$$

If x_{4d} is the desired angle, application of a linear controller $u_2^* = w_2 \cdot (x_{4d} - x_4)$ with constant parameter w_2 leads to a closed-loop system of second order

$$F(s) = \frac{X_4(s)}{X_{4d}(s)} = \frac{w_2}{I_x/L \cdot s^2 - K_2 I_x/L \cdot s + w_2} \quad (11)$$

The same considerations hold for the other angles with linear controllers $u_3^* = w_3 \cdot (x_{5d} - x_5)$ and $u_4^* = w_4 \cdot (x_{6d} - x_6)$, respectively. The dynamics of these closed-loop systems now can be easily adjusted by a choice of a suitable set of parameters $(K_2, w_2), (K_3, w_3), (K_4, w_4)$, respectively, with the only limitation that $K_2, K_3, K_4 < 0$, see [6].

B. Velocity Control System

We now assume that the previously defined inner attitude control loops are adjusted in a way that their dynamic behavior is very fast compared to the outer velocity control loops. Therefore we approximate the inner closed control loops as static blocks with transfer function $F_i(s) = X_i(s)/X_{id}(s) \approx 1, i = 4, 5, 6$. Inserting this in (6), the velocities of the quadrotor UAV then can be approximated by

$$\begin{aligned} \dot{x}_1 &= -(\cos x_{4d} \sin x_{5d} \cos x_{6d} + \sin x_{4d} \sin x_{6d}) \cdot u_1/m \\ \dot{x}_2 &= -(\cos x_{4d} \sin x_{5d} \sin x_{6d} - \sin x_{4d} \cos x_{6d}) \cdot u_1/m \\ \dot{x}_3 &= g - \cos x_{4d} \cos x_{5d} \cdot u_1/m \end{aligned} \quad (12)$$

where all x_{4d}, x_{5d}, x_{6d} and u_1 can be considered as input variables. Equation (12) can be interpreted in a way that all differential equations are of the form

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \boldsymbol{f}(x_{4d}, x_{5d}, x_{6d}, u_1) = \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} \quad (13)$$

with the new input variables $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$ that depend on the other four input variables in a nonlinear form described by the vector function $\boldsymbol{f}$. However, regarding these new input variables, the control task comprises the control of three independent first-order systems which is solved by pure proportional controllers, respectively:

$$\begin{aligned} \tilde{u}_1 &= k_1 \cdot (x_{1d} - x_1) \\ \tilde{u}_2 &= k_2 \cdot (x_{2d} - x_2) \\ \tilde{u}_3 &= k_3 \cdot (x_{3d} - x_3) \end{aligned} \quad (14)$$

Herein the controller parameters k_1, k_2 and k_3 could be chosen in a way that the outer loop is sufficiently fast but not too fast with respect to the inner loop attitude control. In a next step, these transformed input variables $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$ must be used to obtain the real input variables x_{4d}, x_{5d}, x_{6d} and u_1 by using (13). First it becomes obvious that any desired velocity vector can be achieved without any yaw rotation and therefore we can set $x_{6d} = \psi_d = 0$. Under this assumption it is shown in [6] that (13) can be solved analytically by calculating the inverse function of $\boldsymbol{f}$:

$$\begin{pmatrix} x_{4d} \\ x_{5d} \\ u_1 \end{pmatrix} = \boldsymbol{f}^{-1} \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} \quad (15)$$

C. Overall Vehicle Control System

The overall control system consist of the derived inner attitude and the outer velocity control loop. The command to the vehicle control system is a desired velocity vector given by $v_{xd} = x_{1d}, v_{yd} = x_{2d}, v_{zd} = x_{3d}$. Then, (14) is used to calculate the respective values of the variables $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$ which are transferred by static inversion (15) into the values of the desired angles x_{4d} and x_{5d} as well as the input variable u_1. As discussed, the third desired angle is set to $x_{6d} = 0$. The desired angles are used to calculate u_2^*, u_3^*, u_4^* and evaluation of (8) with the measured values of the angular rates x_7, x_8, x_9 and the nonlinear feedback yields the input variables u_2, u_3, u_4. Finally, (5) allows the calculation of the required angular rates of the rotors, namely $\omega_1, \omega_2, \omega_3$ and ω_4. The main advantage of the overall control system is the fact that the feedback linearization and the controllers are comparatively easy to be implemented, while taking the full nonlinear behavior of the vehicle into account. That leads to a fast computation even on standard embedded micro-controller systems. Further details and simulation results are also given in [6], while experimental results will be presented here.

4. Reinforcement Learning of Control Actions

A. Reinforcement Learning

An alternative approach in controlling unmanned aerial vehicles is to design a learning controller. For this work the reinforcement learning technique [7] is adapted. Reinforcement learning is a form of unsupervised learning aiming to map states into actions, so to attain the optimal policy to maximize an overall value function. In the latter framework a reward function is provided in order to deliver either negative or positive values depending on the system's state, in a goal to maximize the overall discounted payoff over the subsequent states that the system might encounter. Concerning reinforcement learning, much work also has been done on continuous state space systems using various approaches and algorithms, see e.g. [7] for an introduction, and reinforcement learning has also been applied to control a quadrotor UAV, see [8]. However, in contrast to [8] where policy iteration has been used, this work focuses on the fitted value iteration (FVI) method to approximate the value function of the quadrotor system so to design the required controller.

In order to further formalize the reinforcement learning problem, the Markov decision processes (MDPs) are employed. MDPs could be defined as a combination of:

- S: the set of states encountered by the system.
- A: the set of actions that could be generated (i.e. the actions that the four rotors could generate in the example).
- R: the reward function which maps the state-action-pair to the set of real numbers ($R : S \times A \rightarrow \mathbb{R}$) or (as in our approach) the state to the set of real numbers ($R : S \rightarrow \mathbb{R}$).
- γ: the discount factor where $\gamma \in [0,1]$.
- P_{sa}: the transition probability being at a state $s \in S$ and taking an action $a \in A$ to transient to a new state $s^{'} \in S$.

The dynamics of MDPs proceed as follows: Starting from an initial state $s_0 \in S$, and getting to choose an action $a_0 \in A$, the MDP will tend to transient to a new state s_1 drawn according to the probabilty distribution $s_1 \sim P_{s_0 a_0}$. Being at the state s_1 another action a_1 is picked which will lead to the transition to the subsequent state $s_2 \sim P_{s_1 a_1}$. The latter process is repeated until almost all the states of the MDP are explored. The above mentioned idea could be visualized as:

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \ldots$$

After visiting the sequence of states $s_0, s_1, \ldots$ with the actions $a_0, a_1, \ldots$, the total payoff is given by:

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots \tag{16}$$

The overall goal of the reinforcement learning algorithm is to choose the correct sequence of actions over time, so to maximize the expected value of (16):

$$E(R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots) \tag{17}$$

A policy $\pi : S \rightarrow A$ is defined as any function that maps the states to the actions. Moreover, a value function is defined as (17) starting at a state s_0 and executing some policy π:

$$V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots | s = s_0, \pi] \tag{18}$$

Given a fixed policy π and a discrete state MDP, (18) could be written as:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s^{'} \in S} P_{s\pi(s)}(s^{'}) V^{\pi}(s^{'}) \tag{19}$$

As mentioned earlier the main objective of the algorithm is to maximize the value function (19) over all possible policies which defines the optimal value function and optimal policy, respectively:

$$V^*(s) = \max_{\pi(s)} V^{\pi}(s) \tag{20}$$

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s^{'} \in S} P_{sa}(s^{'}) V^*(s^{'}) \tag{21}$$

Working with continuous state MDPs (as it would be the case in our application example) gives rise to the value function approximation problem. The first intuitive solution is to discretize the state space of the system so to obtain a discrete space MDP. The latter solution possess two main problems that could be summarized as:

- Curse of dimensionality, summarized by the fact that discretizing n continuous states by k steps will produce n^k discrete states, which leads to high computational time for the learning procedure in the sequel of fitting a controller.
- Naive representation of the value function whereby it will be assumed that it attains only certain constant values, which does not reflect the reality of the variation of that function.

One solution for continuous state space MDPs is the fitted value iteration algorithm which will therefore be explained in more detail in the following section.

B. Fitted Value Iteration (FVI)

In our application example the system is assumed to have continuous states while the action space A is small and discrete, as is the case in many real life applications. The main idea lies behind approximating the optimal value function, since in the latter case (19) is no longer valid. Rather the expected value of the total payoff is represented by an integral as follows:

$$V(s) = R(s) + \gamma \max_a \int_{s^{'}} P_{sa}(s^{'}) V(s^{'}) ds^{'} \tag{22}$$

$$V(s) = R(s) + \gamma \max_a E_{s^{'} \sim P_{sa}}[V(s^{'})] \tag{23}$$

The main idea of the algorithm is to carry over a finite number of states $s^1, s^2, \ldots, s^m$ an approximation of (23). Specifically, a supervised learning algorithm will be used where by the value function will be assumed as some nonlinear function of the states:

$$V(s) = \Theta^T \Phi(s) \tag{24}$$

For every state in the finite samples the algorithm will tend to compute a function y^i which will be an approximation of (23) and then will try to minimize the sum of the least square errors between the y^i's and $V(s)$ in (24).

C. FVI Applied to the Quadrotor

Reinforcement learning with fitted value iteration is now applied to the model (6) of the quadrotor. First, the reward function is chosen as:

$$R(S, S_{ref}) = -c_1(\phi - \phi_{ref})^2 - c_2(\theta - \theta_{ref})^2 - c_3(\psi - \psi_{ref})^2 \tag{25}$$

where $R : \mathbb{R}^{3n_s} \to \mathbb{R}$ is the reward function, $c_1 > 0, c_2 > 0, c_3 > 0$ are constants giving the ability to focus especially on the control of one of the Euler angles rather than the other, and $S_{ref}^T = [\phi_{ref}, \theta_{ref}, \psi_{ref}]$ is the reference state defining the hovering position. Next a parametric approximation of the value function is chosen incorporating quadratic and coupling terms:

$$V(s) = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \end{pmatrix} \cdot \begin{pmatrix} \phi^2 \\ \theta^2 \\ \psi^2 \\ \phi\psi \\ \theta\psi \end{pmatrix} \quad (26)$$

These variables were selected due to their importance as obtained from the theoretical considerations. After randomly sampling 1000 states of the system using the nonlinear model (6) of the quadrotor, the main objective of the algorithm is to adapt the constants $(a_1, a_2, \dots, a_5)$ so to have the best approximation of the value function. In order to obtain the latter constants the normal equations were solved using the *(*QR) factorization technique in computing the pseudo inverse of the design matrix $\boldsymbol{A}$ which had been chosen to satisfy the above mentioned constraints.

$$\boldsymbol{A} = \begin{pmatrix} \phi(i)^2 & \theta(i)^2 & \psi(i)^2 & \phi(i)\psi(i) & \theta(i)\psi(i) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi(m)^2 & \theta(m)^2 & \psi(m)^2 & \phi(m)\theta(m) & \theta(m)\psi(m) \end{pmatrix} \quad (27)$$

where i and m represent the number of the training example that varies from 1 to the number of samples. Then the values of the constants could be obtained using the *QR* factorization. For further clarification we represent a pseudo code of the algorithm: The controller that is finally obtained

Algorithm 1 FVI on the quadrotor

```
Generate $S_0$ random initial states
Initialize $a_1 \dots a_5$ to zeros
repeat
  for $i = 1$ to $length(S_0)$ do
    for each $a \in A$ do
      $q(a)=R(s^{(i)})+\gamma V(s')$
    end for
    $y^{(i)}=\max_a q(a)$
  end for
  Generate the Matrix $\boldsymbol{A}$ using (27)
  Update the $a$'s by minimizing the least square error
  between $y^{(i)}$'s and (26).
until $a$'s don't change within a small tolerance
```

by reinforcement learning has the same structure as the previously presented controller. This is due to the fact that a feedback of the three Euler angles and the angular velocities as states are needed and the agent tries to minimize the error between these variables and their reference values. On the contrary, in the learning approach the controller tries the actions and learns to map them to the vital states so to maximize a value function, rather than having the pre choice of the controller which tries to perform the required task.

5. Simulation and Experimental Results

In order to evaluate the derived control systems, an experimental prototype of the quadrotor has been designed and the dynamic model (6) of this quadrotor has been derived by identification of the system parameters like inertias, dimensions etc., see also [6] for a more detailed description. This dynamic model then has been implemented in MATLAB/SIMULINK for the simulative evaluation and comparison of the two proposed control system, i.e. the nonlinear controller based on feedback linearization (called NFL-controller) and the controller obtained by reinforcement learning (called RL-controller).

In a first simulation of the results obtained with the NFL-controller, we assume an initial deviation of the angles $\boldsymbol{\Omega}^T = (\phi = 30°, \theta = -20°, \psi = 10°)$ where the control goal is to stabilize a hovering position, i.e. $\boldsymbol{v}_d = \boldsymbol{0}$. The obtained control result is shown in Fig. 2 as a time plot of all angles of the quadrotor. There is a very short transition phase with small under- and overshoots and the hovering state is reached after $t \approx 0.6$ sec. Some more simulation results of the NFL-controller are also presented in [6], therefore we present here some results obtained from first experimental test flights with the quadrotor prototype. In the experiment the control goal again was the stabilization of a hovering state starting from any initial position and compensating for any external disturbances. The obtained experimental result is shown in Fig. 3 as a time plot of all angles of the quadrotor. After a very short transition phase the hovering state is reached and maintained. The small constant deviation of the yaw angle results from a slight misalignment of the inertial measurement unit. It becomes obvious from Fig. 3 that external disturbances at 35 seconds of the roll angle, at 45 seconds of the pitch angle and at 50 seconds at the yaw angle are completely compensated. The results prove that the NFL-controller leads to a very satisfactory control result even in the experimental prototype, while the implementation is easy and the computational effort can

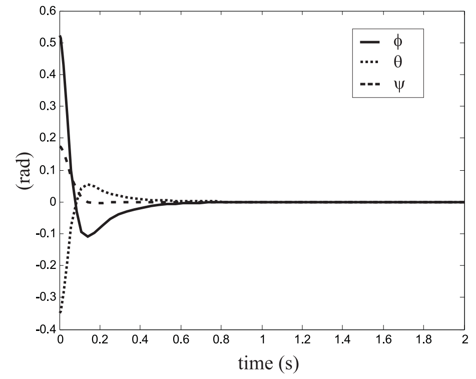

Fig. 2. Simulation results of the NFL-controller.

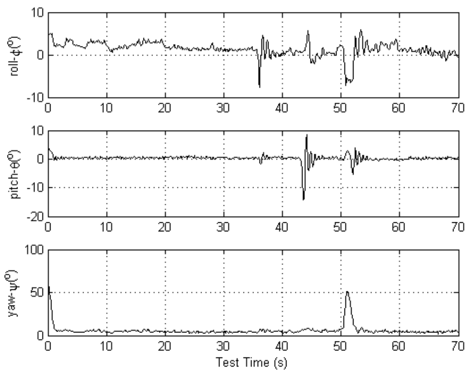

Fig. 3. Experimental results of the NFL-controller.

be handled by a microcontroller. The RL-controller is only tested in simulations so far, however implementation in the experimental platform is currently done. For a comparison, a similar situation as previously explained was assumed: the quadrotor starts at some initial deviations of the angles while the control goal is to achieve and maintain a hovering state. The results obtained with the RL-controller are depicted in Fig. 4, giving the results of the angles as well the angular velocities during compensation. It becomes obvious that also the learned control algorithm is able to stabilize the hovering state, compared to the results of the NFL-controller, it requires a slightly higher settling time (about 1 sec), but the overshoots during compensation are lower. However, there are also small control action in the hovering state. This under performance of the RL-controller could be related to the parametric approximation of the value function that had been chosen, whereby processing a non-parametric approach to approximate the value function would perform better especially with highly nonlinear and real states MDPs. However, the biggest advantage of the RL-controller is the fact that no deeper knowledge in nonlinear control engineering would be required. As a further improvement it is planned to learn also the dynamic model of the quadrotor from experiments which would also relax the modelling task dramatically.

6. Conclusion and Future Works

This paper presents a comparison between two control approaches for a quadrotor UAV. First, a nonlinear controller based on feedback linearization and a cascaded structure has been proposed. Second, a control algorithm has been learned using reinforcement learning and fitted value iteration using the nonlinear dynamic model of the quadrotor. Both control engineering approaches result in a satisfying control result. The performance of the nonlinear controller is better but requires more detailed knowledge in control engineering. One of the advantages of a learning algorithm is the fact that no prior mathematical knowledge of the model is required to design a controller. This reflects that a model of the quadrotor could be approximated using different techniques by just relating the input and output data through a non parametric approach. This idea will be tested in the near future. Further

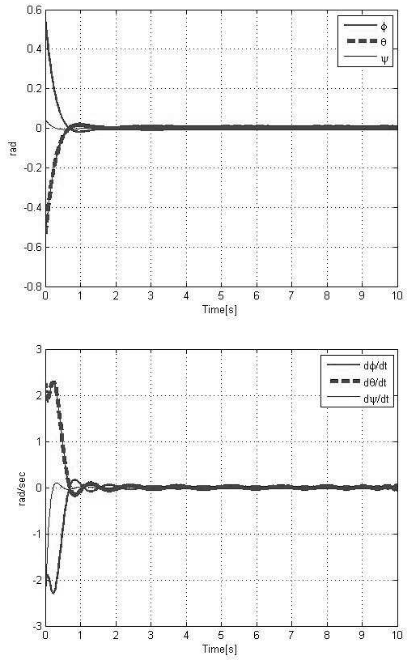

Fig. 4. Control results of the RL-controller.

improvements on the fitted value iteration algorithm could be added such as using a non parametric approximation technique like a wavelet network, see e.g. [9], which seems to be promising. In the ongoing work further experiments will be conducted with different parametric value functions, and the implementation of such wavelet networks will be further studied and tested to orient it towards the quadrotor problem.

References

[1] S. Bouabdallah, R. Siegwart. Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. In Proc. of the IEEE International Conference on Robotics and Automation, 2005, pp. 2247 2252.

[2] A. Tayebi, S. McGilvray. Attitude Stabilization of a VTOL Quadrotor Aircraft. In IEEE Trans. on Control Systems Technology, 2006, Vol. 14, 2006, pp. 562 - 571.

[3] P. Castillo, A. Dzul, R. Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. IEEE Trans. on Control Systems Technology, Vol.12, No. 4, July 2004, pp. 510 - 516.

[4] H. Voos. Nonlinear State-Dependent Riccati Equation Control of a Quadrotor UAV. In Proc. of the IEEE Conference on Control Applications, Munich, 2006.

[5] H. Voos. Nonlinear and Neural Network-based Control of a Small Four-Rotor Aerial Robot. In Proc. of the IEEE/ASME Int. Conference on Advanced Intelligent Mechatronics, Zurich, CH, 2007.

[6] H. Voos. Nonlinear Control of a Quadrotor Micro-UAV using Feedback-Linearization. In Proc. of the IEEE International Conference on Mechatronics (ICM 2009), Málaga, Spain, 14-17 April, 2009.

[7] Sutten, R. S. and Barto, A. G. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.

[8] S.L. Waslander et. al. Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning. In Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2005), 2005, pp. 3712-3717.

[9] Z. Zhang. Learning Algorithm of Wavelet Network Based on Sampling Theory. In Neurocomputing /1 (2007), Elsevier, pp. 244-269.

[10] R. Munos, Szepesvari, C. Finite-Time Bounds for Fitted Value Iteration. Journal of Machine Learning Research 1 (2008), pp. 815-857.