Nonlinear and Neural Network-based Control of a Small Four-Rotor Aerial Robot

Holger Voos

Abstract— **Small four-rotor aerial robots, so called quadrotor UAVs, have an enormous potential for all kind of near-area surveillance and exploration in military and commercial applications. In addition, they offer the possibility to fly either in- or outdoor. However, stabilizing control and guidance of these vehicles is a difficult task because of the nonlinear dynamic behavior. This paper describes the development of a nonlinear vehicle control system based on a combination of state-dependent Riccati equations (SDRE) and neural networks. Some first simulation results underline the performance of this new control approach for the current realization.**

I. INTRODUCTION

Unmanned flying robots or vehicles (UAVs) are gaining increasing interest because of a wide area of possible applications. While the UAV market has first been driven by military applications and large expensive UAVs, recent results in miniaturization, mechatronics and microelectronics also offer an enormous potential for small and inexpensive UAVs for commercial use. These small UAVs would be able to fly either in- or outdoor, leading to completely new applications. However, indoor flight comes up with some very challenging requirements in terms of size, weight and maneuverability of the vehicle that rule out most of the aircraft types, see [2] for an excellent overview. One type of aerial vehicle with a strong potential also for indoor flight is the rotorcraft and the special class of four-rotor aerial vehciles, also called quadrotor. This vehicle, shown in Fig. 1, has been chosen by many researchers as a very promising vehicle, see e.g. [1], [2], [3] and [4].

The quadrotor is a mechatronic system with four propellers in a cross configuration. 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 created. Pitch movement is generated by a difference between the speed of the front and the rear motor while roll movement results from differences between the speed of the left and right rotor, respectively. Yaw rotation results from the difference in the counter-torque between each pair (front-rear and left-right) of rotors. The overall thrust is the sum of the thrusts generated by the four single rotors. Besides this special mechanical construction,

H. Voos is with Faculty of Electrical Engineering and Computer Science, University of Applied Sciences Ravensburg-Weingarten, D-88241 Weingarten, P.O.-Box 1261, Germany, voos@hs-weingarten.de

Fig. 1. A commercially available quadrotor.

all sensors and information processing units are embedded in the vehicle for control purposes in order to operate the UAV autonomously.

However, in spite of the four actuators, the quadrotor is a dynamically unstable system that has to be stabilized by a suitable control system. Unfortunately, the dynamic behavior is nonlinear leading to more complex control algorithms. In addition to this functional complexity, the algorithms also have to be implemented in the embedded hardware and have to fulfil realtime requirements while limited memory and processing onboard capacity have to be considered. There are some contributions in the literature that are concerned with control system design for quadrotor vehicles, see e.g. [1], [2], [3] and [4] 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 or H_∞ control. In [1], a nonlinear controller based on state-dependent Riccati equations (SDRE) has been proposed. The basic idea of the SDRE-approach was developed by [6] and applied to a number of control problems also in aerospace applications, see e.g. [6], [9], but not yet to the control of a small quadrotor UAV. The approach in [1] is further improved here and extended by the suitable addition of neural networks. First simulation results are promising for the current realization and implementation of the algorithms.

II. DYNAMIC MODEL OF THE QUADROTOR

The general dynamic model of a quadrotor has been presented in a number of papers and will not be discussed here

in all details again. For further considerations of modelling, we refer to [1], [2] and [4]. We consider an inertial frame and a body fixed frame whose origin is in the center of mass of the quadrotor as shown in Fig. 2.

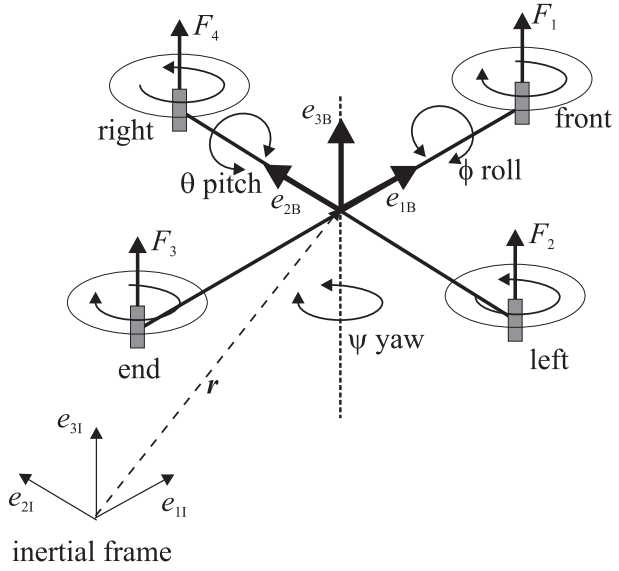

Fig. 2. Configuration, inertial and body fixed frame of the quadrotor.

The orientation of the quadrotor is given by the three Euler angles, namely yaw angle ψ, pitch angle θ and roll angle ϕ that together form the vector $\mathbf{\Omega}^T = (\phi, \theta, \psi)$. The position of the vehicle in the inertial frame is given by the vector $\boldsymbol{r}^T = (x, y, z)$. The transformation of vectors from the body fixed frame to the inertial frame is given by the rotation matrix $\boldsymbol{R}$ where c_θ for example denotes $\cos\theta$ and s_θ denotes $\sin\theta$:

$$\boldsymbol{R} = \begin{pmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix} \quad (1)$$

Since the thrust force generated by rotor $i, i = 1, 2, 3, 4$ is $F_i = b \cdot \omega_i^2$ where b is the thrust factor and ω_i is the speed of rotor i, we obtain a first set of differential equations that describe the acceleration of the quadrotor:

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

With the inertia matrix $\boldsymbol{I}$ (which is a diagonal matrix with the inertias I_x, I_y and I_z on the main diagonal), the rotor inertia J_R and the vector $\boldsymbol{\tau}$ that describes the torque applied to the vehicle's body we obtain a second set of differential equations:

$$\boldsymbol{I}\ddot{\mathbf{\Omega}} = -\dot{\mathbf{\Omega}} \times \boldsymbol{I}\dot{\mathbf{\Omega}} - \sum_{i=1}^{4} J_R(\dot{\mathbf{\Omega}} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix})\omega_i + \boldsymbol{\tau} \quad (3)$$

The vector $\boldsymbol{\tau}$ is defined as

$$\boldsymbol{\tau} = \begin{pmatrix} lb(\omega_4^2 - \omega_2^2) \\ lb(\omega_3^2 - \omega_1^2) \\ d(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2) \end{pmatrix} \quad (4)$$

with the drag factor d and the length l of the lever. The four rotational velocities ω_i of the rotors are the input variables of the real vehicle, but with regard to the obtained model a transformation of the inputs is suitable. Therefore, we obtain the new artificial input variables as follows:

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

However, in the previous equations we obtain an additional variable that also depends on the rotational speeds of the rotors and therefore must be considered as a fifth artificial input:

$$u_5 = \omega_d = \omega_2 + \omega_4 - \omega_1 - \omega_3 \quad (6)$$

Evaluation of (2) and (3) thus yields the overall dynamic model in the following form

$$\begin{aligned} \ddot{x} &= (\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \cdot u_1/m \\ \ddot{y} &= (\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi) \cdot u_1/m \\ \ddot{z} &= -g + (\cos\phi \cos\theta) \cdot u_1/m \\ \ddot{\phi} &= \dot{\theta}\dot{\psi}(\frac{I_y - I_z}{I_x}) - \frac{J_R}{I_x}\dot{\theta}u_5 + \frac{l}{I_x}u_2 \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}(\frac{I_z - I_x}{I_y}) + \frac{J_R}{I_y}\dot{\phi}u_5 + \frac{l}{I_y}u_3 \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}(\frac{I_x - I_y}{I_z}) + \frac{l}{I_z}u_4 \end{aligned} \quad (7)$$

This model can be rewritten in state-space form $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ where $\boldsymbol{u}^T = (u_1, u_2, u_3, u_4, u_5)$ is the vector of (artificial) input variables given in (5) and (6) and $\boldsymbol{x} \in \mathbb{R}^{12}$ is the vector of state variables given as follows:

$$\boldsymbol{x}^T = (x, \dot{x}, y, \dot{y}, z, \dot{z}, \phi, \dot{\phi}, \theta, \dot{\theta}, \psi, \dot{\psi})^T \quad (8)$$

From (7) and (8) we obtain

$$\dot{\boldsymbol{x}} = \begin{pmatrix} x_2 \\ (\cos x_7 \sin x_9 \cos x_{11} + \sin x_7 \sin x_{11}) \cdot u_1/m \\ x_4 \\ (\cos x_7 \sin x_9 \sin x_{11} - \sin x_7 \cos x_{11}) \cdot u_1/m \\ x_6 \\ -g + (\cos x_7 \cos x_9) \cdot u_1/m \\ x_8 \\ x_{12}x_{10}I_1 - \frac{J_R}{I_x}x_{10}u_5 + \frac{l}{I_x}u_2 \\ x_{10} \\ x_{12}x_8 I_2 + \frac{J_R}{I_y}x_8 u_5 + \frac{l}{I_y}u_3 \\ x_{12} \\ x_{10}x_8 I_3 + \frac{l}{I_z}u_4 \end{pmatrix} \quad (9)$$

with 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 space model can be decomposed into one subset of differential equations that describes the dynamics of the attitude (i.e. the angles) and one subset that describes the translation of the UAV. From

(9) we obtain the first subset of differential equations, called submodel M_1 as

$$\begin{pmatrix} \dot{x_8} \\ \dot{x}_{10} \\ \dot{x}_{12} \end{pmatrix} = \begin{pmatrix} x_{12}x_{10}I_1 - \frac{J_R}{I_x}x_{10}u_5 + \frac{l}{I_x}u_2 \\ x_{12}x_8I_2 + \frac{J_R}{I_y}x_8u_5 + \frac{l}{I_y}u_3 \\ x_{10}x_8I_3 + \frac{l}{I_z}u_4 \end{pmatrix} \quad (10)$$

The artificial input variables of that first submodel are the variables u_2, u_3, u_4 and u_5. From the angular rates as the output of M_1 the angles are obtained by pure integration. The three angles (or state variables x_7, x_9 and x_{11}) are the inputs of the next submodel M_2 which is given by

$$\begin{pmatrix} \dot{x_2} \\ \dot{x}_4 \\ \dot{x}_6 \end{pmatrix} = \begin{pmatrix} (\cos x_7 \sin x_9 \cos x_{11} + \sin x_7 \sin x_{11})\frac{u_1}{m} \\ (\cos x_7 \sin x_9 \sin x_{11} - \sin x_7 \cos x_{11})\frac{u_1}{m} \\ -g + (\cos x_7 \cos x_9)\frac{u_1}{m} \end{pmatrix} \quad (11)$$

Herein, the controllable artificial input is the variable u_1. The resulting structure of the quadrotor model is shown in Fig. 3.

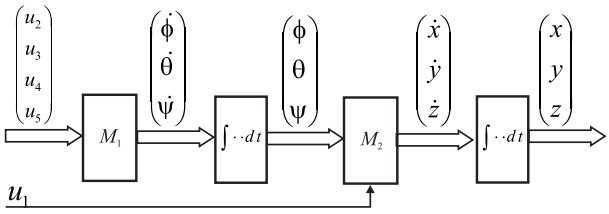

Fig. 3. Decomposed structure of the quadrotor model.

III. VEHICLE CONTROLLER DESIGN

From a control engineering point of view, a UAV system contains two main control loops [5]. The first main and underlying control loop is the vehicle control loop. This control loop is responsible for the generation and stabilization of a currently required movement of the UAV. The second main loop is the mission control loop that comprises the stabilized vehicle as a platform for mission related sensors and actuators and the mission control system. The mission control loop computes the desired flight path, e.g. given by GPS waypoints, and commands current required movements to the vehicle control loop. The question remains what kind of commands will be given to the vehicle control loop. Direct position control as proposed in many papers (see e.g. [3], [4]) is most often not necessary for vehicle guidance and position measurement or estimation is most often not accurate enough for direct feedback control.

For that reason we assume in this approach that the mission control system commands a desired velocity vector to the vehicle control system. This required velocity vector then has to be established and stabilized. In order to obtain the necessary measurements for this velocity control, the vehicle control loop must be equipped with a suitable inertial measurement system (IMU). This IMU delivers the accelerations and angular rates that can be used to further estimate velocities and Euler angles with the help of a Kalman filter. The default command from the mission system is the zero velocity vector, i.e. the quadrotor UAV should hover at the current position. In this paper the main challenge and focus is on the vehicle control loop, i.e. the control of a required velocity vector of the UAV.

The decomposed model structure as shown in Fig. 2 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 control system in an outer-loop velocity control and an inner-loop attitude control. In this structure, the inner attitude control loop has to be much faster than the outer loop and stabilizes the desired angles that are commanded by the outer loop. This nested structure is shown in Fig. 4.

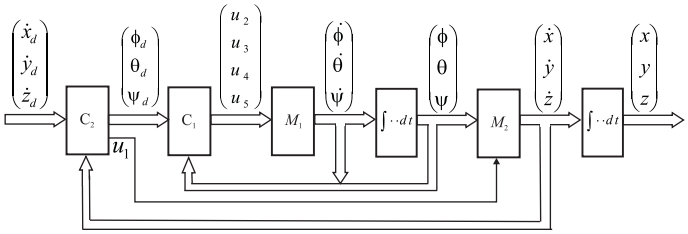

Fig. 4. Nested structure of the UAV vehicle control.

First we consider the inner control loop with controller C_1, the attitude control loop, that has to stabilize the desired roll, pitch and yaw angle, i.e. the desired vector $\boldsymbol{\Omega}_d^T = (\phi_d, \theta_d, \psi_d) = (x_{7,d}, x_{9,d}, x_{11,d})$. The corresponding dynamic model comprises the last six equations of the state space model (8) which is a series of the nonlinear submodel M_1 and an integrator. Then we derive the outer-loop controller C_2 to stabilize a desired velocity vector.

A. State-dependent Riccati Equation Control

The state-dependent Riccati equation (SDRE) control was initially derived by Cloutier, see [6] for an overview. The basic idea was motivated by linear quadratic regulation and introduces a factorization of a nonlinear system in a way that it becomes linear at any fixed state

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \quad (12)$$

where the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ both depends on the current state variables. The controllability issues of such methods are discussed in [6]. Control gains at any state $\boldsymbol{x}$ can be calculated using standard linear optimal control theory, i.e. choosing that control that minimizes the cost function

$$J = 0.5 \int_{t_0}^{\infty} \boldsymbol{x}^T \boldsymbol{Q}(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R}(\boldsymbol{x})\boldsymbol{u} dt \quad (13)$$

where $\boldsymbol{Q}(\boldsymbol{x})$ penalizes the state and $\boldsymbol{R}(\boldsymbol{x})$ penalizes control effort. By solving the algebraic Riccati equation

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} = \boldsymbol{0} \quad (14)$$

we obtain the matrix $\boldsymbol{P}(\boldsymbol{x})$ and the control gains become

$$\boldsymbol{u} = -\boldsymbol{K}(\boldsymbol{x})\boldsymbol{x} = -\boldsymbol{R}(\boldsymbol{x})^{-1}\boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}(\boldsymbol{x})\boldsymbol{x} \quad (15)$$

In general, this technique requires that the algebraic Riccati equation must be solved at every state and therefore also the control gains have to be recalculated at every state. This seems to be computationally complex, but [9] developed some real-time methods that can be implemented on an embedded micro-controller. In addition, stability of the SDRE approach was shown in [7], [8].

B. Attitude Control using SDRE

We apply the SDRE method to the attitude control problem. The vector of state variables for that problem is given by $\boldsymbol{x}_I^T = (x_7, x_8, x_9, x_{10}, x_{11}, x_{12})$ while the vector of artificial input variables is $\boldsymbol{u}_I^T = (u_2, u_3, u_4, u_5)$ and one possible state-dependent model can be obtained from (9) by factorization as

$$\dot{\boldsymbol{x}}_I = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{12}I_1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & x_{12}I_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & x_8 I_3 & 0 & 0 \end{pmatrix} \cdot \boldsymbol{x}_I + \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{l}{I_x} & 0 & 0 & -\frac{J_R}{I_x}x_{10} \\ 0 & 0 & 0 & 0 \\ 0 & \frac{l}{I_y} & 0 & \frac{J_R}{I_y}x_8 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{l}{I_z} & 0 \end{pmatrix} \cdot \boldsymbol{u}_I \tag{16}$$

Please note that this factorization is not unique as also discussed in [6]. In (16) both matrix $\boldsymbol{A}(\boldsymbol{x}_I)$ and $\boldsymbol{B}(\boldsymbol{x}_I)$ are state-dependent, i.e. (16) can be written as

$$\dot{\boldsymbol{x}_I} = \boldsymbol{A}(\boldsymbol{x}_I)\boldsymbol{x}_I + \boldsymbol{B}(\boldsymbol{x}_I)\boldsymbol{u}_I \tag{17}$$

Using (17), the control gain matrix $\boldsymbol{K}(\boldsymbol{x}_I)$ can be calculated, while the overall control input $\boldsymbol{u}_I$ must also take into account that a desired state $\boldsymbol{x}_{I,d}$ given by the outer velocity control loop must be stabilized. The stabilization of a desired state which is not zero can be guaranteed by a pre-filter matrix $\boldsymbol{M}(\boldsymbol{x}_I)$ assuming that the control gain matrix $\boldsymbol{K}(\boldsymbol{x}_I)$ is already determined:

$$\boldsymbol{M}(\boldsymbol{x}_I) = \mathrm{pinv}\big((\boldsymbol{B}(\boldsymbol{x}_I)\boldsymbol{K}(\boldsymbol{x}_I) - \boldsymbol{A}(\boldsymbol{x}_I))^{-1}\boldsymbol{B}(\boldsymbol{x}_I)\big) \tag{18}$$

where pinv() denotes the pseudo-inverse of a non-quadratic matrix. The overall attitude control law, i.e. the controller C_1 can then be summarized as follows:

$$\boldsymbol{u}_I = -\boldsymbol{K}(\boldsymbol{x}_I)\boldsymbol{x}_I + \boldsymbol{M}(\boldsymbol{x}_I)\boldsymbol{x}_{I,d} \tag{19}$$

C. Velocity Control using a Neural Network

If the inner-loop attitude control is sufficiently fast, we can assume that a desired value of the roll, pitch and yaw angle is achieved very fast compared with the outer velocity control loop. Therefore the closed inner control loop can be approximately considered as a static block that just transfers the desired values of roll, pitch and yaw angle to the next model M_2. According to (11), we can describe model M_2 by the following set of nonlinear differential equations:

$$\begin{aligned} \dot{x}_2 &= (\cos x_{7d} \sin x_{9d} \cos x_{11d} + \sin x_{7d} \sin x_{11d}) \cdot u_1/m \\ \dot{x}_4 &= (\cos x_{7d} \sin x_{9d} \sin x_{11d} - \sin x_{7d} \cos x_{11d}) \cdot u_1/m \\ \dot{x}_6 &= \cos x_{7d} \cos x_{9d} \cdot u_1/m - g \end{aligned} \tag{20}$$

where all x_{7d}, x_{9d}, x_{11d} and u_1 are input variables. Equation (20) can be interpreted in a way that all differential equations have the form

$$\begin{aligned} \dot{x}_2 &= \tilde{u}_1 = f_1(x_{7d}, x_{9d}, x_{11d}, u_1) \\ \dot{x}_4 &= \tilde{u}_2 = f_2(x_{7d}, x_{9d}, x_{11d}, u_1) \\ \dot{x}_6 &= \tilde{u}_3 = f_3(x_{7d}, x_{9d}, x_{11d}, u_1) \end{aligned} \tag{21}$$

with the new input variables $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$ that depend on the other four input variables in a very complex nonlinear form. However, concerning the new input variables, the control task is very simple since it comprises the control of three independent systems of first order which might be solved by a pure proportional controller, respectively:

$$\begin{aligned} \tilde{u}_1 &= k_1 \cdot (x_{2d} - x_2) \\ \tilde{u}_2 &= k_2 \cdot (x_{4d} - x_4) \\ \tilde{u}_3 &= k_3 \cdot (x_{6d} - x_6) \end{aligned} \tag{22}$$

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_{7d}, x_{9d}, x_{11d} and u_1 by evaluating (20).

(21) can be rewritten in the more compact form

$$\begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix} = \boldsymbol{f}\left(\begin{pmatrix} x_{7d} \\ x_{9d} \\ x_{11d} \\ u_1 \end{pmatrix}\right) \tag{23}$$

where $\boldsymbol{f}$ is a vector function given by (20). In order to obtain the input variables x_{7d}, x_{9d}, x_{11d} and u_1 for any given values of $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$, we have to calculate the inverse of the vector function $\boldsymbol{f}$, i.e.

$$\begin{pmatrix} x_{7d} \\ x_{9d} \\ x_{11d} \\ u_1 \end{pmatrix} = \boldsymbol{f}^{-1}\left(\begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \end{pmatrix}\right) \tag{24}$$

Because of the complexity of the vector function $\boldsymbol{f}$ it is not possible to obtain the inverse function as a closed-form mathematical expression. For that reason, the inverse $\boldsymbol{f}^{-1}$ is realized here with the help of a neural network. An additional consideration can help to simplify the training of the neural network: in order to achieve any desired velocity vector, it is not necessary to apply a yaw rotation and therefore we can set $x_{11d} = \psi_d = 0$.

The neural network is developed with the help of the Neural Network Toolbox of Matlab. First, a suitable set of training data must be available. For that reason, first a

certain number of random values of x_{7d}, x_{9d}, u_1 is generated and the corresponding values of the variables $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$ are calculated with the help of (20). Hereby, the range of the random variables x_{7d}, x_{9d}, u_1 must be chosen very carefully in order to cover the whole range of possible values in the later implementation. In order to obtain the inverse function $\boldsymbol{f}^{-1}$, this set of values of the calculated variables $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3$ is then taken as input for the neural network and the corresponding outputs are the variables x_{7d}, x_{9d}, u_1.

In the next step, a suitable neural network type, internal structure and learning algorithm had been chosen. Here, a multilayer perceptron as a feedforward neural network is used. A number of 20 neurons in the input layer, 20 neurons in the hidden layer and 4 neurons in the output layer led to a satisfactory result. For learning, the Levenberg-Marquardt back-propagation algorithm has been chosen. Experiments led to the conclusion that a number of 1000 input and output data sets was sufficient for the training of the chosen neural network which was able to realize the required inverse function $\boldsymbol{f}^{-1}$. The following Fig. 5 gives an example of the first 100 data points of the variable x_{7d} for the respective input variables and the corresponding values that are calculated by the neural network.

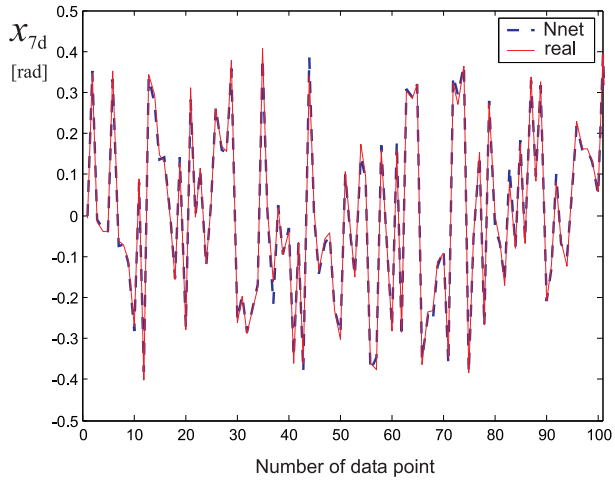

Fig. 5. Example results of the trained neural network.

D. Calculation of the real input variables using a Neural Network

The inner attitude controller C_1 and the outer velocity controller C_2 together generate the artificial control vector $\boldsymbol{u}^T = (u_1, u_2, u_3, u_4, u_5)$. However, as already mentioned, these input variables cannot be applied to the real quadrotor UAV, since the real input variables are the rotational speeds ω_i of the four rotors $i = 1, \ldots, 4$, respectively. Therefore, these four values must be calculated using (5) and (6). Again, we have a set of five nonlinear equations that must be solved to calculate four unknowns, i.e. the values of ω_i. Since there is no closed-form solution, we again apply a neural network in order to perform this calculation.

As described in the previous section, a set of suitable training data had been created first using random values of the rotational speeds ω_i and the corresponding calculated values of $\boldsymbol{u}^T = (u_1, u_2, u_3, u_4, u_5)$ using the equations (5) and (6). For the training, the data set of these calculated values of $\boldsymbol{u}^T$ are then taken as input variables while the corresponding values of $\omega_i, i = 1, \ldots, 4$ are the output variables that have to be learned. Again, a multilayer perceptron was chosen with the Levenberg-Marquardt back-propagation algorithm of the Matlab Neural Network Toolbox. The neural network was structured with 20 neurons in the input, 20 neurons in the hidden and four neurons in the output layer. Here, a number of 4000 data sets has been used for training in order to get a sufficiently accurate result of the calculation. The resulting neural network was then implemented in the control system to transform the artificial input variables $\boldsymbol{u}^T = (u_1, u_2, u_3, u_4, u_5)$ into the real input variables $\omega_i, i = 1, \ldots, 4$. That leads to the overall vehicle control structure shown in Fig. 6.

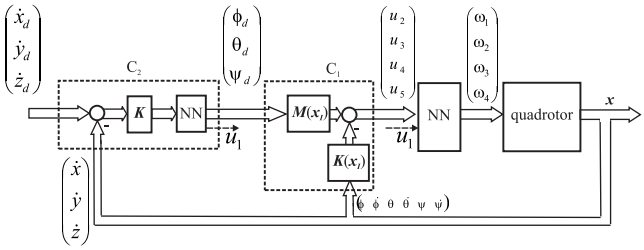

Fig. 6. Overall vehicle control structure.

IV. SIMULATION RESULTS

The quadrotor model and the derived control algorithms have been implemented in Matlab/Simulink for a simulation. For that purpose, the parameters of a real quadrotor had been identified and inserted in the simulation model. The quadrotor model was implemented as given in (9). In a first simulation, we assume an initial velocity $\dot{x} = 0.8\mathrm{m/sec}$, the other two velocities are both zero. The desired state which has to be achieved by the control action is the hovering state where all angles and all velocities are zero. The simulation result for the velocities is shown in Fig. 7 as a time plot. It becomes obvious that after a short transition phase all velocities are stabilized at the required value of zero. During that compensation of the initial disturbances of the velocities, the quadrotor changes the position until a new hovering position is reached and stabilized.

In a next simulation, we assume that the quadrotor starts at an initial position $x = y = z = 3\mathrm{m}$ in the hovering state (i.e. all velocities are zero). The task now is to achieve and stabilize a velocity vector with $\dot{x}_d = \dot{y}_d = \dot{z}_d = 0.5\ \mathrm{m/sec}$ and to generate a linear movement. The time plot of all angular rates during the control action is presented in Fig. 8 while the position in three dimensions is shown in Fig. 9. Again, the desired state is achieved after a short transition phase and the quadrotor is moving with constant velocity. During that constant flight the angles are also kept constant and hence the angular rates are all zero after the initial transition.

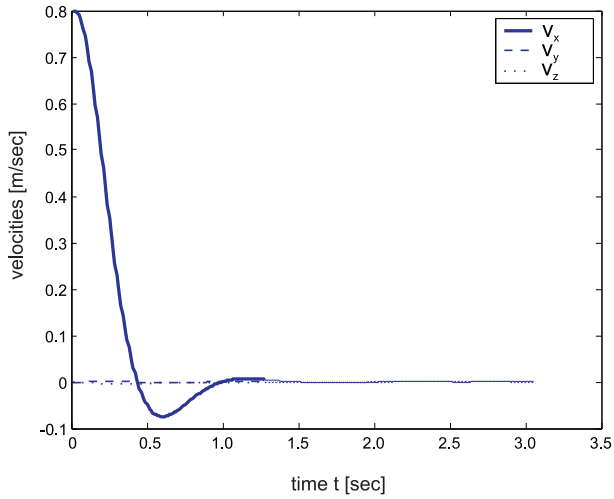

Fig. 7. Time plot of the velocities.

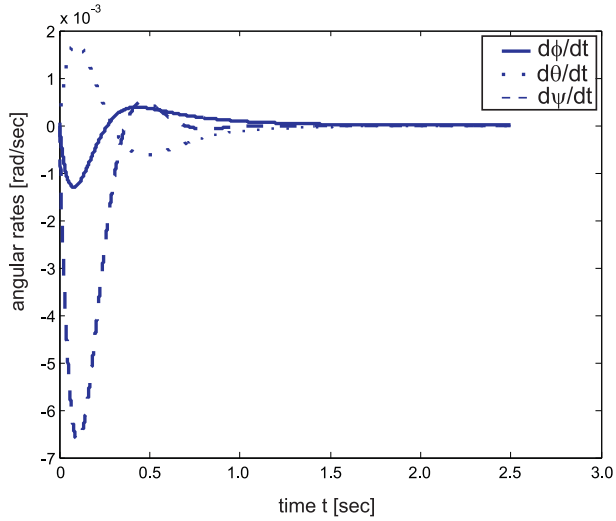

Fig. 8. Time plot of angular rates during control.

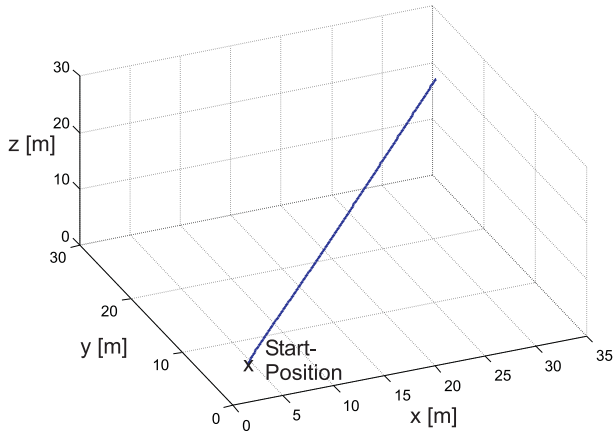

Fig. 9. Plot of position during control.

V. CONCLUSIONS AND FUTURE WORKS

This paper presents a vehicle control system for a small quadrotor UAV based on a combined control strategy including state-dependent Riccati equation (SDRE) control as well as neural networks. Both an inner-loop attitude controller and an outer-loop velocity controller have been developed during the proposed work. The dynamic model of the quadrotor is derived and implemented in a Matlab/Simulink simulation model. With the help of that simulation, the nonlinear vehicle control system is tested and its efficiency demonstrated. In our ongoing work we are currently implementing the proposed control system in the real quadrotor UAV.

REFERENCES

[1] H. Voos, "Nonlinear State-Dependent Riccati Equation Control of a Quadrotor UAV", *in Proc. of the IEEE Conference on Control Applications*, Munich, Germany, 2006.

[2] S. Bouabdallah, P. Murrieri, R. Siegwart, "Design and Control of an Indoor Micro Quadrotor", *in Proc. of the Int. Conf. on Robotics and Automation ICRA'2004*, New Orleans, USA, 2004.

[3] A. Tayebi, S. McGilvray, Attitude stabilization of a four-rotor aerial robot, *in Proc. of 43rd IEEE Conf. on Decision and Control*, Atlantis, Paradise Island, Bahamas, 2004.

[4] 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.

[5] H. Voos, Autonomous Systems Approach to UAVs, *in Proc. of the 18th Bristol International Conference on Unmanned Air Vehicle Systems*, Bristol, UK, 2003.

[6] J.R. Cloutier, "State-Dependent Riccati Equation Techniques: An Overview", *Proc. of the 1997 American Control Conference*, June 1997, Albuquerque, NM.

[7] Y. Zhang, S. Agrawal, P. Hemanshu, M. Piovoso, Optimal Control using State Dependent Riccati Equation (SDRE) for a Flexible Cable Transporter System with Arbitrarily Varying Lengths, *Proc. of the 2005 IEEE Conference on Control Applications*, Toronto, Canada, August 2005, pp. 1063 - 1068.

[8] C. Willard, B. Randal, "Ensuring Stability of State-dependent Riccati Equation Controllers Via Satisficing", *Proc. of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada USA, Dec. 2002, pp. 2645-2650.

[9] P.K. Menon, T. Lam, L.S. Crawford, V.H. Cheng, "Real-time Computational Methods for SDRE Nonlinear Control of Missiles", *Proc. of the 2002 American Control Conference*, May 2002, Anchorage, AK.