Indoor Navigation for Quadrotor UAVs Using Schematic Environment Maps

Tomasz Krokowicz*†, Miguel Gasca*, Holger Voos*, Dariusz Ucinski†

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

†University of Zielona Gora, Institute of Control and Computational Engineering,
65-246 Zielona Gora, Poland, Email: {d.ucinski}@issi.uz.zgora.pl

***Abstract*— Quadrotor UAVs are a very promising type of small unmanned aerial vehicles for indoor applications because of the easy construction and propulsion principle that also allows very slow and hovering flight. However, the payload of the vehicle is very limited and the nonlinear dynamics requires advanced stabilizing control. In this paper, a flight control and navigation system for an autonomous quadrotor UAV is proposed which is especially suitable for indoor applications. The basic flight controller comprises a cascaded nonlinear control structure which finally stabilizes a commanded velocity vector. The navigation systems is based on schematic maps of the environment and sensor information which is delivered by simple and small ultrasonic sensors. First simulation and experimental results underline the performance of the obtained solution.**

1. Introduction

One special area of application for unmanned aerial vehicles (UAVs) is indoor flight, e.g. for surveillance tasks. Indoor flight requires a suitable vehicle type 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 symmetric 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. By varying the speed of the single motors, the lift force can be changed and vertical and/or lateral motion can be generated, see also Fig. 1 for a sketch of a quadrotor UAV.

Besides the right vehicle type, the most challenging problems with regard to UAV indoor flight comprise the limited payload, the fact that GPS-based localization cannot be applied and that very precise flight control algorithms are necessary. However, the limited payload dramatically reduces the possible size, available power and also processing capacity of any sensors and on-board computer systems. Regarding flight control for quadrotor UAVs, a considerable number of solutions is already proposed in the literature, see e.g. [1], [2], [3], [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 [2], H_∞ or SDRE control [4]. Recently, a new cascaded nonlinear flight controller with high accuracy and easy implementation was proposed in [5], and experimental results of this controller are also presented here.

Based on this nonlinear flight controller, this paper also proposes a new approach for UAV indoor navigation. While indoor navigation of ground-based robots is extensively studied, see e.g. [6] for an overview, the development of navigation approaches for UAV indoor flight is just at the beginning. Since GPS-based navigation is not possible for inddor applications, most of the already derived approaches are based on cameras as optical sensors and image processing or optical flow methods, see e.g. [7], [8]. The drawback of these methods is the necessary electrical power for the sensors and the enormous processing capacity for the image processing algorithms. In this paper we propose a novel method for UAV indoor navigation that is based on low-cost and -power ultrasonic sensors and rough schematic maps of the environment. The resulting localization and navigation is not very accurate but sufficient for most indoor applications and has the big advantage of a comparatively simple algorithm and implementation. In the following, the basic flight control structure is first summarized in order to provide the basis for the development of the navigation system based on schematic maps. The underlying algorithms and navigation strategies are then derived in more detail, first simulation and experimental results underline the obtained performance.

2. Modelling and Basic Control

A. 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], [2], [4] or [5], 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 $\mathbf{\Omega}^T = (\phi, \theta, \psi)$, 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 [5]). The

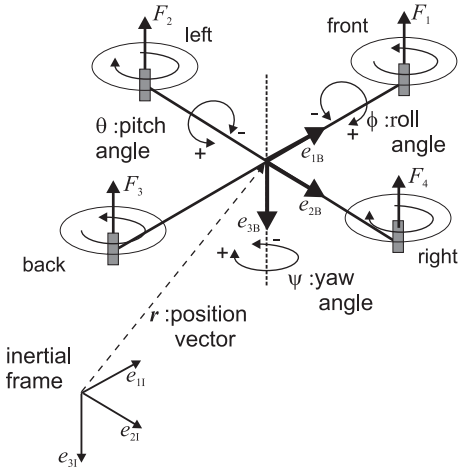

Fig. 1. Configuration, inertial and body fixed frame of the quadrotor.

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 of conservation of momentum yields

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

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:

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

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

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

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

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

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:

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

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:

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

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).

B. VEHICLE 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.

1) 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

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

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

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

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

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

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 [5] 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:

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

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

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

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 [5].

2) 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

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

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

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

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:

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

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 [5] that (13) can be solved analytically by calculating the inverse function of $\boldsymbol{f}$:

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

3) 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 [5], while some experimental results will be presented in this paper.

The next step of designing a navigation system however requires a dynamic model of the controlled quadrotor, i.e. a dynamic model of the two cascaded control loops of attitude and velocity control loop. If we assume that the inner attitude control loops are sufficiently fast and could be approximated by a static system as discussed before, the overall vehicle control system consists of three independent velocity control loops which can be approximated by linear first-order system, respectively,

$$\frac{V_x(s)}{V_{xd}(s)} = \frac{X_1(s)}{X_{1d}(s)} \approx \frac{1}{T_1 \cdot s + 1} \tag{16}$$

with $T_i = 1/k_i, i = 1, 2, 3$, see (14). A simulation of the step response also supports this approximation, see Fig. 2 for the

example of the step response with regard to the velocity v_x in x-direction. Similar results are obtained for the step response of the other two velocities. These first-order approximations of the controlled quadrotor UAV is now used for the design of the navigation system.

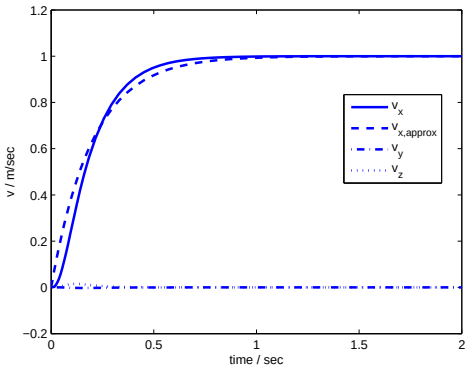

Fig. 2. Step response with regard to v_x of the controlled quadrotor.

3. Indoor Navigation Using Schematic Maps

The algorithm is a simplification of the 3D navigation problem to 2D, and was intentionally developed for a ground-based mobile robot ([**?**]). The schema of the algorithm is based on human behaviour - e.g. it is enough to tell to the human to go straight and turn second left. The idea is the same – to achieve a goal point from a starting point it is needed to provide a list of manoeuvres between those points. It is assumed that a map of the environment is known. In future steps a mapping algorithm will be developed. Obstacles in the environment are binarized – there is only a description of the order of the obstacles. The algorithm does not need to know anything about the obstacles – it needs to know where obstacle should appear. It is also assumed that obstacles are represented by geometric shapes - each environment might be mapped with rectangles. The minimum rectangle size has to be big enough to be recognised using ultrasonic sensors.

Because of the characteristics of ultrasonic sensors the vehicle has to move within a range to the wall, since it measures the distance to the obstacle with a relatively wide angle and it might be impossible to find a gap between two obstacles as showed on Fig. 3.

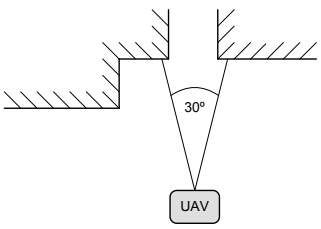

Fig. 3. Wide angle error

TABLE I
Robot directives

Type	meaning
TURN_LEFT TURN_RIGHT	right/ left turn
WALL_TURN	turn before the wall
TURN_LEFT_BEFORE TURN_RIGHT_BEFORE	turn before obstacle
above + JOB_DONE	with task end

A. Algorithm schema

The first step is to find the characteristic points. These points are points close to corners of obstacles - only in those points the algorithm can be sure regarding its position. It means that only in the moment when switching from an obstacle to the next the algorithm knows where robot is. While following track of a wall the robot only knows in which part of the path it is.

Since characteristic points are known it is possible to convert the map to graph representation. It makes the path choosing easier with using graph theory algorithms. In this case the Dijkstry algorithm is used. Without odometry it is difficult to define criteria for path optimisation – the distance between characteristic points is not known. A number of logical blocks to determine distance is used but it may cause a non optimal path. It is assumed that it is not primary to follow the optimal path but reaching goal point without an accurate map.

When the path is found the algorithm determines directives for the robot. Based on characteristic points a list of the manoeuvres is prepared. That list contains the type of behaviour and a counter of obstacles after which it should turn. Possible types of manoeuvres are shown in table 1. An important assumption of the algorithm is that robot is fully autonomous. But still there is the need to provide communication between the robot and a base station. Communication is needed for sending the map to the UAV and receiving feedback from it, and for security reasons for manual steering. In any case it is not a critical part of the implementation. In the case of a broken link the quadcopter continues working on the current task.

Schematic map is an array with specified obstacles. Definition of the map requires placing of the obstacles in proper order. Each obstacle has to be placed on the map. Size of the obstacles doesn't influence to the accuracy of the algorithm. Map is defined by the user using simple diagram of the environment by placing obstacles in proper places (see Fig. 4). Using the map directives for robot are prepared. According to possible directives (shown in table 1), list of it is prepared. There is additional parameter given for each manoeuvre given which meaning is different for each manoeuvre. For given map directives list is shown in table 1.

B. Algorithm verification

The algorithm was developed for a ground-based robot. The first implementation has been done for a Lego NXT set

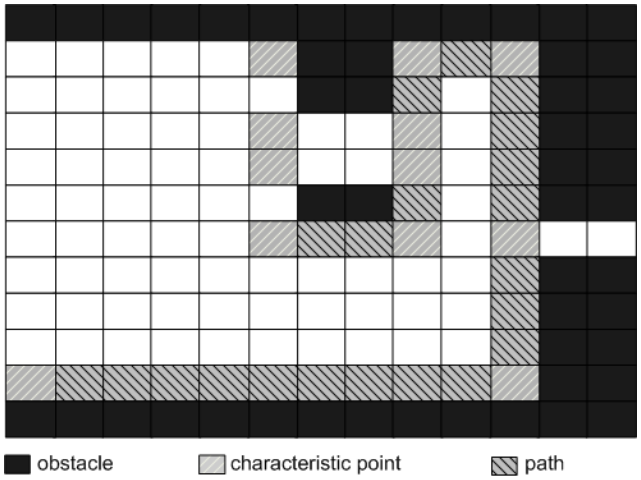

Fig. 4. Map example

TABLE II
DIRECTIVES FOR GIVEN MAP

Manoeuvre	parameters/meaning
WALL_TURN	-1/left turn
WALL_TURN	-1/left turn
WALL_TURN	-1/left turn
TURN_RIGHT + JOB_DONE	2/second possible turn

based robot [9]. It was equipped in 2 ultrasonic sensors on both sides for obstacle detection, on left and right side of the robot, and one touch sensor in the front.

Inaccuracy in the scale of the map had no influence on path following as far as logical structure of mapped environment is kept. In case of inaccurate logical mapping there is a need for additional-local navigation.

C. Control

The main difference in control between the ground based robot and the quadrotor is the danger of getting too close to the wall for the integrity of the UAV. Therefore, it has to be assured that it will stay within a safety distance to the wall. For that reason, a P-controller is used to control each direction $(\dot{x}, \dot{y}, \dot{z})$. Since the quadrotor can move in any direction, we assume that it does not have a preferred direction of movement and we keep always a constant orientation (ψ constant).

A PID type controller was chosen because of its speed comparing to the CPU power needed, as well as for the possibility to implement it into hardware, for a more robust, and fault tolerant control.

Mainly, the linear speed $\dot{x}, \dot{y}, \dot{z}$ of the quadcopter is controlled to be the desired one while secondly trying to keep the angles (pitch, yaw, roll) as close to zero as possible. While moving in x direction, meaning while commanding a velocity in x direction, the controller takes care of keeping a constant distance to the wall in the y direction and a constant altitude z, and vice versa.

The control algorithm, which takes care to keep a constant distance to the wall, reads the ultrasonic sensors, and

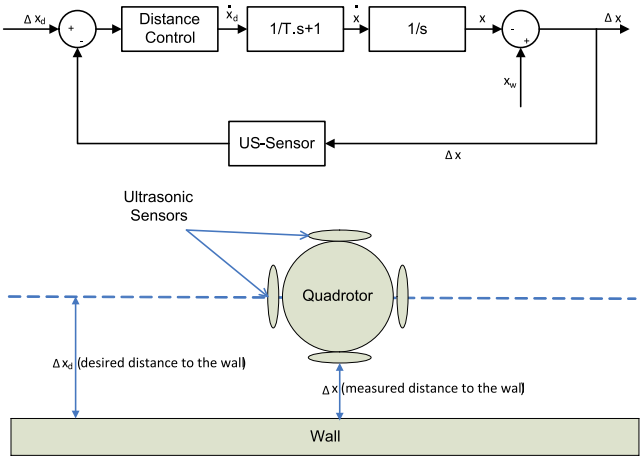

Fig. 5. Control Loop in X and schematic explanation

generates a desired velocity as a function of the given path, which is then given forward to the stabilisation layer of the UAV.

Since the three directions are decoupled, the system which can be approximated to a first order system, with the same behaviour in x,y and z direction. This allows to employ three identical control loops as can be seen in fig. 5

The main benefit of the ultrasonic sensors are from a physical point of view their low weight, and low energy consumption. This two factors are important in mobile robotics, but even more important for a flying vehicle, because of the difficulty to carry a power source, and limited payload. Furthermore, they require only a low amount of CPU power without the need of big filtering, which, again, is an important factor in our case. Nevertheless, they give us a rough scheme of the environment, which is accurate enough to navigate with the help of the schematic map. For future applications, such as mapping, a more advance sensor may be employed additionally.

4. SYSTEM STRUCTURE

The design of the quadrotor electronics is as follows. The Gumstix/Robostixplatform combination is used together with additional onboard sensors. For stabilisation we use an IMU (Inertial Measurement Unit) sensor by XSensand for distance measurements and collision avoidance ultrasonic sensors by Devantech. Hardware bases on two devices - microcontroller - Robostix and higher level unit - Gumstix. These popular in small UAVs are providing all necessary functions and are powerful enough for all tasks.

The Microcontroller is responsible for control and steering (control and steering layers). On the Atmega 128 processor control algorithms and sensors readings have been implemented since being independent of the higher layer level operations. The Robostix board communicates with the Gumstix board (through I2C bus) and takes commands from that device. Research has shown that control for quadrocopter flight can be developed on that kind of microcontroller and now algorithms are being moved to the Robostix board.

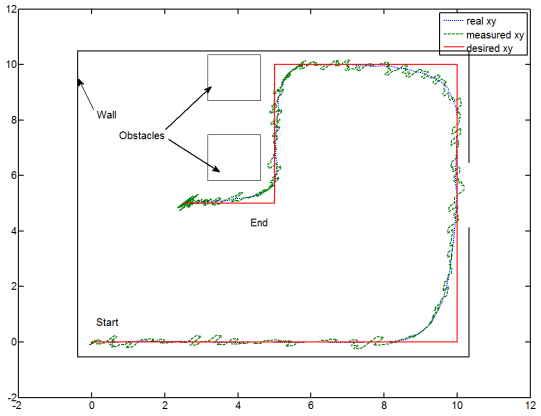

Fig. 6. Path followed by the Quadcopter

5. Simulation Results

The simulation has been done completely inside Matlab and Simulink environment. As one can see in Fig. 6 the simulation shows that the quadcopter is capable of following the desired path with neglectable error, besides the high inaccuracy of the simulated sensors. It keeps always a steady safety distance to the wall and obstacles, so avoiding danger to collision.

The simulation consists of the quadcopter flying inside a room, with one big closet in the middle. The UAV will follow the desired path along the wall, until reaching the goal as can be seen on Fig. 6. This path is inputted to the controller by a higher level algorithm.

In order to simulate the sensor noise a Gaussian distribution has been used.

The error in x and y position is less than 5 cm, besides being the sensor error as big as 2,5 cm. The linear velocities are kept with a maximum of 0.2 m/s, with smooth changes. Further, as can be seen on Fig. 7, the angles hold by the quadrotor are inside a controlled range of [-1 1] degrees, as well as the angle velocities, [-20 20] degrees/s. The altitude is kept inside a controlled range of $\pm$ 0.2 m of the desired value.

The quadrotor is able to follow a steady path, as can be seen on Fig. 6 by the little deviation between the real path and the desired path. This is possible despite the high error of the simulated sensors.

The distance to the wall is kept at a constant pre-programmed value, assuring that the quadrotor will not collide against the wall or other obstacles. The smooth linear velocities show that the quadrotor moves in a steady, controlled manner.

6. Summary and Future Work

Our team is now working on repeating the experiments on the real quadrotor. The next steps are adapting the ultrasonic sensors to the quadrotor requirements. With the help of an experimental platform the same readings as in the quadcopter will be achieved, due to the same sensor placement. This will allow easy testing without the disadvantages of the flying vehicle. There is a need for a proper sensor reading

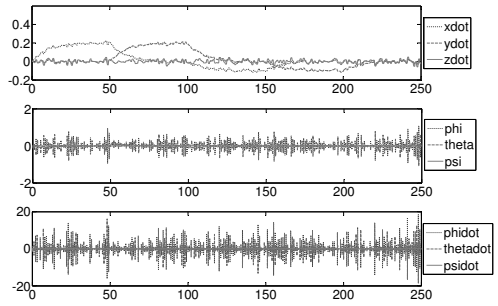

Fig. 7. Linear velocities (m/s), angles ($degree$), and angular velocities ($degree/s$)

and merging, as well as the programming of probabilistic methods for obstacle avoidance.

The algorithms mentioned in this paper have not been developed for use on the quadrotor hardware so far, so there is a need to adapt them to the Gumstix board. This work will include tests on different programing environments – original software has been built based on Java SE, and performance of it has to be tested on embedded Linux.

A simulation of a real environment is being developed. The goal is to provide a testing of the UAVs which gives all data as in the real operating environment including sensor reading and camera pictures. It should map a real environment with real obstacles. This simulation will include HIL (Hardware in the Loop) simulation for the software, as well as the hardware.

References

[1] 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.

[2] 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.

[3] 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.

[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 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.

[6] B. Siciliano, O. Khatib (Eds.). Springer Handbook of Robotics. Berlin, Heidelberg, New York: Springer, 2008.

[7] A. Beyeler, J.C. Zufferey, D. Floreano. Vision-based control of near-obstacle flight. Springer: Autonomous Robots (2009) 27: 201219.

[8] F. Kendoul, K. Nonami, I. Fantoni, R. Lozano. An adaptive vision-based autopilot for mini flying machines guidance, navigation and control. Springer: Autonomous Robots (2009) 27: 165188.

[9] T. Krokowicz. Mobile robot navigation using schematic environment maps. Master Thesis Report, University of Zielona Gora, 2008.