The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Markerless, Vision-Assisted Flight Control of a Quadrocopter

Sebastian Klose, Jian Wang, Michael Achtelik, Giorgio Panin, Florian Holzapfel and Alois Knoll

Abstract— **In this paper, we present a system for controlling a quadrocopter using both optical and inertial measurements. We show how to use external stereo camera measurements for visual servoing, by onboard fusion at high rates, only natural features provided by the vehicle and without any active marker. In our experiments, we show the accuracy and robustness of our system during indoor flights, as well as robustness to external flight disturbances.**

I. INTRODUCTION

Unmanned aerial vehicles (UAV) and in particular multi-rotor systems [1][2] have gained much interest within the last years, because of the wide range of potential applications. Due to the small size, indoor applications are considered particularly interesting for these devices. Unfortunately, for indoor position control one cannot rely on GPS information, and the on-board inertial measurement units (IMU) are subject to drift over time. Instead, visual cues can provide rich and precise information for navigation purposes.

In this paper, we present an external visual stereo system for tracking a quadrocopter, providing an absolute position measurement for the controller of the vehicle. Moreover, we provide a marker-less approach, only using the CAD model of the vehicle, automatically sampling visual contour features. The benefit of our approach is an accurate and cheap system, more robust to occlusions with respect to traditional marker-based systems.

A lot of research is being done in the direction of visual control of UAVs. Most systems have the camera attached directly to the vehicle, providing on-board measurements of natural features detected in the environment (e.g. [3], [4], [5], [6]). Our system instead uses an off-board tracking approach, which on one hand shall serve as an evaluation platform for future developments of on-board applications, and on the other hand can be used for formation flight of multiple UAVs. One popular related work is the *RAVEN* system [7][8], that makes use of *VICON* [9] to track multiple quadrocopter. RAVEN is used for development of novel swarm intelligence, cooperation algorithms, and related applications. Altug et. al. [10] use a hybrid camera setup, where one camera is set on the ground and a second one is attached to the vehicle, facing forwards in order to compute its 3D pose for control. They are solely relying on visual measurements for both position and orientation estimation, which results in inaccuracies due to the relatively slow update rate (20Hz), especially for the attitude estimate. Furthermore, in the paper is stated that the vision system is not reliable when the scene lighting changes. In [11] a single down-facing onboard camera is used, to track artificial features on the ground. The system uses direct image-based visual servoing to control the UAV instead of pose-based visual servoing. In [12] a trinocular ground camera system, consisting of Firewire cameras, is used for estimating the 3D position of the vehicle, by tracking four colored markers attached to it. Probably the work most related to our system is described in [13]. Here the authors also use a cheap, transportable stereo-system based on standard webcams and active LED markers attached to the vehicle, externally tracking the quadrocopter at 15Hz. Their system has a very similar hardware setup to ours, but it differs in the algorithmic design, especially of the visual tracking part.

The present paper is organized as follows: In Section II we give an overview of the whole setup, and describe the hardware and software resources, as well as the overall algorithmic structure. Afterwards, in Section III we explain the details of the control system, while in Section IV we investigate the visual tracking part. Section V shows experimental results and evaluation. Finally, in Section VI we summarize our work and provide an outlook of future developments and utilization of the current system.

II. SYSTEM OVERVIEW

In this Section, we describe the hardware setup and give an abstract overview of the developed system parts, which will be investigated more in detail in the respective Sections.

A. Hardware

For providing a safe demonstration and testing setup, we built a box of sizes $2 \times 2 \times 3m$, with the cameras mounted on the upper-left and upper-right corners of one of the longer sides. Visual input is given by standard webcams (Logitech Quickcam Vision Pro), capable of delivering frames at 25Hz with a resolution of 800x600 pixels, covering a field of view of approximately 60°.

The quadrocopter device is a *Hummingbird Autopilot* from *Ascending Technologies* [14], which is based on the one described in [2], [13]. The on-board inertial measurement unit (IMU) consists of three gyroscopes and three acceleration sensors, and its flight control system operates on two ARM-7 microprocessors. A low-level controller performs data fusion (attitude angles) and can also perform attitude control. The

Sebastian Klose, Giorgio Panin and Alois Knoll are with the Chair for Robotics and Embedded Systems at the Department of Informatics, Technische Universität München, 85754 Garching, Germany kloses@in.tum.de, panin@in.tum.de, knoll@in.tum.de

Jian Wang, Michael Achtelik and Florian Holzapfel are with Faculty of Mechanical Engineering chair for Flight System Dynamics, Technische Universität München, 85754 Garching, Germany jian.wang@tum.de, michael@achtelik.net, florian.holzapfel@tum.de

high-level controller is dedicated to user-specific tasks, and can interact with the low-level processor in different ways. In our case, position and attitude control is performed by the high-level processor, and the sampling rate of both controllers is 1KHz (see [2]). Communication to a ground station PC, operating Matlab-Simulink, is done via *XBeePro* modules [15]. For security reasons, and for a more clear appearance of the object silhouette during flight, we employ an additional rotor guard and casing for the vehicle, provided by Ascending Technologies, which is also shown on the lower left of Figure 1, together with the used CAD model. This ensures safety for the operators and the audience, during public tests of the system.

B. Software

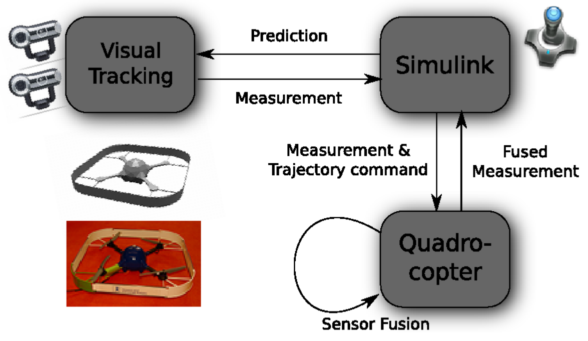


Fig. 1. Subsystems of the application

The main parts of the software setup can be seen in Fig. 1. Visual tracking operates on the predicted pose of the vehicle from the last time step, using a known CAD model of the vehicle and the current camera images. Communication of the measurements to Simulink is done via *S-Functions.*

The Simulink model is responsible for generating trajectory commands (predefined, or given by joystick), using the current visual data together with the past information received from the onboard sensors. These positional commands are sent to the vehicle, where they are integrated into the on-board control loop.

The tracking system has been developed using the OpenTL C++ library [16], and runs on the same ground station PC as the Simulink model. As already stated, our control loops, are running on the high-level processor, but we also have a redundant safety control system (for the attitude only) on the low-level processor, which can be activated and controlled by the remote control.

The processing hardware consists of a *Core i7* Quad-Core with 6GB memory and a Geforce 250GTS graphics processor. The current operating system is Microsoft Windows XP, 64-bit Edition.

III. CONTROL LOOP AND ONBOARD DATA FUSION

The on-board flight control system has a position controller, receiving commands from the ground PC with Matlab-Simulink. The position command is either generated by predefined trajectories, or through a joystick, which allows to fly freely within a fixed range.

A. Data Fusion

In order to achieve a high bandwidth for the system dynamics, fast data fusion is an essential requirement. For the attitude angles and angular rates (excluding the heading), we use the fused 1kHz IMU data calculated on the low-level controller, provided by Ascending Technologies [14]. For position control, we also need a fast data fusion of all kinematic measurements, that consist of accelerations updated at 1kHz rate, as well as position and heading, computed by the two cameras at 25Hz rate. Therefore, an adequate on-board kinematic filter has been developed, taking into account the computational power of the ARM-7 processor, where a full state filter for position and attitude would not be feasible. In particular, the position filter is a modified Luenberger observer [17]

$$\dot{\hat{x}} = \begin{bmatrix} 0_3 & I_3 \\ 0_3 & 0_3 \end{bmatrix} \cdot \hat{\mathbf{x}} + \begin{bmatrix} 0_3 \\ I_3 \end{bmatrix} \cdot u + L\left(\vec{y} - I_6 \hat{\mathbf{x}}\right) \quad \text{with}$$
$$\mathbf{x} = \begin{pmatrix} \vec{x} \\ \vec{v} \end{pmatrix} \quad u = \vec{a} \quad y = \begin{pmatrix} \vec{x}_{\text{meas}} \\ \vec{v}_{\text{meas}} \end{pmatrix} \quad L = \begin{bmatrix} L_1 I_3 & L_2 I_3 \\ L_3 I_3 & L_4 I_3 \end{bmatrix}$$

where $\mathbf{x}$ is the state vector (position and velocity), $\hat{\mathbf{x}}$ the estimated state vector from the observer, $\vec{a}$ the acceleration vector, $\mathbf{y}$ the measured state and I_n are $(n \times n)$ identity matrices. The L-matrix is calculated as the optimal Kalman gain of the system, given the process and measurement noise covariances.

In order to ensure stability of the observer dynamics, a complimentary filter generates velocity measurements, by combining the high frequency acceleration with the low frequency position feedbacks

$$\begin{aligned} V =& \frac{T \cdot s + 1}{T \cdot s + 1} \cdot V \\ =& \frac{T}{T \cdot s + 1} \cdot s \cdot V_{\text{IMU}} + \frac{1}{T \cdot s + 1} \cdot s \cdot \mathbf{x}_{\text{Vision}} \\ =& \frac{T}{T \cdot s + 1} \cdot a_{\text{IMU}} + \frac{s}{T \cdot s + 1} \cdot \mathbf{x}_{\text{Vision}} \end{aligned} \qquad (1)$$

with V denoting the velocity and $\mathbf{x}_{\text{Vision}}$ the positional measurements from the visual tracking system.

Furthermore, $\frac{1}{T}$ is the cross-over frequency of the filter, typically chosen to trade off between the two measurements. It is designed based on physical consideration and verified by experiments, considering the noise of the combined position measurements.

The latency, due to image processing ($\approx 40ms$) and to the communication between Simulink and the quadrocopter ($\approx 40ms$), was taken into account by adjusting the position measurement with a velocity compensation. In particular, the fused velocity is multiplied by the time delay, to give a prediction for the position change in the last delay cycle.

The new position measurement, which is the filter input, is the sum of the position change during the delay time, and the position measurement from the vision algorithm. The implementation is done in Simulink, using only fixed-point data types. Furthermore, it has been optimized by removing many unnecessary matrix computations.

The most important advantage of the modified observer running onboard, is the update rate of 1 kHz. Therefore, disturbances detected by the acceleration sensors can be compensated within milliseconds, before it is possible to observe a position error by the vision system. On the other hand, this structure uses the vision signal in order to compensate for the acceleration sensors drift. In fact, using only a complementary filter gives a trade-off between direct reaction and noise, whereas our filter structure combines the advantages of low and high time constants of the complementary filter, and shows very fast reaction and little noise.

For the heading, a simple complimentary filter using the estimated angle from the vision system, and the IMU gyroscope (angular rate) measurements, has been implemented.

B. Position Controller

The position controller onboard has two cascaded loops, namely the position control loop with a relative degree of two, and an attitude control loop, also with degree two. Since the quadrocopter is a highly nonlinear, multi-variable and strongly coupled system, as also described in [18], a nonlinear controller based on feedback linearization has been developed.

Feedback linearization as a baseline control strategy can exploit the physical capability of the plant, and therefore accomplish complex, highly curved three-dimensional trajectories. With an adequate knowledge of the plant dynamics, the control approach can transform the nonlinear system into an equivalent linear system without any simplification, through exact state transformation and suitable control inputs [19]. The overall structure of the control system, including the respective approximate processing rates, is shown in Fig. 2.

Based on this transformation, linear controllers like a PID can be used for error control.

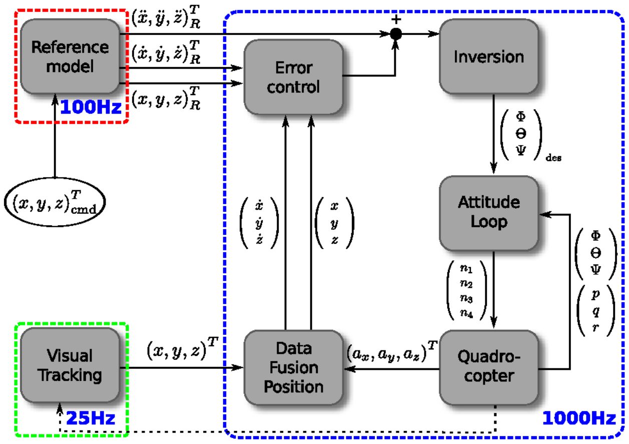

$(\ddot{x},\ddot{y},\ddot{z})_R^T$
$(\dot{x},\dot{y},\dot{z})_R^T$
$(x,y,z)_R^T$
Reference model
100Hz
+
Error control
Inversion
$\begin{pmatrix}\Phi\\\Theta\\\Psi\end{pmatrix}_{des}$
$(x,y,z)_{cmd}^T$
$\begin{pmatrix}\dot{x}\\\dot{y}\\\dot{z}\end{pmatrix}$
$\begin{pmatrix}x\\y\\z\end{pmatrix}$
Attitude Loop
$\begin{pmatrix}n_1\\n_2\\n_3\\n_4\end{pmatrix}$
$\begin{pmatrix}\Phi\\\Theta\\\Psi\end{pmatrix}$
$\begin{pmatrix}p\\q\\r\end{pmatrix}$
Visual Tracking
$(x,y,z)^T$
Data Fusion Position
$(a_x,a_y,a_z)^T$
Quadro-copter
25Hz
1000Hz

Fig. 2. Structure of the controller including the different processing rates. In the figure, x, y and z denote the position, Φ, Θ and Ψ denote the bank, pitch and yaw angles. p, q and r are the respective angular rates in the same order. n_1, n_2, n_3 and n_4 are the four rotations per minute (RPM) of the rotors, and $a_x, a_y,$ and a_z are the accelerations obtained from the IMU

The attitude loop was previously designed at the Institute of Flight System Dynamics [20] and therefore is not the main focus of this paper. Input commands are desired Euler angles and thrust, and output signals are the commanded rotational speeds of the four motors. The attitude loop is transformed in an equivalent linear system as described above, by inverting the attitude and momentum dynamics. The required angular accelerations, as inputs for the inversions blocks of this relative degree-2 system, have to be computed in the reference model.

In the position loop, the input command is the desired position in world coordinates, while control signals are given as input for the attitude loop. In both cascades, 2nd-order reference models are used for generating reference signals, governed by the differential equation

$$\ddot{y}_R + 2\zeta\omega_0 \cdot \dot{y}_R + \omega_0^2 \cdot y_R = \omega_0^2 \cdot y_c \qquad (2)$$

with reference signal y_R, relative damping ζ, natural frequency ω_0, and input command y_c.

The attitude loop is able to handle reference signals with a natural frequency of 20 Hz, and a relative damping ratio of 1. To ensure a stable and robust tracking performance, the outer loop reference dynamics are set with a natural frequency of 4 Hz, and relative damping 1.

The feedback error is used with proportional and integral gains. Since the error dynamics should follow the reference dynamics, the gains can be calculated accordingly. Because of the nonlinear dynamic inversion of the system and the high update rate, the linear error controller is sufficient to achieve good tracking results.

Since the output signals of the position loop controller and reference models are desired accelerations, the translation dynamics has to be inverted. For this purpose, a dynamic inversion block maps acceleration commands to Euler angle commands. In this application, it is sufficient to use the world frame (denoted by W), neglecting earth rotation and angular rate due to translation, as inertial frame, in order to apply Newton's second law.

The complete reference model and error control is then expressed in the W frame. To simplify the inversion, the $\bar{O}$ frame has been introduced: it is a leveled frame with the same azimuth rotation as the local body frame B. In the dynamic inversion, the angular rate between the W frame and the $\bar{O}$ frame is rather small, because for position control the quadrocopter does not need to yaw, and can be neglected compared to the sensor noise in the state vectors.

In a first step, the desired accelerations are transformed to the $\bar{O}$ frame, by a simple rotation through Ψ. By applying Newton's second law, we get:

$$m \cdot \left(\dot{\vec{V}}\right)_{\bar{O}} = \left(\vec{F}^G\right)_{\bar{O}} + \left(\vec{F}^G_{\text{grav}}\right)_{\bar{O}} = M_{\bar{O}B}\left(\vec{F}^G\right)_B + \left(\vec{F}^G_{\text{grav}}\right)_{\bar{O}}$$

with F denoting the forces, m the mass of the vehicle and $M_{\bar{O}B}$ the transformation matrix between $\bar{O}$ and B frames.

Solving for the pitch angle Θ, bank angle Φ and thrust T, we get the inversion equations:

$$\frac{T}{m} = \sqrt{\dot{u}^2 + \dot{v}^2 + (\dot{w} - g)^2} \qquad (3)$$

$$\Theta = \arctan\frac{\dot{u}}{\dot{w} - g} \qquad \Phi = \arcsin\frac{m\dot{v}}{T}$$

where u, and v are the velocities in x- and y-direction. Using the above equations, the error controller, the described reference model and the position loop can all be implemented in Simulink, using only fixed-point data types, to ensure high computational efficiency on the microprocessor hardware.

IV. VISUAL TRACKING SYSTEM

In this section we describe the visual tracking system. As already mentioned, the visual tracker relies, to some extent, on the results of the fused measurements from the onboard controller.

We use two standard webcams mounted at the upper corners of the box. The cameras have been calibrated to get their intrinsic parameters, as well as their relative positions and orientation, by using OpenCV [21] and a calibration pattern. Furthermore, the global world frame is set approximately to the center of the box, with the Z-direction pointing downwards for convenience during control (approximating a North-East-Down frame).

A. Tracking loop

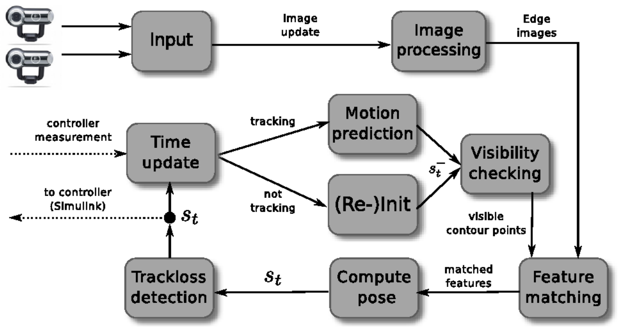


Fig. 3. Flow of the visual tracking system

The visual tracking system is based on well-known edge matching algorithms [22], [23], [24] for pose estimation, that are given to a Kalman Filter, in a Bayesian prediction-correction scheme [25].

The processing flow of the visual tracker is shown in Fig. 3. At each time step, the fused state results s_t^{fused} from the onboard controller of the previous time step are used as prior state estimate.

Image pre-processing is done by Canny edge detection [26], subsequently used for model matching and pose estimation. Edge detection is the computationally most expensive part of the system; therefore, for each camera a separate processing *thread* is responsible for this operation. Each thread is triggered immediately after a new image is acquired, while model matching is performed only after all threads have finished.

B. Detection

In the very first frame, or to recover from a track loss, a system initialization, or detection, is required. For this purpose, we exploit the color of the vehicle.

In fact, the center of the vehicle is covered by a blue marker, while the heading direction is marked with green. This information, together with the constraints given by the area in which the vehicle can operate, due to the limited field of view of the cameras, is used to triangulate the 3D vehicle position and heading. The association between segmented colors in HS-space on the two views, is done by using the epipolar lines. Triangulated positions are also restricted to the possible operation volume, in order to sort out wrong matches.

Finally, we take the combination of central position and heading information which is consistent with the model, in terms of an approximate distance from the green blob center to the blue one. The *pitch* and *bank* angles are taken from the previous result s_{t-1}^{fused}. Detection of the heading is essential, especially in the very beginning, and therefore we cannot rely on the initial value estimated by the IMU.

The system stays in detection mode, until an initial pose can be computed, which is then taken as the current prior s_t^- for tracking.

C. Tracking

In tracking mode, we predict the prior state s_t^- by applying a constant velocity dynamics to the previous estimate s_{t-1}^{fused}, after fusion.

Afterwards, we sample visible points from the model contours at the predicted state s_t^- in both camera views, by applying the GPU-based algorithm described in [27]. The result of this algorithm is a set of contour points, together with the corresponding screen normals and Jacobians, needed for feature matching and pose update. In particular, matching is done by performing a nearest neighbor search along the normals to each contour points, for the closest edge in the Canny image, as described in [24].

Furthermore, we also use RANSAC [28] in order to remove outliers, by enforcing collinearity of points belonging to a straight model line. The result is a set of predicted model features h_i and associated image features z_i, with residuals E_i and Jacobians J_i given by

$$E_i = n_i^T (h_i - z_i) \qquad J_i = n_i^T \frac{\partial h_i}{\partial p} \tag{4}$$

where n_i are the respective screen normals. In the pose update step, matched measurements are stacked together into matrices for each camera, in order to compute a joint non-linear least-squares optimization on both views c

$$T^* = \underset{T}{\operatorname{argmin}} \left[\sum_c \left(\mathbf{E}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{E} \right)_c \right] \tag{5}$$

where $\mathbf{R}$ denotes the block-diagonal measurement covariance, which is fully diagonal under the assumption of independent measurements per point. Pose update is done by means of Gauss-Newton

$$\Delta p = \mathbf{H}^{-1}\mathbf{g} \qquad \text{with} \tag{6}$$
$$\mathbf{H} = \sum_c \left(\mathbf{J}^T \mathbf{R}^{-1} \mathbf{J} \right)_c \qquad \mathbf{g} = \sum_c \left(\mathbf{J}^T \mathbf{R}^{-1} \mathbf{E} \right)_c$$

The resulting incremental pose Δp is used to update the homogeneous transformation matrix T, using the exponential map

$$T_{k+1} = T_k \exp\left(\Delta p_i \mathbf{G}_i\right) \tag{7}$$

with $\mathbf{G}_i$ the generators of the Lie algebra [23] for the Euclidean transformation group. The state estimate is finally updated using standard Kalman filtering [25].

The last step in the loop consists of detecting tracking failures (loss detection). For this purpose, we restrict the output of the visual system to lie within a certain range. Especially the position of the vehicle is limited by the controller, but also the attitude is restricted, such that extreme angles ($> 60°$) indicate a problem in tracking since they are never commanded by design. Furthermore, the estimate of the pose covariance matrix gives a hint for the quality of the estimate. By choosing an empirical maximum threshold for the determinant of the posterior covariance, we can detect a track loss.

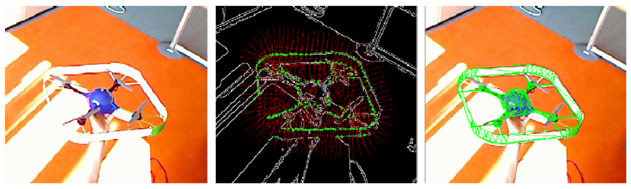


Fig. 4. Tracking steps for one camera. Left: input image. Middle: Canny edge map, overlayed with visible contour points for the predicted state, search normals at each point (red), and associated edges along the normals (green). Right: input image superimposed with CAD model drawn at the output pose

Finally the resulting state estimate s_t, and some flags signalling the state of the tracking system, are sent to the Simulink controller. A snapshot of the tracking procedure is shown in Fig. 4.

V. EXPERIMENTAL RESULTS

In order to show the performance of our system, we tested different trajectories while recording on-line data. We only show here two pre-defined trajectories, namely a circle and a planar, ∞-shaped trajectory. For other trajectories, and flight under joystick input, we kindly refer to the accompanying video of the paper.

Since we do not have access to real ground-truth data, which could be obtained e.g. using a VICON tracker, we can only assess correctness of the visual measurements by sight. However, we also estimated the measurement noise, by putting the quadrocopter at fixed positions within the box, and let the tracking system run for a while. We found the jitter to lie in the range of $\approx 4mm$ for the positions.

In each figure, we are showing the commanded reference signal (green) against the visual measurements (red). Fig. 5(a) shows the result of a commanded circle in X and Y direction, with a radius of 0,4m. Fig. 5(b) shows the result with an ∞-trajectory in the Y-Z plane.

To show the accuracy of our system, we calculated the errors between the commanded and measured trajectories, given in Table V. It can be seen, that the RMS-errors are below 4cm for each trajectory, and the mean 3D distance from the nominal trajectory is approximately 5cm.

The endurance of our system has been tested during the Embedded World Exhibition, where it has been the main demonstration setup at the Mathworks trade-show booth, flying approximately 6h/day over three days.

Trajectory	RMS in X	RMS in Y	RMS in Z	3D mean error	3D std. dev.
Circle	0.033m	0.037m	0.006m	0.045m	0.022m
Infinity sign	0.036m	0.030m	0.022m	0.049m	0.017m

TABLE I

RMS ERRORS, MEAN 3D ERROR AND STANDARD DEVIATION FOR EXAMPLE TRAJECTORIES

The main weakness of our system consists in visual tracking under very fast vehicle movements. Due to the limited frame rate and the low image quality of the USB cameras, image edges get too much blurred and cause a tracking loss. Nevertheless, during control the velocity of the vehicle is kept limited, such that tracking can run robustly. During our test flights we only lost track in case of rough disturbances by hand. Also severe lighting, as can be seen in Fig. 4, may lead to a performance loss.

VI. CONCLUSIONS AND FUTURE WORKS

With the current tracking algorithm and position control system, the vehicle is able to fly with good accuracy and a certain degree of robustness.

The controller and data fusion algorithm are more baseline systems. Based on the available structure, new control theory and specific data fusion algorithm can be tested. Pseudo-control hedging could be implemented based on the current structure, and specific data fusion algorithms can be developed. Extended Kalman filter estimation of the relative position of two adjacent UAVs is being developed, for application of autonomous formation flight. Adaptive control, especially Kalman filter-based, is of particular interest to the authors in the next research phase. For the visual part, we are going to move the cameras and a significant amount of processing onboard, for autonomous indoor flights, so that will use our current system as an evaluation platform for the onboard system.

VII. ACKNOWLEDGMENTS

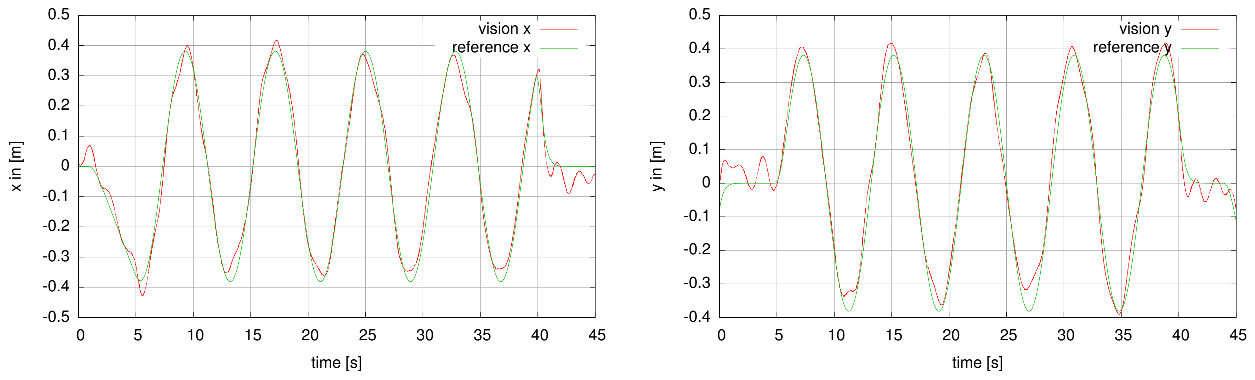
The authors gratefully acknowledge the support by the International Graduate School of Science and Engineering (IGSSE), TU-München, project 4.03 Image Aided Flight Control. Furthermore we would like to thank the Mathworks Team, for the endurance test of the system during the Embedded World 2010.

REFERENCES

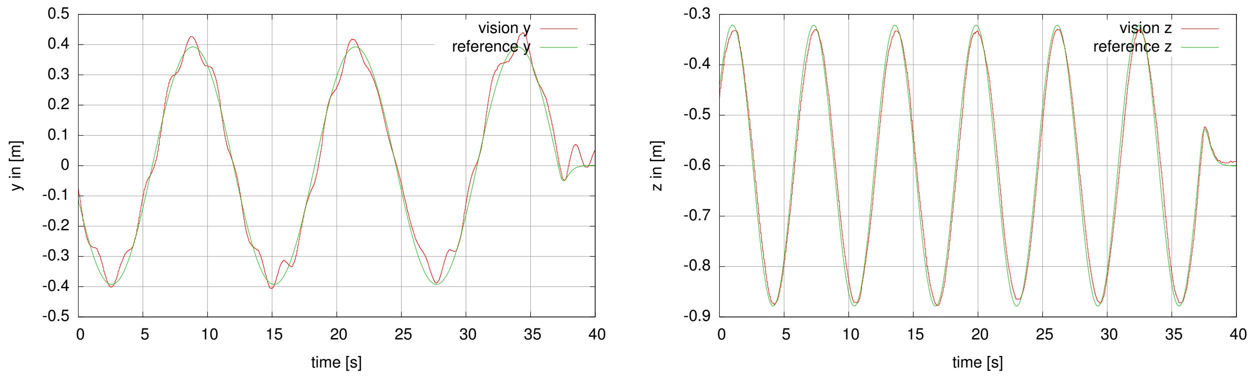
[1] P. Pounds, R. Mahony, J. Gresham, P. Corke, and J. Roberts, "Towards dynamically-favourable quad-rotor aerial robots," *Australian Conference on Robotics and Automation*, 2004.

[2] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1khz," *IEEE International Conference on Robotics and Automation*, pp. 361–366, April 2007.

[3] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares, "Visual 3-D SLAM from UAVs," *Journal of Intelligent and Robotic Systems*, vol. 55, pp. 299–321, August 2009.



(a) Circular Trajectory



(b) Infinity sign trajectory

Fig. 5. Generated reference signal (red) and position measurement obtained from the vision system. Fig. 5(a) shows the X and Y components for the circular trajectory and Fig. 5(b) the Y and Z components for the trajectory shaped like an infinity sign.

[4] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments," *IEEE International Conference on Robotics and Automation*, May 2009.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[6] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conference, Manchester*, 1988, pp. 147–151.

[7] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-time indoor autonomous vehicle test environment," *IEEE Control Systems Magazine*, April 2008.

[8] M. Valenti, B. Bethke, G. Fiore, and J. P. How, "Indoor multi-vehicle flight testbed for fault detection, isolation and recovery," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, August 2006.

[9] (2010, July) Motion capture systems from vicon. [Online]. Available: http://www.vicon.com/

[10] M. Altug, J. Ostrowski, and C. Taylor, "Control of a quadrotor helicopter using dual camera visual feedback," *The International Journal of Robotics Research*, vol. 24, pp. 329–341, 2005.

[11] N. Guenard, T. Hamel, and R. Mahony, "A Practical Visual Servo Control for an Unmanned Aerial Vehicle," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, Apr. 2008.

[12] C. Martínez, P. Campoy, I. Mondragón, and M. A. Alivares-Méndez, "Trinocular ground system to control UAVs," *IEEE International Conference on Intelligent Robots and Systems*, October 2009.

[13] M. Achtelik, T. Zhang, K. Kühnlenz, and M. Buss, "Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors," *International Conference on Mechatronics and Automation*, 2009.

[14] Ascending Technologies GmbH. (2010, July). [Online]. Available: http://www.asctec.de/

[15] Digi International. (2010, July). [Online]. Available: http://www.digi.com/

[16] Chair for Robotics and Embedded Systems, Technische Universität München. (2010, July) OpenTL - A general-purpose tracking library. [Online]. Available: http://www.opentl.org/

[17] K. Narendra and A. Annaswamy, *Stable Adaptive Control*. Prentice Hall, 1990.

[18] T. Hamel, R. Mahony, R. Lozano, and J. Ostrowski, "DYNAMIC MODELLING AND CONFIGURATION STABILIZATION FOR AN X4-FLYER." *Proceedings of the 15th IFAC World Congress*, 2002.

[19] A. Isidori, *Nonlinear Control Systems*, 3rd ed. Springer, 1995.

[20] F. Holzapfel and G. Sachs, "DYNAMIC INVERSION BASED CONTROL CONCEPT WITH APPLICATION TO AN UNMANNED AERIAL VEHICLE," *AIAA Guidance, Navigation, and Control*, no. August, pp. 4907–4907, 2004.

[21] Willowgarage. (2010, July) Open Source Computer Vision Library. [Online]. Available: http://opencv.willowgarage.com/

[22] M. Armstrong and A. Zisserman, "Robust object tracking," *Asian Conference on Computer Vision*, vol. 1, pp. 58–61, 1995.

[23] T. Drummond and R. Cipolla, "Visual tracking and control using lie algebras," *CVPR*, vol. 02, p. 2652, 1999.

[24] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, 2002.

[25] G. Welch and G. Bishop, "An introduction to the kalman filter," Tech. Rep., 2004.

[26] F. J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[27] E. Roth, G. Panin, and A. Knoll, "International workshop on vision, modeling and visualization," *Sampling feature points for contour tracking with graphics hardware*, October 2008.

[28] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.