Visual Tracking and Control of a Quadcopter Using a Stereo Camera System and Inertial Sensors*

Markus Achtelik, Tianguang Zhang, Kolja Kühnlenz and Martin Buss

Institute of Automatic Control Engineering (LSR)
Technische Universität München
D-80290 Munich, Germany
markus@achtelik.net, {tg.zhang, kolja.kuehnlenz, m.buss}@ieee.org

Abstract— In this paper a complete system is designed and implemented, in which the motion of a quadcopter is stably controlled based on visual feedback and measurements of inertial sensors. We focus on developing a cost effective and easy-to-setup vision system. Active markers were finely designed to improve visibility under various perspectives as well as robustness towards disturbances in the image-based pose estimation. Moreover, position- and heading controllers for the quadcopter were implemented to show the system's capabilities. The performance of the controllers was further improved by the use of inertial sensors of the quadcopter. A closed-loop control system is successfully conducted.

Index Terms— Vision system, motion estimation, multiple sensing system, UAV/MAV control.

Fig. 1: The quadcopter with illuminated markers

I. Introduction

In recent years, unmanned aerial vehicle (UAV) and micro aerial vehicle (MAV) are major focuses of active researches, since they can extend our capability in a variety of areas, especially for applications like search-and-rescue and inspection. A significant challenge in developing UAVs is to extract and fuse the useful information in a robust manner and to provide stable flight and an accurate navigation. To exploit various flight behaviors and control strategies of UAVs, we use a quadcopter [1] [2] as testbed.

The control of quadcopter during autonomous flights relies on knowledge of variables like position, velocity and orientation, which can be partly calculated using information provided by on-board inertial sensors. However, the drift of inertial sensors leads to errors during time-discrete integration, making a steadily accurate estimation of the absolute pose nearly impossible. Therefore, an absolute reference is needed, e.g. a GPS system for outdoor applications. Indoors, where no GPS is available, a visual tracking system could be applied. In [3], an efficient iterative algorithm is proposed in order to achieve a globally convergent pose estimation of an object with known 3D geometry. Another accurate pose estimation of moving rigid object using a calibrated stereoscopic vision 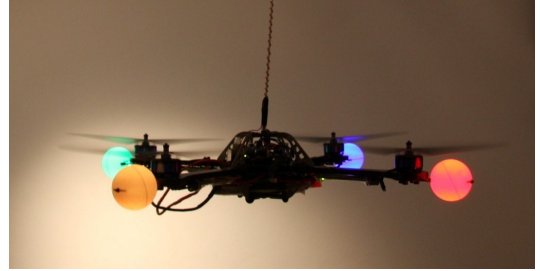setup is described in [4], in which a novel correction scheme compensating accumulated errors was integrated. However, the real time capability of them is not sufficient for quadcopter motion control. In [2] a position controller for a quadcopter was implemented using an optical tracking system by VICON in the "Holodeck" lab at MIT. This intelligent space is nevertheless very expensive and not transportable. 3D-pose estimation and control of a quadcopter using only visual feedback had been done in [5] based on one camera fixed on the ground an additional second camera mounted on the quadcopter facing downwards. The flight volume in this work is very limited and the feature detection applied is sensitive to the light condition and therefore not reliable enough. We see our system not competing with systems developed in [6] [7] where on-board sensors such as IMU, camera and sonars or laser rangefinders are used for pose estimation and obstacle avoidance for an autonomous flight. In contrast, it could be an extremely helpful addition during the development of onboard pose estimation systems mentioned above by providing absolute position information and ground truth. These algorithms usually need a well structured environment that might be totally different from laboratories where tracking systems like VICON are installed, for which reason we like to emphasize the transportability and easy setup of our system.

To deal the aforementioned problems, a complete system is designed and implemented, in which the motion of a quadcopter is stably controlled based on visual feedback and measurements of inertial sensors. We focus on developing a cost effective and easy-to-setup vision system. We use an off-

board stereo camera setup consisting of two web cameras to provide accurate position information. Active markers were finely designed with high-power LEDs of different colors. This design allows visibility under various perspectives and improves robustness towards disturbances. Finally, the image-based pose estimation was experimentally evaluated using a positioning table. We also show the implementation of a position- and heading controller as well as an approach on how to incorporate IMU data. Hovering and autonomous flight are excellently implemented as shown in video.

This paper is organized as follows: firstly, in Section II, we describe the hardware configuration in our application as well as their functional principle. In Section III, marker detection and pose estimation of the quadcopter are introduced. Processing times and accuracy are evaluated in Section IV. The implementation of the controllers and experimental results are shown in Section V. Conclusions are given in Section VI.

II. HARDWARE CONFIGURATION

A. Quadcopter

The 6 DOF of the quadrotor are controlled by four inputs (roll, pitch, yaw, thrust) by varying the lift forces and the balance of reaction torques through changing the rotating speed of the rotors. Two pairs of rotors are spinning clockwise and counterclockwise respectively, so that the sum of their reaction torques is zero during hovering. Unlike normal helicopters, the propellers of the quadcopter applied in this work have fixed pitch angles. This minimalistic hardware design (no rotor linkages etc) makes the quadcopter robust such that it can survive crashes during experiments without getting seriously damaged. Fig. 2 shows the basic design of the quadcopter. The z-axis points towards the ground.
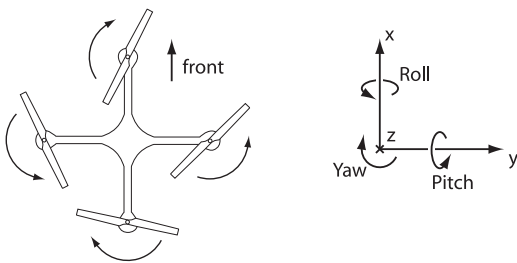

Fig. 2: Basic design of the quadcopter

The quadrotor applied in this work is a "Hummingbird" with an "AutoPilot" controllerboard from Ascending Technologies[1]. The configuration of the quadcopter is basically the same as described in [2]. The main controller board (in the remainder refered to as the inertial measurement unit – IMU) contains inertial sensors consisting of three gyroscopes

for each axis of rotation and three acceleration sensors for each translation axis. The IMU estimates the absolute roll- and pitch angles by fusing the acceleration vector and angular speeds measured by the gyroscopes. A relative yaw-angle is estimated by integrating the angular speed measured by the yaw-gyro. The IMU is equipped with two processors - one "black box" performing the low level attitude and heading control and another high-level processor for user specific applications. Communication to a ground station is realized over a serial port and wifi modules forwarding serial data transparently, like XBeePro modules from Digi[2].

As the quadcopter itself is unstable during flight, all three axes of rotation (roll, pitch, yaw) are already stabilized by the IMU with three independent PD controllers on each angle at a control loop frequency of 1 kHz [2]. Using these inertial sensors, it is only possible to determine the roll and pitch DOFs absolutely without drift or integration. In all other DOFs, only angular velocities or translational accelerations can be measured directly. Although the errors due to time discrete integration are small at the rate of 1 kHz, an external reference is needed to compensate for the drift and to control all six DOFs of the quadcopter.

B. Cameras

As one of the goals of this work was to keep costs small, only webcameras were used and were connected to the PC via USB. Quickcam9000 Pro cameras from Logitech were used for this work. The cameras acquire images in VGA resolution at a framerate of up to 30 FPS. The field of view of $60°$ should be a good trade-off between accuracy and maximum possible as well as equally spaced flight volume. The maximum distance, at which the markers could be recognized properly was determined to be approximately 5 m in the experiments.

Accuracy is reduced when the cameras are aligned that their optical axes are parallel, which can be seen in Fig. 3: there is always an unused field of view of $\alpha - \alpha'$. Therefore, an alignment of the cameras with a distance of 1.4 m and a converging angle of $15°$ for each camera was determined to provide best results.
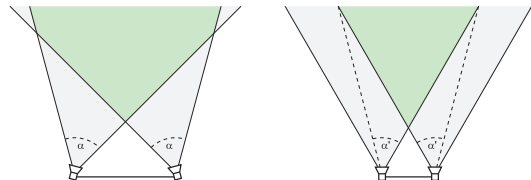

Fig. 3: Comparison between fields of view of converging and parallel cameras

---

[1]Ascending Technologies multi-rotor air-vehicles: http://www.asctec.de

[2]Digi networking solutions: http://www.digi.com

C. Markers

To be more independent from light conditions, active markers were mounted at the end of each boom of the quadcopter. The basic idea was to illuminate the markers as bright as possible and to reduce the exposure time of the cameras then. This method reduces motion blur as well as the influence of ambient light.

To compute all six DOFs of the quadcopter from the tracked marker-positions, three markers would be sufficient. For redundancy reasons and to avoid potential failure due to self-occlusion, four markers were applied. Two high power LEDs each were aligend inside a table-tennis ball dispersing the light of the LEDs diffusely. As a result, a homogeneously lit sphere could be recognized by the cameras as a circle from all directions of view. Fig. 4a and 4b shows the design and a lightened view of a marker.
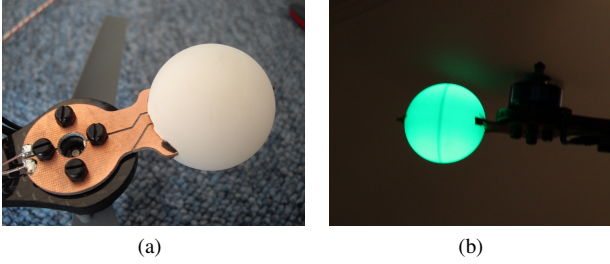

(a)   (b)

Fig. 4: Design of the markers

III. TRACKING AND POSE RECONSTRUCTION

A. Recognition of the markers

For each marker, the area and a center of gravity (COG) is calculated which is then used for computing the stereo correspondences needed for the 3-D reconstruction afterwards. The following steps are performed for each of the four markers in the image of each camera.

First of all, the image is transformed from RGB into the YUV-colorspace so that the search for colors can be performed independently from lighting conditions. Each pixel that falls within predifined ranges is called *hit*. Before computing the COG, a robust filter is applied: only *hits* whose coordinates in both x and y are close to the median of the coordinates will be considered in the calculation of the center of gravity. Compared to a median filter on the intensity values of a neighborhood of each pixel, the procedure of sorting the coordinates in order to calculate the median is not expensive: due to the line-by-line search, the y-coordinates are already sorted and the x-coordinates are presorted. Once a valid center of gravity is found, a region of interest (ROI) is determined around this position and search in the next timestep only takes place in this region.

In case a marker was partially occluded by parts of the quadcopter, the COG of a marker shifts although the marker

itself did not. The markers are bullets, so a circle in the binary image gained from the thresholds above is expected. To reconstruct the real center of the marker, the foreground of the binary image is corrected by morphological closing to fill gaps in the foreground but keep the basic structure approximately and edge filtering is done afterwards. Now, a circle can be fitted into the edges found in the step before. The deviations of the center of the markers could be reduced significantly using this method, especially on partially occluded markers. Fig. 5 shows step-by-step the procedure of recovering the circle.
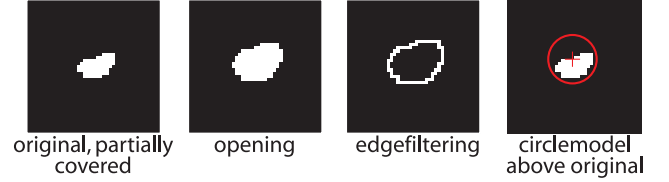

original, partially   opening   edgefiltering   circlemodel
covered   above original

Fig. 5: Step-by-step fitting of the circle

B. 3-D reconstruction of the marker's positions

After the 2-D coordinates of the center of the markers have been successfully determined in each image, their 3-D coordinates can be recovered. The coordinates from markers of the same color now serve as a corresponding pair of coordinates for the reconstruction. A 3-D point in space is projected into the camera's image planes up to a scalar factor λ by the following equations [8]. All coordinates are represented as homogeneous coordinates.

$$\lambda \boldsymbol{x}_1 = \begin{bmatrix} K_1 & 0 \end{bmatrix} \cdot \boldsymbol{X}_0 = \Pi_1 \cdot \boldsymbol{X}_0 \tag{1}$$

$$\lambda \boldsymbol{x}_2 = \begin{bmatrix} K_2 R & K_2 T \end{bmatrix} \cdot \boldsymbol{X}_0 = \Pi_2 \cdot \boldsymbol{X}_0 \tag{2}$$

$\boldsymbol{x}_i \in \mathbb{R}^{3 \times 1}$ denotes the projection of $\boldsymbol{X}_0 \in \mathbb{R}^{4 \times 1}$ into the image plane of the left and right cameras through the projection matrices $\Pi_i \in \mathbb{R}^{3 \times 4}$ which consist of the intrinsic parameter matrices K_i as well as of the extrinsic parameters R and T. The frame of the left camera is taken as reference frame. Rewriting of (1, 2) leads to four constraints for the point $\boldsymbol{X}_0$ we wish to recover from both views:

$$\left(x_1 \underline{\pi}_1^{3T} - \underline{\pi}_1^{1T} \right) \boldsymbol{X}_0 = 0 \quad \left(y_1 \underline{\pi}_1^{3T} - \underline{\pi}_1^{2T} \right) \boldsymbol{X}_0 = 0 \tag{3}$$

$$\left(x_2 \underline{\pi}_2^{3T} - \underline{\pi}_2^{1T} \right) \boldsymbol{X}_0 = 0 \quad \left(y_2 \underline{\pi}_2^{3T} - \underline{\pi}_2^{2T} \right) \boldsymbol{X}_0 = 0 \tag{4}$$

Where $\boldsymbol{x}_i = \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^T$. $\underline{\pi}_i^{jT}$ are the jth row vectors of the projection matrices $\Pi_i = \begin{bmatrix} \underline{\pi}_i^1 & \underline{\pi}_i^2 & \underline{\pi}_i^3 \end{bmatrix}^T$. (3) and (4) now form a 4×4 homogeneous system of equations that has to be solved: $M \boldsymbol{X}_0 = 0$. In practice, this system does not have a solution due to small errors in the correspondences or in the camera calibration. The best solution in the sense of least squares for $\boldsymbol{X}_0$ is the eigenvector of $M^T M$ corresponding to its smallest eigenvalue. To complete the reconstruction, $\boldsymbol{X}_0$ has to be normalized so that its last coordinate is equal to 1.

C. Reconstruction of the quadcopter's pose

Having reconstructed the 3-D positions of the markers, the position and orientation of the quadcopter can be determined. In [9], Umeyama proposes an approach to determine the rotation R and the translation T from two given sets of points X_i and Y_i where the mean squared errors $e(R, T)$ are minimized.

Y_i were chosen as initial positions of the quadcopter's markers. As the positions of the markers were reconstructed w.r.t. the left camera's frame, this frame was also taken as reference frame for the initial positions. These initial positions have the coordinates of the markers mounted at the end of the quadcopter's booms and lie in the x-y plane. The transformation between the initial positions and the current positions of the markers can be described as follows:

$$X_i = R \cdot Y_i + T \qquad (5)$$

To gain R and T uniquely in 3-D space, at least three correspondences are necessary. Firstly, the rotation will be calculated. The sets of points are translated by their centroids μ_x and μ_y to the origin and the covariance matrix Σ_{xy} of the translated sets of points is computed. n denotes the number of corresponding pairs of points.

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} X_i \quad \mu_y = \frac{1}{n}\sum_{i=1}^{n} Y_i \qquad (6)$$

$$\Sigma_{xy} = \frac{1}{n}\sum_{i=1}^{n} (Y_i - \mu_y)(X_i - \mu_x)^T \qquad (7)$$

The best solution for R in the sense of minimizing the squared errors is now achieved by computing the singular value decomposition $\Sigma_{xy} = USV^T$ and setting the singular values equal to 1. If $\mathrm{rank}\,(\Sigma_{xy}) = 2$, which is always the case for only three corresponding pairs of points or more than three points that are coplanar, the transformation matrix itself is exactly determined, but not uniquely if it is a rotation or a reflection. This can be checked by calculating $\det(R)$. Then, if necessary, S has to be corrected:

$$R = U\tilde{S}V^T \qquad (8)$$

$$\text{with } \tilde{S} = \begin{cases} \mathrm{diag}\,(1,1,1) & \text{for } \det(R) = 1 \\ \mathrm{diag}\,(1,1,-1) & \text{for } \det(R) = -1 \end{cases} \qquad (9)$$

Having calculated the rotation matrix, the translation vector can now be gained by simply rewriting (5) and using the mean vectors μ_x and μ_y of the two sets of points:

$$T = \mu_y - R\mu_x \qquad (10)$$

Although the distinction between rotation and reflection was made through calculating the determinant of R, this decision caused problems in practice. For example, the roll angle calculated from the rotation matrix (see below) jumped between $0°$ and $-180°$ in certain cases. To avoid this effect,

a "helper point" normal to the plane spanned by the markers, was constructed for each the initial (Y_i) and current (X_i) sets of points, which is shown in Fig. 6. To ensure that this helper point is always located on the same side of the plane, the crossproduct of the vectors connecting the markers is always calculated in the same sense of rotation. This is done by the following steps, before computing R and T:

1) Span the vectors l_1 and l_2 between the markers (X_{front}, X_{right}, X_{back}, X_{left}):
   If all four markers were recognized:

$$l_1 = X_{front} - X_{back}; \quad l_2 = X_{right} - X_{left} \quad (11)$$

   If only three markers were recognized: sort the markers in the order front-right-back-left
   $\Rightarrow X_1, X_2, X_3$

$$l_1 = X_1 - X_2; \quad l_2 = X_1 - X_3 \qquad (12)$$

2) Calculate the vector normal to the plane:

$$n = l_1 \times l_2 \qquad (13)$$

3) Calculate the mean vector μ as in (6)
4) Calculate the helper point:

$$X_{helper} = \mu + n \qquad (14)$$
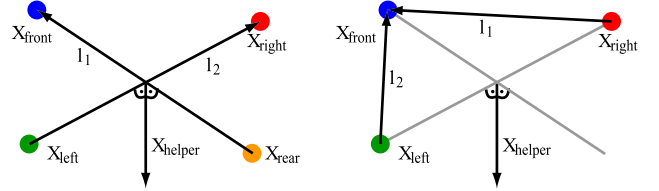
Fig. 6: Construction of the helper points

The covariance matrix Σ_{xy} calculated from X_i, X_{helper}, Y_i and Y_{helper} by (6), (7) has full rank and R will always be a valid rotation matrix. T can still be computed similar to (10).

Thus far, the rotation and translation of the quadcopter are still computed w.r.t. the left camera's frame. In most cases, the camera rig will be placed "somewhere" in a room, so the pose of the camera would not be a desirable reference frame for further applications such as position control. Thus, another transformation from the camera to world coordinates is required. The idea is to determine a new reference frame by the frame spanned by the quadcopter lying e.g. at its start position. The desired rotation/translation of the quadcopter w.r.t. its start position is given by:

$$R_0 = R_{QC,start}^T \cdot R_{QC,camera} \qquad (15)$$

$$T_0 = R_{QC,start}^T \cdot T_{QC,camera} - R_{QC,start}^T \cdot T_{QC,start} \qquad (16)$$

$R_{QC,camera}$ denotes the rotation w.r.t. the camera frame, $R_{QC,start}$ the start position (i.e. the new reference frame) and

R_0 the desired rotation w.r.t. the startposition, T respectively. When the camera rig's position changes, simply R_0 and T_0 need to be stored at the quadcopter's start position which are derived by the algorithms above.

IV. PERFORMANCE OF THE SYSTEM

After the implementation of the tracking of the quadcopter, its performance was evaluated before attempting position control.

A. Processing times and delays

To measure the total delay between a change of the quadcopter and a control reaction, processing times of certain algorithms were logged. In addition, the time between a change and the moment that the image is delivered by the camera driver was measured with a digital oscilloscope. As it can be seen in Table I, the largest part of the delay was caused by the cameras. This delay included an interframe delay of 33 ms on average or 66 ms at maximum, assuming 15 FPS. Compared to the communication- and camera delays, the other processing times were negligible. In the worst case, it takes $121\,\mathrm{ms} + 96/2\,\mathrm{ms} = 169\,\mathrm{ms}$ and in the mean $71\,\mathrm{ms}$ from a change in position to a response at the quadcopter.

TABLE I: Processing times and delays

	$\bar{t}$ [ms]	std(t)[ms]	t_{max} [ms]
load image from driver	0.5	0.4	1.9
LED-recogn., all LEDs	0.7	0.2	2.1
LED-recogn., three LEDs	5.8	0.6	7.3
LED-recogn., no LED	20.1	1.2	28.4
3D-reconstruction	< 0.1	-	-
position- and orientation	< 0.1	-	-
communication, roundtrip	55.1	18.8	96
camera delay	86.9	20.6	121

The Software was executed on a laptop with a 2.4 GHz Core2Duo processor and 3 GB RAM. CPU load with only 3 recognized LEDs at 15 FPS was about 25-30 % including 15 % needed by the cameradrivers to capture images. Memory usage was about 50 MB.

B. Pose accuracy

Absolute as well as relative accuracy was measured by mounting the quadcopter on a programmable 2D-positioning table with a traverse path of 600 mm in x- and 1200 mm in y- direction. The quadcopter could also be rotated around its z-axis. Trajectories were programmed and taken as reference. The experiments were repeated with the cameras' optical axis aligned parallel as well as converging at certain distances D to the positioning table. To measure the absolute accuracy, a rectangular trajectory that used the whole traverse paths was programmed.

Fig. 7 shows the measured values (blue) by the tracking system compared to the desired trajectory (red). The mean

absolute deviations to the trajectory $\overline{\Delta}$ were computed as a quality criterion which are listed in Table II.
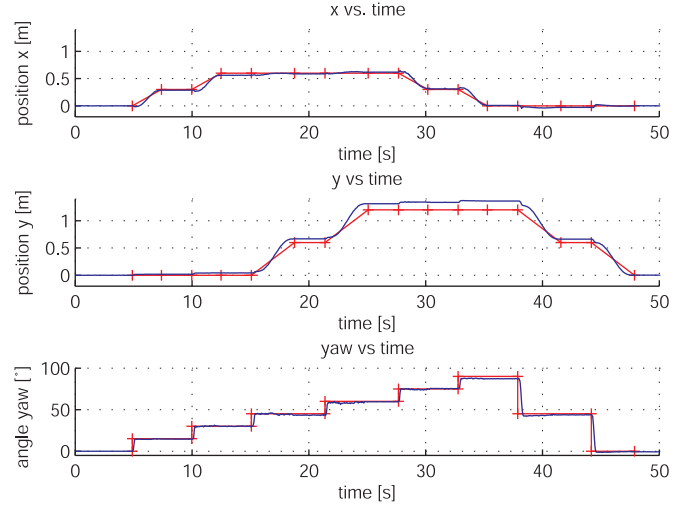
Fig. 7: measured values (blue) vs. desired trajectory (red)

TABLE II: Evaluation of absolute position accuracy

alignment	$\overline{\Delta}_x$ [mm]	$\overline{\Delta}_y$ [mm]	$\overline{\Delta}_{yaw}$ [°]
parallel, $D = 2\,\mathrm{m}$	43	102	1.1
parallel, $D = 4\,\mathrm{m}$	98	137	2.8
converging, $D = 2\,\mathrm{m}$	18	87	1.1
converging, $D = 4\,\mathrm{m}$	42	120	2.0

To measure how accurately small changes could be recognized by the system, the position was increased in only one axis by steps of 50 mm. This was repeated for each of the x- and y-axis and the camera alignments listed above. This time, the mean value $\bar{\delta}$ of absolute deviations to the desired step size of 50 mm was calculated over all steps, shown in Table III.

TABLE III: Evaluation of relative position accuracy

alignment	$\bar{\delta}_x$ [mm]	$\bar{\delta}_x$ [%]	$\bar{\delta}_y$ [mm]	$\bar{\delta}_y$ [%]
parallel, $D = 2\,\mathrm{m}$	4.3	8.7	1.8	3.5
parallel, $D = 4\,\mathrm{m}$	26.0	52.1	3.7	7.3
converging, $D = 2\,\mathrm{m}$	4.2	8.3	0.9	1.7
converging, $D = 4\,\mathrm{m}$	6.6	13.1	1.5	3.1

The convergent camera-alignment should be clearly preferred especially regarding the results of relative accuracy for larger distances. Overall, the position resolution is sufficient to stabilize the quadcopter on a desired trajectory. A resonable flight volume ranges from 1.5 m distance from the cameras to 4 m and up to 1.5 in height.

V. POSITION- AND HEADINGCONTROLLER

We show an exemplary design of a position controller to demonstrate the capabilities of the overall system. The

controllers are implemented on the high-level processor of the IMU (II-A) using a Matlab/Simulink SDK available for it.

A. Basic controller design and settings

Four DOF (x, y, z, yaw) must be controlled with the help of an absolute reference to achieve stable hovering. The IMU's high-level processor receives filtered and fused sensor data from the low level processor and can send control commands both at a rate of 1 kHz. All four controllers are implemented independently as PID (x, y, z) or PI (yaw) controllers. The interface of the low-level processor offers the following input signals:

Roll: roll-angle $\Rightarrow$ acceleration in y-direction
Pitch: pitch-angle $\Rightarrow$ acceleration in x-direction
Thrust: momentum of motors $\Rightarrow$ acceleration in z-direction
Yaw: angular velocity around z-axis

As no dynamic or aerobatic maneuvers should be performed, the control inputs were assumed to be linear to the resulting accelerations when the quadcopter is hovering (small angle approximation). The controller's differential components are computed from vision- data since we have no sensors to directly measure the speed. The integral component is non-linear, i.e. errors are not summed up when the quadcopter already moves towards its setpoint, so that overshooting is reduced.

For the yaw controller, only a P-term is needed to compensate for drift because the angular velocity was already controlled by the IMU. A small I-term was added to avoid constant errors. The P-terms of the x-, y- and z- controllers had to be chosen small until oscillations occurred due to the reduced bandwidth of the controller caused by delays induced by the cameras and data transmission. For the height or z-controller, an I-term is needed to keep the height constant at changing battery voltages or different payloads.

B. Improvements by using onboard inertial sensors

We were able to achieve stable hovering with the quadcopter with the controllers described above. Inertial measurements are provided by the IMU's low level processor at a rate of 1 kHz (section II-A). We show a simple approach to improve the controller's performance by incorporating this additional information. In order to achieve better damping of the fast dynamics of the vehicle and fast responses to disturbances, especially the speed-signal has to be improved. Therefore, we used the acceleration measurements as well as the differential terms from vision data as inputs for a complementary filter. The filter and controller is implemented on the onboard high-level processor running at a rate of 1 kHz. This avoids delays induced by wireless transmission to a ground station and back. Just pose information and setpoint commands have to be transmitted to the quadcopter at much slower rates from the ground station evaluating vision data.

Fig. 8: system structure

The system is now able to respond quickly based on the fast onboard velocity estimates whereas drift resulting from integrating noisy signals is compensated by the much slower, but absolute vision signal. Other filtering techniques like the family of Kalman filters to combine vision and inertial data would also be suitable and could further improve results, but are not implemented yet.

Important is the choice of the reference frame in which the quadcopter is controlled. We use the world frame as reference but omitting yaw, denoted by a subscript $\bar{0}$. This is possible, since yaw is already controlled by the low-level processor on the IMU that uses an estimated heading from the yaw gyroscope. With this approach, control outputs gained from faster IMU data can be mapped directly into appropriate motor commands. In contrast, control in the "pure" world reference frame (subscript 0) would require IMU data available at much higher rates to be transformed w.r.t. the heading (yaw) determined by the vision system. Control outputs in the x- and y- axis would then rely on the heading provided from vision that might be up to 50-100 IMU-timesteps old. Our controllers can now damp the vehicle's dynamics and can quickly respond to disturbances. In the slower vision loop, the absolute position errors gained from slower vision and setpoint commands are then rotated into the quadcopter frame to compensate for the remaining drift.

The final resulting controller structure can be seen in Fig. 8. The acceleration vector $\ddot{\boldsymbol{x}}_B$ is measured w.r.t the quadcopter's current attitude. Before integrating accelerations, this vector has first to be rotated from body to $\bar{0}$ - coordinates by $R_{\bar{0}B}$. This rotation matrix is determined from the pitch- and roll-angles estimated by the IMU. $R_{\bar{0}0}$ rotates the quadcopter's pose w.r.t. the world reference frame by the yaw-angle to $\bar{0}$. The "inversion" block transforms the acceleration commands into appropriate control commands for the quadcopter. In the current implementation, it is a linear block for small angle approximation.

C. Results

In the following graphs, the behavior of the system in certain situations is shown. Fig. 9 shows stationary hovering. Only slow oscillation with an amplitude of $\approx 15\,\text{cm}$ occur. Fig. 10 shows how disturbances at $t = 12.5\,\text{s}$ and $t = 3.5\,\text{s}$ in the x- and y-axis were handled. The large disturbance of the y-axis for example could be compensated within $5\,\text{s}$. Fig. 11 shows the behavior when following a trajectory. The graph of the yaw-angle is a lot more smother because the angle was controlled by the IMU at 1 kHz, so that our controller only needs to compensate for drifts (see also II-A). A video of the results can be seen at the author's website[3].
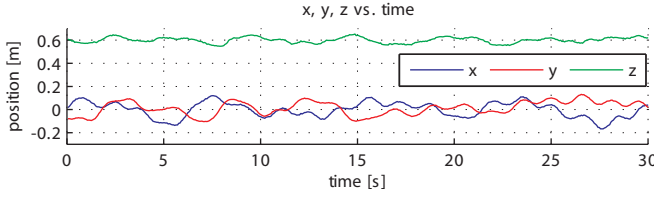
Fig. 9: stationary hovering
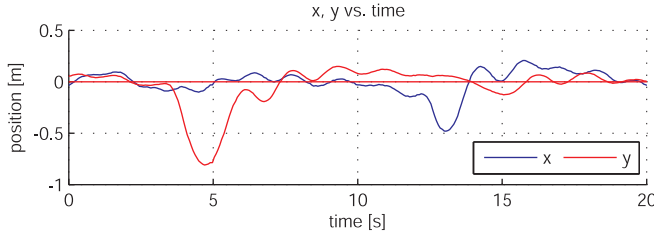
Fig. 10: disturbances of x- and y-position

VI. CONCLUSIONS

We presented an effective and reliable method for tracking a quadcopter or general objects with active markers. The tracking system is highly transportable, easy to set up and the can be executed on a laptop. We also showed that negative influence of the time delays on the performance of the controllers could be compensated by evaluating the onboard acceleration sensors of the quadcopter. In future, we plan to apply more than two cameras to gain a larger flight volume. Position control and sensor data fusion could be further enhanced by using techniques like the family of Kalman filters.

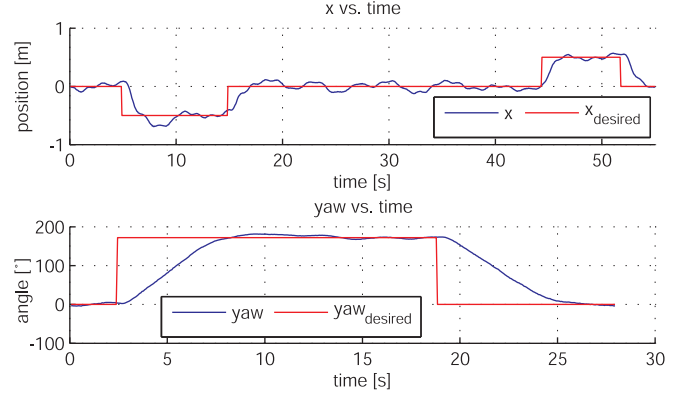[3]http://www.lsr.ei.tum.de/research/videos/robotics/qc-tracking-high

Fig. 11: following trajectories for x and yaw

REFERENCES

[1] P. Pounds, R. Mahony, J. Gresham, P. Corke, and J. Roberts, "Towards dynamically-favourable quad-rotor aerial robots," in *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 1 khz," in *IEEE International Conference on Robotics and Automation*, Roma, Italy, Apr. 2007, pp. 361 – 366.

[3] C. Lu, G. D. Hager, and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 610–622, 2000.

[4] R. Laganiere, S. Gilbert, and G. Roth, "Robust object pose estimation from feature-based stereo," *IEEE Transactions on Instrumentation and Measurement. Volume 55, Number 4. August 2006.*, vol. 55, pp. 1270–1280, 2006.

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

[6] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, Ecole polytechnique fdrale de Lausanne (EPFL), 2007.

[7] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environments," in *Proc. ICRA*, Los Angeles, CA, 2008, pp. 1814–1820.

[8] Y. Ma, S. Soatto, J. Kosecká, and S. S. Sastry, *An Invitation to 3-D Vision*. Springer Verlag New York, 2005.

[9] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, Apr. 1991, pp. 376–380.

[10] G. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2007.