

©2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

### **Progress**

- January 2012: Accepted for publication in *Proc. of the IEEE Intl. Multi-Conference on Systems, Signals and Devices, SSD12*.
- February 2012: Final version transmitted.

### **DOI**

Not yet available.

### **URL**

Not yet available.

### **Citation**

```
@INPROCEEDINGS{Lange12,  
  author = {Sven Lange and Peter Protzel},  
  title = {{Cost-Efficient Mono-Camera Tracking System for a Multicopter UAV  
    Aimed for Hardware-in-the-Loop Experiments}},  
  booktitle = {Proc. of the IEEE Intl. Multi-Conference on Systems, Signals and  
    Devices, SSD12},  
  year = {2012}  
}
```

# Cost-Efficient Mono-Camera Tracking System for a Multirotor UAV Aimed for Hardware-in-the-Loop Experiments

Sven Lange and Peter Protzel

Department of Electrical Engineering and Information Technology  
Chemnitz University of Technology, 09126 Chemnitz, Germany  
{sven.lange, peter.protzel}@etit.tu-chemnitz.de

**Abstract**—We describe a low-cost multirotor Unmanned Aerial Vehicle (UAV) testbed as a solution for typical challenges in system identification and control design which are dominant in early stages of multirotor research. Our main contribution is the development of an easy-to-use interface for rapid prototyping of control strategies within Matlab Simulink by using a self-made mono-camera tracking system for global position and orientation measurements. The system works with active markers on the UAV for fast and robust tracking capabilities within a wide range of illumination conditions. Its accuracy is evaluated theoretically as well as in practical experiments including runtime considerations. Following the idea of an easy-to-use hardware setup to support smaller research groups without access to expensive motion capture systems, the implemented software will be published as open source.

## I. INTRODUCTION AND RELATED WORK

One of our research projects focuses on enabling micro aerial vehicles to autonomously operate in GPS-denied environments, especially in indoor scenarios. Autonomous flight in confined spaces is a challenging task for UAVs and calls for accurate motion control as well as accurate environmental perception and modelling. The design of accurate motion control can only be achieved by knowing the specific system model and system parameters of the UAV at hand. Therefore, a global position and orientation measurement system is needed to realize a system identification and continue with further work in designing suitable control algorithms. Again, for evaluating the control performance, a ground truth measurement is needed.

Our paper explores the possibility to use a single camera in combination with active markers for recording ground truth data as basis for further work. The tracker can be used in a variety of research projects where ground truth pose information is necessary. Possible applications reach from system identification and control design to evaluation of visual odometry and sensor fusion. Our Matlab Simulink interface provides additional capabilities for fast workflow and, in particular, real-time pose information of the tracking system. So, our Simulink blocks can be effectively used for hardware-in-the-loop experiments.

This work has been funded by the European Union with the European Social Fund (ESF) and by the state of Saxony.

Our work is mainly inspired by [1] and [2]. The authors of [1] present a visual tracking system based on identically constructed active markers. Likewise they are aiming to build a cost-efficient tracking system with an easy setup. However, their system uses two webcams in a stereo camera setup to reconstruct pose and position of the quadrotor. We argue that the system can easily be simplified by using a single camera only to get an even more transportable system without the time-consuming calibration of a stereo camera system. In addition, we have implemented the interface to Matlab Simulink which was inspired by the authors of [2]. They used ROS [3] in combination with the ROS-Matlab bridge [4] to interface a VICON motion capture system [5] with Matlab for their experimental evaluation. Concerning the expensive motion capture system they used, we argue that the precision and working space provided by our low-cost system are sufficient for many applications. Furthermore, the source code of our system will be provided.

## II. HARDWARE

Multirotor UAVs have become very popular in the research community over the last few years. The availability of UAVs at affordable prices and a growing open source community may play a significant role. We decided to use a commercially produced system from Ascending Technologies for our research, so we don't have to worry about low-level control and hardware design. Hence our testbed is designed for use with a *Hummingbird* [6] or a *Pelican* UAV, but it can be used for any kind of small multirotor vehicles as well.

In Fig.1, an overview with focus on all involved hardware components of our system is shown. A detailed description of each component will be given in the following.

The *camera* is the main component for successfully detecting the active markers. We are using a uEye UI-5240SE color camera from the company IDS-Imaging. One key feature is the GigE connectivity for fast image transfer with a maximum image frequency of 50 Hz resulting in a sufficiently high measurement frequency. The camera has a resolution of  $1280 \times 1024$  pix with the capability of vertical and horizontal pixel binning for reducing the image size and increasing the sensor's sensitivity to light. A global shutter and the possibility to measure the exact time an image was taken – by a

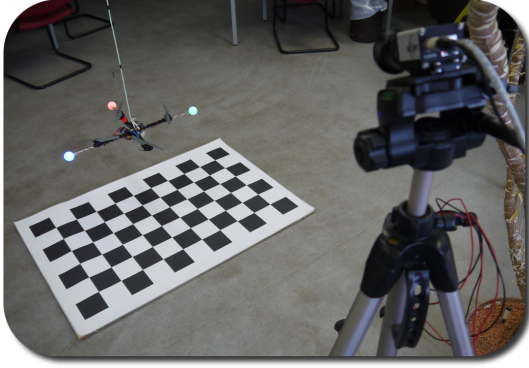


Fig. 1. The image shows the main parts of the tracking system. The Ethernet camera with tripod is visible on the right side of the image. A checkerboard is used to initially calculate the rotation and translation of the camera relative to the ground level. Additionally the Hummingbird quadcopter can be seen above the checkerboard. It is equipped with the three differently coloured active markers.

synchronized internal clock – are two other important features for getting measurements of the high dynamic quadrotor movements. By mounting a CS-mount lens with a focal length of 3.5 mm, we get a horizontal and vertical field of view (FOV) of 85° and 75°.

A common Intel® Core™2 Quad 2.66 GHz desktop PC is used as ground station running all processes like image processing, ROS nodes and Matlab. For serial communication with the UAV, we use an XBeePro radio module.

The *Hummingbird* and *Pelican* UAV we use for our experiments is equipped with three differently coloured *active markers*. Each marker consists of one LED with a luminous flux between 2lm and 5.5lm depending on its wave length and 360° angle of radiation. A diffusor is used for spreading the LED's light equally, so the marker can easily be detected within the image. As the authors of [1], we used ordinary table tennis balls. Since the active markers are emitting light in a constant intensity, the camera's automatic gain and exposure modes are deactivated and the exposure time is selected manually. In this way we ensure the robust marker detection within the camera image.

We mounted the second *XBeePro* radio module onto the UAV and connected it with the UAV's low level processor (LLP) for serial communication with the ground station.

### III. TRACKING SYSTEM

Besides the already presented hardware components, the tracking system consists of several software components as outlined in Fig.2. The following sections provide an overview of these components including their working principle.

#### A. Image Processing

As a first step we need to find the correct locations of the markers' projections within the image. Therefore, we transfer the image into the HSV space. Then, we use specific thresholds for every marker and every HSV plane to get three binary images for each marker. For each marker, we link the binary

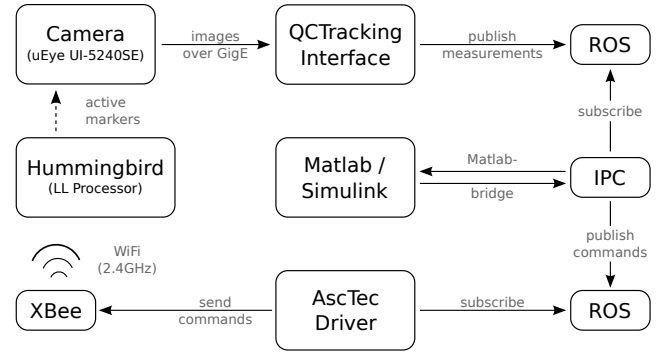


Fig. 2. The diagram outlines a complete system overview including the flow of information. Our QCTrackingInterface calculates the position and orientation measurement, while ROS and IPC is used for communication with our special Matlab Simulink blocks. The blocks can be used both ways, for receiving the measurement information and sending information back into the ROS framework. Finally, the AscTec driver relays the generated control commands to the quadrotor.

images by a logical *and* operation and do a morphological opening to reduce image noise. The remaining areas in the linked binary image are labeled and analyzed separately. Each found area is only considered as marker, if the following assumptions are met:

- area has a minimum number of pixels
- the roundness value of the area is bigger than a certain threshold
- the radius of the area lies within a minimum and maximum value.

If more than one candidate is found, the candidate which is located nearest to the last found marker position will be selected.

For runtime considerations, we optimized the image processing by introducing search windows for each marker. The idea is to process only a little part of the image, where the marker was found recently. Each search window will constantly grow in size, unless the marker is found in the current image or the search window already covers the whole image. This will be the case, if no marker is found within five successive iterations. On the other hand, the window will collapse to its smallest size again, if the marker is found.

#### B. 3D to 2D Correspondence

While the image processing step is giving us 2D information of the marker positions within the image, we are more interested in the markers' 3D pose. This task is also known as the three-point-perspective-pose estimation problem [7] or perspective-three-point (P3P) problem, where we have three non-collinear world points, which are forming a triangle with known dimensions, and their corresponding image points. The task is to calculate the pose of the triangle within the camera frame. To accomplish this, we need at least six constraints to get a solution for the six free parameters of the orientation and position. Each point-to-image correspondence will provide us with two constraints [8] and will lead to a maximum of

four possible solutions for the P3P problem. Which camera-to-triangle configuration will cause a specific number of ambiguities, is explained by the authors of [9].

For calculating the camera pose, we used the PnP algorithm provided by the computer vision library OpenCV [10]. This iterative method can be used not only for the above stated P3P problem, but also for the more general PnP problem where we have more than three correspondences. The used algorithm minimizes the reprojection error between the observed image points and the projected image points. Regarding the ambiguity of the solution it is necessary to set a good initial guess for the pose, and as long as the markers are tracked consecutively, the optimization will not run into a wrong solution.

As the initial guess is important for a correct solution of the PnP optimization, we used the closed form equations of [11] to get all possible solutions. Afterwards we can rule out the three wrong solutions by calculating the reprojection error and by using some constraints of our system, e.g. the quadrotor's z-axis will always point more or less towards the ground because we don't want to fly loopings.

As we are interested in the world positions and orientations of the tracked device, we need to calibrate the camera pose within the world frame. This can be easily realized by using the same PnP algorithm while positioning a checkerboard within the field of view of the camera as can be seen in Fig.1.

### C. Theoretical Accuracy

Since we want to use the tracking system as ground truth sensor, we need some information about the systems accuracy. Unfortunately, we do not have a better tracking system for comparative ground truth measurements, so we decided to do some theoretical considerations for the expected standard deviation of the camera's pose through back-propagation [8]. Here, the camera's covariance matrix within the camera frame is given by  $\Sigma_{cam}^C = (J^T \Sigma_{uv,i} J)^{-1}$ .  $J$  is the partial derivative (Jacobian) of the projection function  $f$  with respect to the camera's rotation and translation and  $\Sigma_{uv,i}$  is the  $i$ -th image point's covariance matrix ( $i = 1, 2, 3$ ).

As the accuracy of our system is strongly depending on the relative position between camera and markers, we need to calculate the camera's covariance matrix for several positions within the working space. For every position we get a new Jacobi matrix for calculating the camera's covariance matrix. As we are interested in the pose accuracy within the world frame, we have to transform the covariance matrix. Therefore, we used the same camera pose as in our experiments. The results can be seen in Fig.3. According to this calculations, a typical position of one metre in front of the camera will lead to a position standard deviation of  $\sigma = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} = 0.5$  cm. As the distance is growing to two metres, the position accuracy will grow to  $\sigma = 1.5$  cm.

Besides the strong dependency on the position between camera and markers, the tracking system's accuracy depends strongly on the relative orientation between both components. If the camera is positioned directly above the UAV, there will

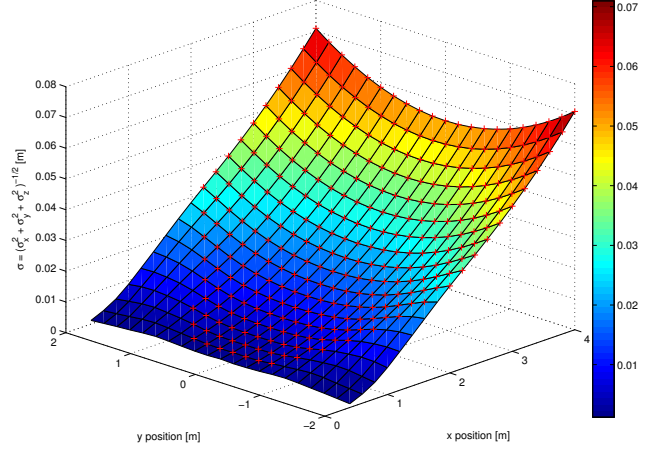


Fig. 3. The diagram shows the theoretically determined uncertainty in terms of the absolute one-sigma error for various positions of the active markers. Red crosses indicate the points within the camera's field of view. For comparison, the simulation parameters are chosen to match the real-world experiments closely: The camera has a tilt of  $35^\circ$  towards the ground plane and a height of 1.6 m above ground level. Additionally the intrinsic parameters of the real-world camera where used. The calculations are based on a pixel uncertainty of  $\sigma_u, \sigma_v = 0.5$  px and are carried out for several x and y positions within the world frame. The height of the active markers was set to a constant value of 1 m.

be a better position accuracy, but less accuracy in the orientation measurements. Likewise the orientation measurements are better, if the camera is positioned in front of the quadrotor. Then, the position accuracy will decrease, especially in the camera frame's z-direction.

### D. Communication

The tracking system provides its measurements through a ROS interface. Besides the current measurements of rotation, translation and velocity within world and body frame, it provides a timestamp. The timestamp is the exact time, the image was taken by the camera according to the camera's internal clock. This can become very handy, for example if the system is used for system identification.

## IV. MATLAB SIMULINK INTERFACE

We extended our visual tracking system by an interface for Matlab Simulink to get a rapid-prototyping system for testing new control designs. Basically, our system provides two different types of Simulink blocks. Type 1) offers access to ROS messages, so the block consists of output ports only. Type 2) offers the possibility to send ROS messages, so the Simulink block consists only of input ports. Currently, there are two blocks of type 1) and one of type 2). The latter is used for sending new control commands to the quadrotor via the AscTec driver [12]. One of the other blocks provides measurements of the UAV through accessing the AscTec driver and the other one provides the measurements of the tracking system.

### A. Working Principle of the Simulink Blocks

As ROS has no capability to communicate with Matlab directly, we have to use an additional communication layer. Here, the IPC bridge [4] offers us a way for linking a ROS topic with Matlab and vice versa. To get it working, we only need to write interface definitions for each ROS message we want to publish or to which we want to subscribe through Matlab. The interface definitions can be found in our repository.

The Simulink blocks itself are implemented as S-Function blocks with a constant and parameterizable sampling time. For communication with ROS, they integrate the IPC bridge functions for subscribing to a topic and listening to a topic. Additionally, type 1 blocks forward the ROS messages' timestamp for using it as delay measurements within Matlab. The timestamps can also be forwarded to the type 2 blocks to determine the whole runtime for one ROS message. For example, we get a new measurement of the Tracking System which has the exact timestamp of image capturing. This measurement is then forwarded into a Simulink model where we can incorporate the time delay into control algorithms or forward it again to use the time delay within the ROS framework. There, it can become handy to determine the time between measurement and resulting control command.

As real-time capabilities are essential for hardware-in-the-loop experiments, the model should provide it. Since this is not the default behaviour of a Matlab Simulink model, we used a simple soft real-time block [13] and added warnings, if the model's runtime is slower than real-time.

### B. Timings

The previously described time measurement capabilities can easily be used to get time delays of each of the tracking systems' component. Table I shows the result of several system iterations on a standard Intel® Core™2 Quad 2.66 GHz desktop PC. As can be seen, the biggest time delay of 30 ms is caused by the image acquisition and processing. The main reason can be found within the image transfer time from camera to PC. Additionally, the sampling time of the Simulink blocks determines a big part of the overall runtime. For transferring new commands to the UAV, time measurement was not possible, but it should be at least 3 ms taking the packet size and serial communication bandwidth of 57 600 baud into consideration.

## V. RESULTS

In the following two experiments, we will demonstrate the system's capabilities in real-world conditions. Therefore, we performed an experiment for validating the system's accuracy and an experiment for demonstrating the functionality of the whole system. Additionally, we describe another use-case in [14], where we used the system for evaluating a new optical flow based Extended Kalman Filter (EKF) design for our quadrotor system.

TABLE I  
DELAY TIMES, BROKEN DOWN TO THE SYSTEM'S COMPONENTS

Task	Delay
Image acquisition and processing	30 ms
Make measurements available as IPC message	< 1 ms
Measurements are provided as block output (worst-case is 10 ms, corresponding to the block's sampling time)	< 10 ms
New commands are processed and sent as IPC message via input block (delay corresponds to sampling time)	10 ms
Make commands available in ROS	< 1 ms
Send commands to UAV (over XBee) (theoretical transmission time, best-case)	> 3 ms
<b>In total</b>	$\approx 50$ ms

### A. Accuracy of the Tracking System

Besides the calculation of the tracking system's theoretical accuracy (see section III-C), we wanted to validate the real-world performance. This is hard to achieve without a superior measuring system, so we decided to use a Pan-Tilt-Unit (PTU) for approaching several poses with the high accuracy of the PTU. We used a PTU-D46-70 from *FLIR Motion Control Systems* with a resolution of  $0.013^\circ$  and mounted the quadrotor on top of it, as can be seen in Fig.4. With this setup, we are able to approach known poses and calculate the end effector's pose by applying forward kinematics. This was done by implementing a model of the PTU within the *Robotics Toolkit for Matlab* [15]. This model is shown together with the other components in Fig.5.

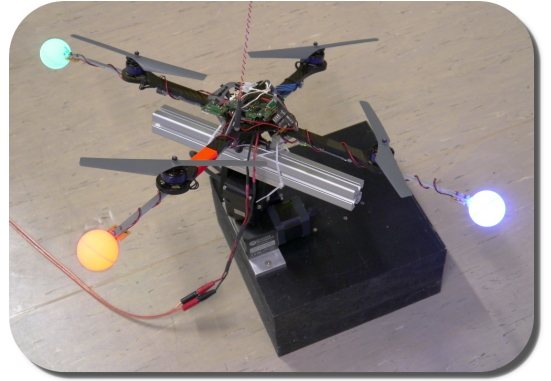


Fig. 4. Complete setup for system performance evaluation. The quadcopter is mounted on top of the PTU-D46, which is used for high precision approach of several poses.

The downside of the described setup is that we need to calibrate frame transformations of the whole system (see table II) to extrapolate the markers' pose within the world frame. Of course, we can measure all parameters manually, but this is difficult and inaccurate, especially for the rotations. So we decided to use the *Manifold Toolkit for Matlab* [16] to get the calibration parameters.

Unfortunately, it is inevitable to use the tracking system's measurements itself for doing the calibration, as can be seen



TABLE II  
INVOLVED TRANSFORMATIONS FOR ACCURACY EXPERIMENT.

Parameter	Transformation between	Value
$\mathbf{T}^{WPb}$	const. world and PTU base	unknown
$\mathbf{T}^{PbPe}$	dyn. PTU base and end effector	forward kinematics
$\mathbf{T}^{PeB}$	const. PTU end effector and body	unknown
$\mathbf{T}^{WB}$	dyn. body and world	tracking system

in the involved transformations. However, this is the only way we can think of for getting some clue of the performance.

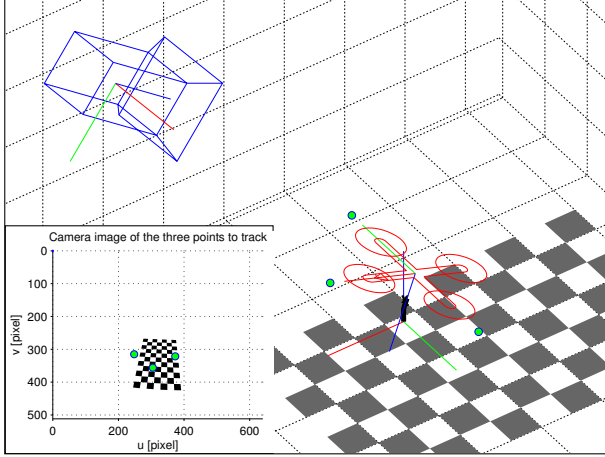


Fig. 5. Schematic view of the experiment with camera, PTU, checkerboard as used for calibration, quadrotor and the active markers. All poses are based on the real-world parameters of the experiment. The frame in the lower left side is showing the camera image.

For the calibration we used the markers' image positions as input and did a series of measurements with several pan and tilt configurations. The pan started at  $-40^\circ$  and ended at  $40^\circ$  with a resolution of  $20^\circ$ . For each pan configuration we traversed the tilt position from  $-20^\circ$  to  $20^\circ$  with a resolution of  $20^\circ$ . The camera parameters were determined previously and hold constant within the calibration procedure.

After calibration, we can use the transformation parameters and calculate the expected pose for a given PTU joint configuration. So the second measurement series is used for calculating the error between the predicted pose and the measured pose by the tracking system. This time we run through pan configurations from  $-40^\circ$  till  $40^\circ$  with  $1^\circ$  resolution and a tilt configuration from  $-20^\circ$  till  $20^\circ$  with  $1^\circ$  resolution and collected three measurements at each configuration. Histogram plots of the position and orientation error can be seen in Fig.6. In short, we reached a mean position error of 3.2 mm with a standard deviation of 2.4 mm and a mean orientation error of  $0.9^\circ$  with  $0.62^\circ$  standard deviation. Of course, the results depend on the PTU's position within the camera frame, as visualised in Fig.3. Here the distance between the quadrotor's starting position and the camera frame's origin was 1.6 m. If we calculate the theoretical accuracy as described in section III-C, we will get a standard deviation of

$$\sigma_{\text{trans}} = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} = 9.5 \text{ mm for the translation and } \sigma_{\text{rot}} = 1^\circ \text{ for the orientation.}$$

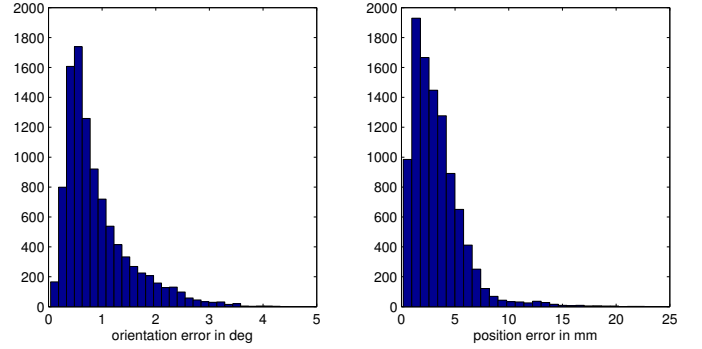


Fig. 6. Resulting error histogram of the accuracy experiment. Orientation error is shown on the left and position error on the right side.

### B. Velocity Controller

With this example, we want to demonstrate the capabilities of the whole system. Our aim is to design a velocity controller to stabilize the quadrotor within the tracking system's working space. As a little reminder: if we don't have a working velocity or position controller, the quadrotor will immediately drift into some direction, as we raise the thrust values. So we have to generate roll, pitch, yaw and thrust commands for the quadrotor to overcome this unwanted behaviour.

As already mentioned, we connected an XBee module to the serial interface of the quadrotor. This enables us to send the desired steering commands, as soon as we enable the quadrotor's automatic flight mode switch. This switch is a build in function of the quadrotor and provides a kind of safety system. As long as the switch is deactivated, the system will only use the commands given by the remote control.

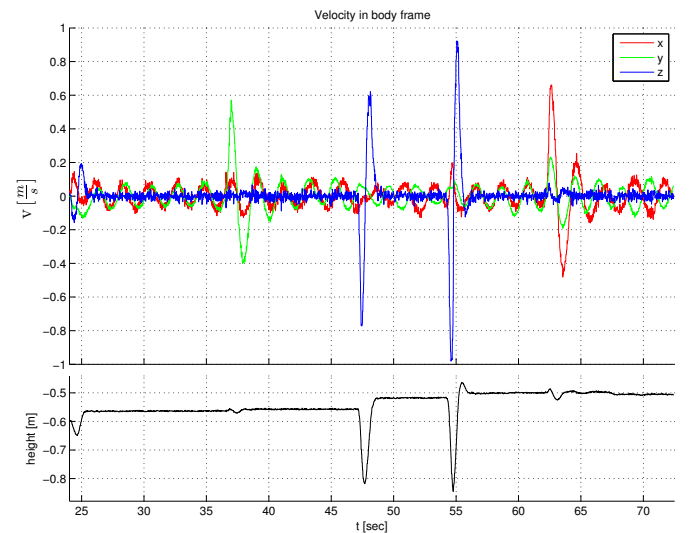


Fig. 7. Velocity (top) and height (bottom) plot of the velocity controller's results. We added four disturbances from manual pushing of the quadrotor, as can be seen around times: 37 s, 47 s, 55 s and 63 s.

For the controller's implementation, we used Simulink and our new communication blocks IPC $\leftrightarrow$ ROS, as can be seen in Fig.8. On the left side, we have the blocks for receiving measurements from the tracking system and the quadrotor's internal data and, on the right side, the output block publishes new control commands to a ROS topic. The quadrotor's internal data is only used to get the state of the previously mentioned autonomous flight-mode switch for enabling and disabling our controller.

The controller itself is built up of four independent PI controllers, where the controllers for roll, pitch and thrust use the body's velocity as input and the yaw component of the measured body rotation is used as input for the yaw controller. So, strictly speaking, the yaw controller is more a position controller.

For getting good controller parameters, we did a system identification with Matlab's *System Identification Toolbox* and determined the parameters through iterative optimization over the integral criterion of the system model's output.

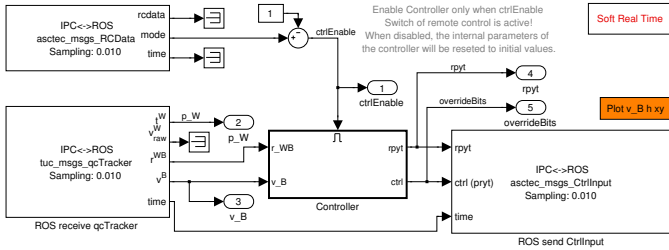


Fig. 8. Model of the simple velocity controller as used for the experiment. The *controller* submodel consists of four PI controllers for roll, pitch, yaw and thrust.

Finally, we tested the controller with and without external distortions. The first case's results are plotted within Fig.7. We pushed the quadrotor two times to the side and two times we pulled it upwards. As can be seen, the quadrotor is stabilizing its velocity very quickly. Only little oscillating behaviour remains, which is caused either by some time delay, not so optimal controller parameters or the nonlinear structure of the system itself. Obviously, we need a more sophisticated controller for better results. Anyway, the implemented controller works as expected. Without external distortions, we got a mean velocity error of  $|\bar{v}_x^B| = 52 \text{ mm s}^{-1}$ ,  $|\bar{v}_y^B| = 57 \text{ mm s}^{-1}$  and  $|\bar{v}_z^B| = 23 \text{ mm s}^{-1}$ . The standard deviation of the velocity is  $\sigma_{vx} = 76 \text{ mm s}^{-1}$ ,  $\sigma_{vy} = 82 \text{ mm s}^{-1}$  and  $\sigma_{vz} = 63 \text{ mm s}^{-1}$  over a time of 100s. All values are provided within the quadrotor's body Frame *B*.

## VI. CONCLUSIONS AND FURTHER WORK

Our work demonstrated and described a cost-efficient mono-camera tracking system and its performance. Regarding the measurement frequency and accuracy it is well suited for several applications where ground truth measurements are needed. Future work will concentrate on extending the system's working space by tuning some of its parameters and adding one or more cameras. As the current system is working

perfectly, it will be used for future work in control design, filter design and system identification.

The source code of the described system is available to the community as part of our ROS package at <http://www.ros.org/wiki/tuc-ros-pkg>.

## REFERENCES

- [1] M. Achtelik, T. Zhang, K. Kuhnlenz, and M. Buss, "Visual Tracking and Control of a Quadcopter Using a Stereo Camera System and Inertial Sensors," in *Proc. of the International Conference on Mechatronics and Automation, ICMA09*, 2009, pp. 2863–2869.
- [2] D. Mellinger, N. Michael, and V. Kumar, "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors," in *Proc. of the International Symposium on Experimental Robotics, ISER10*, 2010.
- [3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.
- [4] N. Michael and B. Cohen, "ROS-Matlab Bridge," Website, 2011. [Online]. Available: <https://github.com/nmichael/ipc-bridge>
- [5] "Vicon Motion Systems," Website, 2011. [Online]. Available: <http://www.vicon.com/>
- [6] 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 *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2007.
- [7] R. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem," *Int. J. Comput. Vision*, vol. 13, pp. 331–356, 1994.
- [8] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.
- [9] W. J. Wolfe, D. Mathis, C. W. Sklair, and M. Magee, "The Perspective View of Three Points," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 66–73, 1991.
- [10] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [11] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR11*, 2011, pp. 2969–2976.
- [12] W. Morris, I. Dryanovski *et al.*, "asctec\_drivers, part of ccny-ros-pkg," ROS Package, 2011. [Online]. Available: [http://www.ros.org/wiki/asctec\\_drivers](http://www.ros.org/wiki/asctec_drivers)
- [13] G. Rouleau. (2012) Simulink Real Time Execution. Website. [Online]. Available: <http://mathworks.com/matlabcentral/fileexchange/21908>
- [14] D. Wunschel, S. Lange, and P. Protzel, "Motion Estimation for Autonomous Quadcopter Indoor Flight," in *Proc. of the IEEE Intl. Multi-Conference on Systems, Signals and Devices, SSD12*, 2012.
- [15] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®*, ser. Springer Tracts in Advanced Robotics. Springer, 2011, vol. 73.
- [16] R. Wagner, O. Birbach, and U. Frese, "Rapid Development of Manifold-Based Graph Optimization Systems for Multi-Sensor Calibration and SLAM," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS11*. IEEE, 2011, pp. 3305–3312.