Framework for Autonomous Onboard Navigation with the AR.Drone

Jacobo Jiménez Lugo and Andreas Zell
Cognitive Systems, University of Tübingen
Tübingen, Germany
Email: {jacobo.jimenez,andreas.zell}@uni-tuebingen.de

***Abstract*—We present a framework for autonomous flying using the AR.Drone low cost quadrotor. The system performs all sensing and computations onboard, making the system independent of any base station or remote control. High level navigation and control tasks are carried out in a microcontroller that steers the vehicle to a desired location. We experimentally demonstrate the properties and capabilities of the system autonomously following several trajectory patterns of different complexity levels and evaluate the performance of our system.**

I. Introduction

The popularity of the research on micro unmanned aerial vehicles has increased in the recent years. Quadrotors are a very popular choice among MAV platforms due to their robustness, mechanical simplicity, low weight and small size. Their agility and maneuverability have been demonstrated through triple flips [1], ball juggling [2] and aggressive maneuvering through windows [3]. These characteristics make the quadrotor an ideal platform for operations such as exploration, surveillance, search and rescue or even artistic performances.

Several quadrotor platforms are available for the research community. Commercial quadrotors from companies like Ascending Technologies[1] or Microdrones[2] are among the most frequently used in research due to their robustness, though they are expensive and the firmware is not always customizable. The other option for researchers is to develop their own platforms according to the application's needs. This idea was exploited in several projects, for example, the Stanford's STARMAC quadrotor testbed [4] created for multi agent experimentation or the pixhawk project from ETH Zurich [5] designed for on-board computer vision, which features a single board computer with a multi-core processor. Using an already available flight controller boards can speed up the process of building a quadrotor. Lim et al. [6] provide an overview of the characteristics of many open source platforms.

However, building a platform can be expensive and requires time and knowledge to design a flight controller. Low cost solutions are favored in scenarios with a high risk of losing the vehicle. For example, Tayebi et al. [7] exploited low cost gyroscopes for attitude estimation. Low cost sensors can be used for autonomous navigation of quadrotors. The infrared camera of the Nintendo Wii remote controller was used for autonomous take off and landing on a moving ground vehicle [8] and for following another quadrotor [9]. The camera tracked active markers on the targets to estimate the relative position of the quadrotor.

In 2010, the french company Parrot launched the AR.Drone, a $300 quadrotor that was developed for the mass market of video games. It quickly attracted the attention of the research community with features such as on-board cameras and stable hovering. The AR.Drone has become a popular platform for research and education. It has been used for object following, position stabilization and autonomous navigation [10]. Faigl et al. [11] used the AR.Drone for surveillance. The AR.Drone's camera was used to learn a feature map of a determined area. In subsequent flights the feature map was used

[1] www.asctec.de
[2] www.microdrones.com

together with the AR.Drone's odometry to navigate through the area. Bills et al. [12] classified indoor environments based on images from the AR.Drone. They used visual cues inherent in the environment to fly through it. Engel et al. [13] demonstrated figure flying using images from the AR.Drone as an input for the parallel tracking and mapping (PTAM) algorithm [14]. The output of PTAM was fused with the vehicle's inertial measurement unit (IMU) data to produce a more accurate position estimate. The mentioned AR.Drone applications depend on a base station that extracts relevant information for path planning, navigation, control algorithms and generate a steering command. The base station communicates with the AR.Drone through a wireless network, which limits the working distance of the system and introduces a delay in the information exchange of sensor data and control commands.

The motivation behind this work is to extend the AR.Drone to enable autonomous flight and show that figure flying is possible with a low-cost system just over $300. The system does not need a base station, relying only on on-board sensing and processing, thus minimizing the communication delays.

We present a framework for autonomous navigation of the AR.Drone with minimal hardware and software additions. The framework can be easily customized or upgraded. We implemented high level control and trajectory planning tasks for autonomous figure flying. We evaluated our system in a series of indoor experiments.

The remainder of the paper is as follows. We provide an overview of the AR.Drone's hardware and software including vision and control algorithms in section II. We describe our system's architecture in section III. We present the evaluation results in section IV and conclude the paper in section V.

II. AR.Drone Platform

This section provides a brief overview of the AR.Drone quadrotor. A more complete description of the algorithms can be found in [15].

A. Hardware

The AR.Drone is a quadrotor with a size of 55 cm rotor-tip to rotor-tip and 380-420 grams of weight

Fig. 1: Low cost quadrotor AR.Drone version 1.0 and 8-bit microcontroller. The AR.Drone used for trajectory following experiments. The microcontroller is in charge of path planning and control tasks.

(see Figure 1). The central cross of the AR.Drone is made of carbon fiber tubes joined by a central plastic plate. A brushless motor is mounted at the end of each arm. The electronics are housed by a plastic basket on the center of the cross and an Expanded Poly Propylene body.
The electronic hardware comprises two boards connected to each other and attached to the plastic basket in the center of the cross to reduce vibrations. One board, the motherboard, contains the main processing unit (an ARM9 processor running at 468 MHz on AR.Drone 1.0 and an ARM Cortex-A8 at 1 GHz on AR.Drone 2.0), two cameras, a Wi-Fi module and a connector for software flashing and debugging. The cameras have different orientations. One is oriented vertically, pointing to the ground. It has an opening angle of 63 °, a frame rate of 60 frames per second (fps) and a a resolution of 320×240 pixels. This camera is used to estimate the horizontal speed of the vehicle. The second camera is pointing forwards. It has an opening angle of 93 °, a frame rate of 15 fps and a resolution of 640×480 pixels on version 1.0. Version 2.0 has an improved front camera that can deliver high

definition images at 720p resolution at 30 fps . Both cameras can be used for detection of markers such as stickers, caps or hulls of other AR.Drones.
The second board, the navigation board, contains all the sensors necessary for the state estimation of the quadrotor. The sensor suite includes a low cost inertial measurement unit (IMU) that updates the state of the AR.Drone at 200 Hz. The IMU consists of a 3-axis accelerometer, 2-axis gyroscope to measure angular velocities on pitch and roll axes and a more precise gyroscope to measure the angular velocity on the yaw axis. The distance to the ground is measured by an ultrasonic sensor with a maximum range of 6 m. The AR.Drone version 2.0 also equips a barometric sensor to measure its height beyond the range of the ultrasonic sensor. This measurement is also used to determine the scene depth in the images of the vertical camera and to calculate vertical displacement of the vehicle.

B. Software

The AR.Drone can be remote controlled via a smartphone or a tablet PC. It can also accept commands or send data to a PC via an AR.Drone API. The drone can send two types of data streams: a *navdata* stream that contains data related to the state of the vehicle and a *video* stream that provides encoded video from the cameras. The navigation data includes the status of the vehicle, motors and communications as well as raw and filtered IMU measurements, attitude (roll, pitch, yaw), altitude, linear velocities and an position with respect to take off point computed from visual information. The navigation data also includes information from the detection of visual tags. The received video can be from either of the two cameras or a picture-in-picture video with one of the camera images superposed on the top left corner of the other one.

The ARM processor runs an embedded Linux operating system that simultaneously manages the wireless communications, visual-inertial state estimation and control algorithms.

1) Navigation Algorithms: The AR.Drone provides sophisticated navigation algorithms and assistance in maneuvers such as take off, landing and hovering-in-position to ensure user's safety. To achieve a stable hovering and position control, the AR.Drone estimates its horizontal velocity using its vertical camera. Two different algorithms are used to estimate the horizontal velocity. One tracks local interest points (FAST corners [16]) over different frames and calculates the velocity from the displacement of these points. It provides a more accurate estimate of the velocity and is used when the vehicle's speed is low and there is enough texture in the picture. The second algorithm estimates the horizontal speed by computing the optical flow on pyramidal images. It is the default algorithm during flight. It is less precise but more robust since it does not rely on highly textured or high-contrast scenes.

The AR.Drone uses inertial information from its IMU for estimating the state of the vehicle. It fuses the IMU data with information from the vision algorithms and an aerodynamics model to estimate the velocity of the vehicle.

2) Control: The control of the AR.Drone is performed in a nested fashion. The innermost loop controls the attitude of the vehicle using a PID controller to compute a desired angular rate based on the difference between the current estimate of the attitude and the attitude set point defined by the user controls. The second loop uses a proportional controller to drive the motors.

When the controls are released, the AR.Drone computes a trajectory that will take it to zero velocity and zero attitude in a short time. This technique is designed off-line and uses feed-forward control over the inverted dynamics of the quadrotor.

III. System Architecture

Our system consists of two main components: a microcontroller and a proxy program that serves as a bridge between the AR.Drone and the microcontroller (Figure 2). The microcontroller we used is an ATMEL ATmega 1284 8-bit microcontroller. It runs at 14.7456 Mhz, has 128 Kb of memory, 32 I/O pins, counters and timers with PWM capabilities as well as two USART ports and a 10-bit analog to digital converter. It extends the capabilities of the AR.Drone to enable autonomous flying without the need of a ground station or any remote

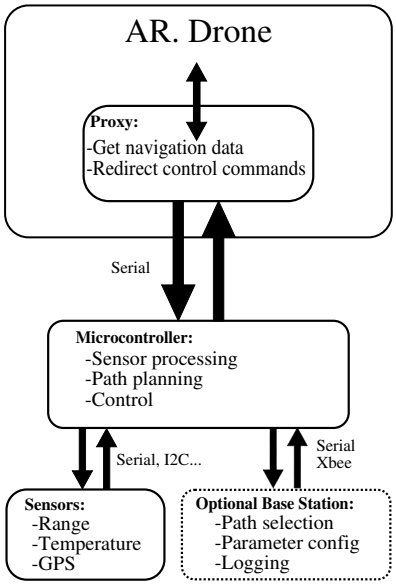


Fig. 2: System software components. The proxy application runs on the AR.Drone and enables exchange of information with the microcontroller. The microcontroller plans a trajectory using the position estimation of the AR.Drone and computes control commands for the vehicle. The microcontroller can interface with different. An optional base station communicates with the microcontroller to configure the flight and log flight data.

control device, thus widening its application range and eliminating delays that may be caused by the communication with the base. The microcontroller (Figure 1) is fitted inside the hull and plugged into the serial (debugging) port on the motherboard.

1) AR.Drone Proxy: We uploaded an external application to the AR.Drone to exchange information with the microcontroller, i.e. send steering commands, retrieve its status, sensor measurements, attitude and pose estimates. The microcontroller starts the application after the AR.Drone finishes its booting sequence. The application connects to the navigation and command ports of the AR.Drone main program and requests the navigation data (velocities, altitude, orientation and position from take off point). The application acts as a proxy: It listens on the navigation port for incoming data, sorts the data, packs it into a binary message and sends it to the microcontroller. At the same time it listens for AR.Drone's configuration or control commands coming from the microcontroller. These commands are directly forwarded to the AR.Drone's command port.

2) Microcontroller: To execute autonomous navigation and control algorithms on board, we attached the microcontroller to the debugging port of the AR.Drone. With these algorithms running on the microcontroller all the computational resources on the AR.Drone are reserved for its pose estimation and control algorithms.The microcontroller allows us to interface with a wide variety of sensors for example, ultrasonic or infrared range sensors, GPS, temperature sensors or radio communications. The number of sensors that can be added is limited by the microcontroller's specifications, their power consumption and the payload of the AR.Drone.

The additional hardware weights approximately 25 g. With the weight located at the center of the AR.Drone, there is no noticeable influence on flight performance with the outdoor protective hull and a 10 cm height loss on the hovering after the take off which the controllers are able to correct. With our current setup, flight time is reduced by approximately 2-3 minutes due to the extra weight and power consumption of the microcontroller.

The software framework for autonomous figure flying runs on this microcontroller. It consists of three modules: position estimation, path planning and control. The first module exploits the AR.Drone's position estimates from vision algorithms (see section II). The position and heading are provided in world coordinates with the origin at the take off location and the height value as the current measurement over the ground. The proxy application gets these data from the AR.Drone's navigation data stream. Then the proxy application builds a binary message with the AR.Drone's status, its position and heading and sends it to the microcontroller. We treat the AR.Drone as an additional sensor providing position information to the planning and control modules running on the microcontroller.

The second module, path planning, computes a trajectory for the AR.Drone to follow, based on predefined shapes or user defined waypoints. To follow a predefined shape, which can be a simple straight line, a circle or an eight shape, we calculated mathematical equations that parametrize the shape according to its size. To follow waypoints, we define a series of coordinates that outline a desired trajectory. Each waypoint w is defined as a triplet $w = [x, y, z]$ where x, y, z are in world coordinates with the origin as the take off location of the vehicle. The trajectory is computed as a series of smooth connected functions of time for each of the axes. The individual functions have the form $at + bt^2 + ct^3$ where the coefficients a, b and c are computed before the flight using the desired velocity of the vehicle. The yaw angle ϕ is computed so that the front camera of the AR.Drone points in the direction of the next waypoint. Navigating in this way keeps the camera pointing always in the direction of flight but is not optimal for the control of quadrotors. Yaw rotation produces centrifugal forces that push the vehicle away from the desired path requiring additional maneuvering to correct its trajectory.

In the third module, trajectory following control, we feed the AR.Drone's position estimate and the planned set point position to four PID controllers. The PID controllers compute commands for the roll, pitch and yaw angles and for the desired height of the vehicle independently from each other. These commands are then sent back to the AR.Drone via the proxy application that redirects it to the vehicle's command port. The control gains for each controller were experimentally determined.

Path planning and PID controller parameters can be configured from an optional base station. The base station communicates with the microcontroller on the AR.Drone via a radio module. It records flight data and displays useful information from the vehicle like its status and battery level and a visualization of the path.

Fig. 3: Indoor environment for the path planning experiments. The room is equipped with a motion capture system to provide positioning ground truth. The figure of eight is marked on the floor to provide texture for the AR.Drone's vision algorithms.

IV. Experiments and Results

A. Experimental setup

To test the capabilities of our system and analyze its performance, we performed a series of autonomous flights. We used an AR.Drone version 1.0 communicating with the microcontroller as described in section III. Flight data are logged at a frequency of approximately 30 Hz on the base station that communicates with the microcontroller using a radio module. The flight experiments were performed in two different rooms, both equipped with a motion capture system that monitors the position of the vehicle at a frequency of 100 Hz with millimeter precision for accurate ground truth measurements. One environment was a small room where the motion capture system covers a volume of $3 \times 3 \times 2\,\mathrm{m}^3$. The room has a carpet that provides texture for the estimation algorithms of the AR.Drone. The second room covers a larger volume of $10 \times 12 \times 6\,\mathrm{m}^3$. This room lacks texture on the floor so we laid colored paper on the floor forming the figure of an eight to provide the texture (see Figure 3).

To test the trajectory following, we defined waypoints within a $2 \times 2\,\mathrm{m}^2$ area of the small room arranged to form different patterns. After loading parameters for the flight to the microcontroller, it calculated the trajectory to follow. We then commanded the AR.Drone to hover at 1.10 m height and then start the autonomous flight. The AR.Drone flew three figures with varying level of difficulty: a square, a zig-zag and a spiral. In the larger room,

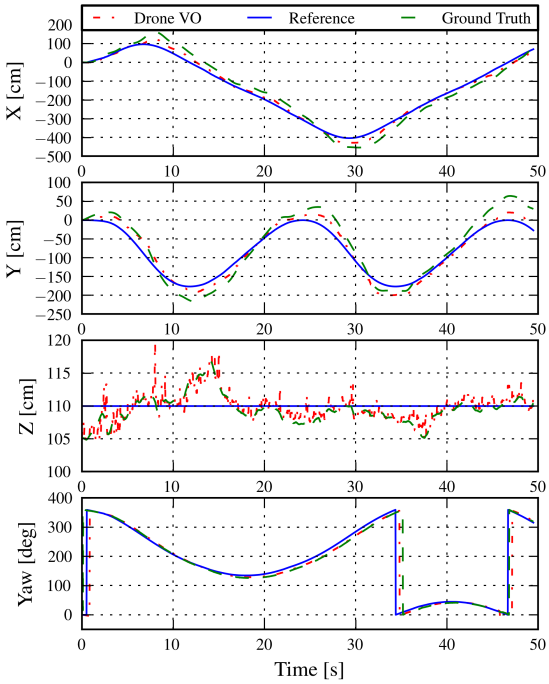


Fig. 4: Measurements from a typical trajectory following experiment. The AR.Drone was commanded to follow an eight shape trajectory of size 5 m × 1.75 m at an altitude of 1.10 m. Planned path (solid blue), AR.Drone's estimated position (dash-dotted red) and the ground truth (dashed green) are shown. Note that the yaw angle trajectory is smooth and the plot shows the wraparound at 360 °.

the AR.Drone was commanded to fly a predefined figure-eight of 5 m length and 1.75 m width over the marked figure on the floor of the room while keeping a constant height of 1.10 m.

B. Results

To evaluate the performance of our system, we analyzed the data collected from 20 flights. We first synchronized the data logged on the base station with the data provided by the tracking system. The data from both systems is synchronized by noting the start of the experiment (movement along x-y axis) on both data logs, the end of the AR.Drone's log and the total time of the experiment from these

TABLE I: Error analysis of AR.Drone's odometry

	Short trajectory			
	x [cm]	y [cm]	z [cm]	yaw [deg.]
RMSE	4.88	3.14	0.69	3.05
Std. Dev.	2.77	3.01	0.78	1.50
Max Error	11.81	10.32	6.74	6.40
	Long trajectory			
	x [cm]	y [cm]	z [cm]	yaw [deg.]
RMSE	8.04	12.82	1.21	9.20
Std. Dev.	6.04	11.57	1.18	5.80
Max Error	25.26	40.60	10.28	17.74

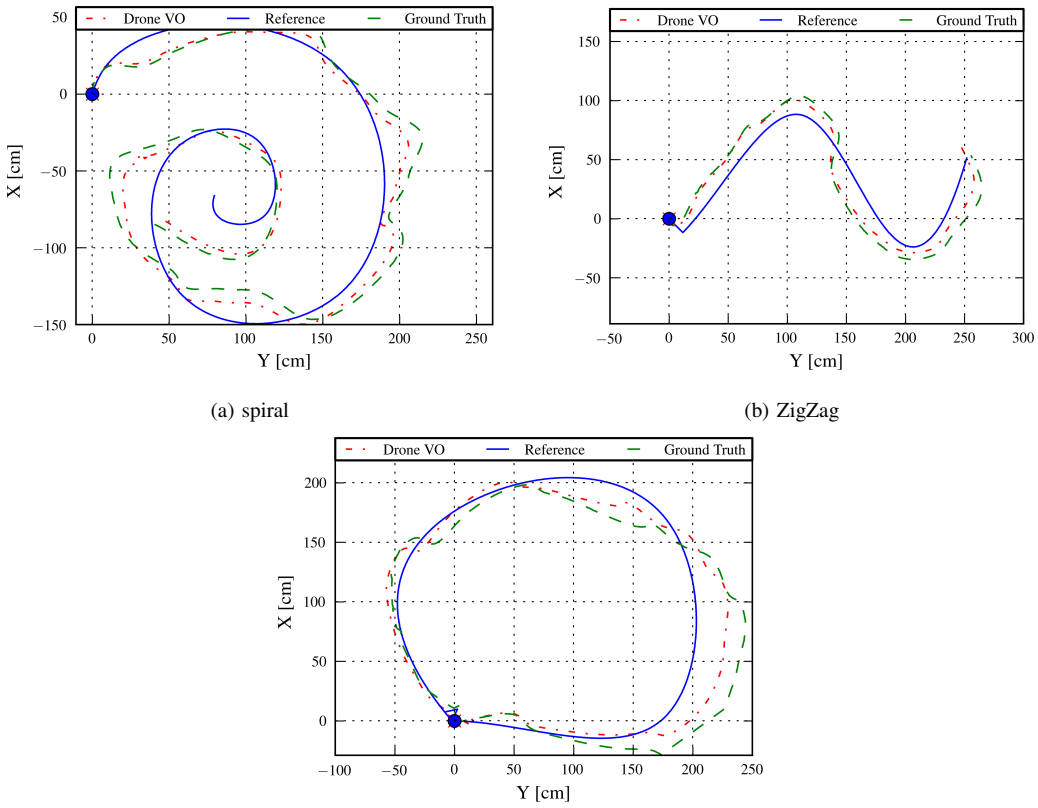
TABLE II: Error analysis of trajectory tracking

	Short trajectory			
	x [cm]	y [cm]	z [cm]	yaw [deg.]
RMSE	7.55	12.02	3.06	4.02
Std. Dev.	7.23	7.28	2.74	2.82
Max Error	25.7	29.7	11.30	13.90
	Long trajectory			
	x [cm]	y [cm]	z [cm]	yaw [deg.]
RMSE	7.58	11.34	3.14	5.93
Std. Dev.	4.91	6.90	2.04	3.34
Max Error	23.20	26.00	12.50	14.51

data and interpolating the tracking system's data at the time marked by the AR.Drone's data log timestamp. We measured the accuracy of the AR.Drone pose estimation by comparing it with ground truth data (see Table I). The root mean square error on the short trajectories was smaller than 5 cm and on the longer trajectory around 10 cm on x and y axes while on the yaw angle the RMSE was around 3 ° and 10 °, respectively. The difference in error values can be caused by drift in the sensors, which translates to a small drift of the position estimation. As it was expected, the height estimation from the ultrasonic range sensor yields comparable results in both experiments (around 1 cm). The errors and standard deviations in all axes showed that the AR.Drone's odometry provides enough accuracy for flying figures autonomously.

Figure 4 shows the measurements of the position and heading of the vehicle during a typical flight of the figure eight. A top view of one of the experiments where we flew the figure eight is shown in Figure 5. Figure 6 shows the top view of the spiral, zig-zag and square figures flown in the small room.

We analyzed the performance of the controllers



(a) spiral

(b) ZigZag

(c) Square

Fig. 6: X-Y projections of typical flights of the spiral, zig-zag and square figures flown in the small room. The different trajectories were computed from waypoints defined in the $3 \times 3 \times 2\,\mathrm{m}^3$ volume of the room.

for trajectory following by comparing the planned trajectory with the pose estimate of the AR.Drone (see Table II). The controllers performed very similar on both short and long trajectories as suggested by the RMSE and standard deviation values on all axes. There was a slightly larger error and standard deviation on the shorter flights, since the trajectories involved more changes in the heading direction of the vehicle than in the larger figure-eight, and in the case of the spiral, the changes were increasingly faster. With these properties, our controllers are capable of autonomous operation of the AR.Drone in reduced and narrow spaces with average errors 5 times smaller than the size of the vehicle and maximum errors of about half a vehicle length.

V. Conclusion

We presented a framework that allows the low-cost quadrotor AR.Drone to perform autonomous flights. Our method uses on board sensing only, and all the computations performed are on board as well. The system can be used for extended distances since it does not require any base station or remote control to operate. It can be easily customized and extended with additional sensors or a more powerful microcontroller that can speed up the computations or carry out more computationally challenging tasks.

We evaluated the system capabilities by following predefined trajectories and a series of user

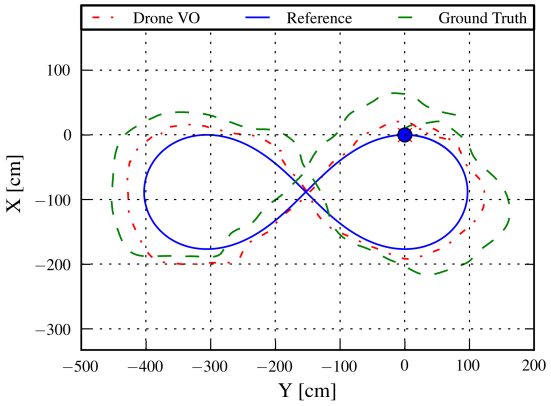


Fig. 5: X-Y projection of a typical eight shape trajectory flight In our experiments in the large room the AR.Drone followed an eight shape trajectory at a constant height of 1.10 m. The stating point is chosen so the AR.Drone starts with a yaw angle of 0 °. Planned path (solid blue), AR.Drone's estimated position (dash-dotted red) and the ground truth (dashed green) are shown.

defined waypoints. We used the AR.Drone's visual odometry which provided reliable estimates for this task. The trajectory tracking errors were small in comparison with the size of the quadrotor. This shows that our system is capable of flying in narrow environments.

We found that sometimes the visual odometry of the AR.Drone delivers position estimates with a very high error leading ultimately to a large deviation from the planned trajectory without the system being able to correct it. The problem was solved after resetting the AR.Drone. Another restriction is the limited computational power of the microcontroller which is insufficient to process the images of the on board cameras. This will be solved in the future by mounting a more powerful, lightweight single board computer.

Acknowledgment

The authors would like to thank Volker Grabe and Dr. Paolo Robuffo Giordano from the Max Planck Institute of Biological Cybernetics for letting us use their tracking system room and Karl E. Wenzel for his valuable input and advice with the AR.Drone.

References

[1] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," *IEEE International Conference on Robotics and Automation*, pp. 1642–1648, 2010.

[2] M. Muller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5113–5120, 2011.

[3] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive manuevers with quadrotors," *International Journal of Robotics Research*, 2012.

[4] G. Hoffmann, D. Rajnarayan, S. Waslander, D. Dostal, J. Jang, and C. Tomlin, "The stanford testbed of autonomous rotorcraft for multi agent control (starmac)," in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2. IEEE, 2004, pp. 12–E.

[5] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," *IEEE International Conference on Robotics and Automation (ICRA), 2011.*, pp. 2992 – 2997, may 2011.

[6] H. Lim, J. Park, D. Lee, and H. J. Kim, "Build your own quadrotor: Open-source projects on unmanned aerial vehicles," *IEEE Robot. Automat. Mag.*, vol. 19, no. 3, pp. 33–45, 2012.

[7] A. Tayebi, S. McGilvray, A. Roberts, and M. Moallem, "Attitude estimation and stabilization of a rigid body using low-cost sensors," *46th IEEE Conference on Decision and Control, 2007.*, pp. 6424 –6429, dec. 2007.

[8] K. Wenzel, A. Masselli, and A. Zell, "Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle," *Journal of Intelligent & Robotic Systems*, vol. 61, pp. 221–238, 2011, 10.1007/s10846-010-9473-0.

[9] K. E. Wenzel, A. Masselli, and A. Zell, "Visual tracking and following of a quadrocopter by another quadrocopter," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pp. 1–6, October 7-12 2012, accepted for publication.

[10] M. Saska, T. Krajník, J. Faigl, V. Vonásek, and L. Preucil, "Low cost mav platform ar-drone in experimental verifications of methods for vision based autonomous navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, 2012, pp. 4808–4809.

[11] J. Faigl, T. Krajník, V. Vonásek, and L. Přeučil, "Surveillance planning with localization uncertainty for uavs," pp. –, 2010. [Online]. Available: http://www.icr2010.org.il/

[12] C. Bills, J. Chen, and A. Saxena, "Autonomous mav flight in indoor environments using single image perspective cues," *IEEE International Conference on Robotics and Automation*, pp. 5776 –5783, may 2011.

[13] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[14] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[15] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the ar.drone micro uav," in *18th IFAC World Congress*, Milano, Italy, 2011, pp. 1477–1484.

[16] M. Trajkovic and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, no. 2, pp. 75 – 87, 1998.