Noname manuscript No.
(will be inserted by the editor)

A Vision-Based Quadrotor Multi-Robot Solution for the Indoor Autonomy Challenge of the 2013 International Micro Air Vehicle Competition

Jesús Pestana · Jose Luis Sanchez-Lopez · Paloma de la Puente · Adrian Carrio · Pascual Campoy

Received: date / Accepted: date

Abstract This paper presents a completely autonomous solution to participate in the Indoor Challenge of the 2013 International Micro Air Vehicle Competition (IMAV 2013). Our proposal is a multi-robot system with no centralized coordination whose robotic agents share their position estimates. The capability of each agent to navigate avoiding collisions is a consequence of the resulting emergent behavior. Each agent consists of a ground station running an instance of the proposed architecture that communicates over WiFi with an AR Drone 2.0 quadrotor. Visual markers are employed to sense and map obstacles and to improve the pose estimation based on Inertial Measurement Unit (IMU) and ground optical flow data. Based on our architecture, each robotic agent can navigate avoiding obstacles and other members of the multi-robot system. The solution is demonstrated and the achieved navigation performance is evaluated by means of experimental flights. This work also analyzes the capabilities of the presented solution in simulated flights of the IMAV 2013 Indoor Challenge. The performance of the CVG_UPM team was awarded with the First Prize in the Indoor Autonomy Challenge of the IMAV 2013 competition.

Keywords Aerial Robotics · Distributed Robot Systems · Multi-Robot Coordination · Visual Navigation · Quadrotor · Obstacle Avoidance · Mobile Robots · Remotely Operated Vehicles · MAV

1 Introduction

This paper presents a vision-based solution designed to participate in the 2013 edition of the International Micro Air Vehicle Flight Competition (IMAV 2013). These competitions are very relevant at the European level in the field of Autonomous Aerial Robotics and Small Remotely Piloted Air Systems (sRPAS). Another motivation for this paper is the development of an aerial multi-robot architecture that can be employed in different civilian applications.

The Computer Vision Group (CVG), our research group, was awarded for its participation in the IMAV 2012[3][29], showcasing the potential of the CVG in the field of autonomous Unmanned Aerial Systems (UAS). The learning experience and the development of common core software modules encouraged us to keep on working in the same direction and try a multi-robot approach in the 2013 edition. This edition's rules were significantly different with respect to the former edition's. At IMAV 2013 there was only one indoor competition, which could be addressed with different levels of autonomy. The scenario was a

The authors would like to thank the Consejo Superior de Investigaciones Cientificas (CSIC) of Spain for the JAE-Predoctoral scholarships of two of the authors, and the Spanish Ministry of Science MICYT DPI2010-20751-C02-01 for project funding.

Jesús Pestana · Jose Luis Sanchez-Lopez · Adrian Carrio · Pascual Campoy
Computer Vision Group,
Centre for Automation and Robotics, CSIC-UPM.
Calle Jose Gutierrz Abascal, 2; 28006 Madrid (Spain)
E-mail: {jesus.pestana, pascual.campoy}@upm.es
www.vision4uav.eu

Paloma de la Puente
Institute of Automation and Control,
Vienna University of Technology.

Pascual Campoy
Robotics Institute - TUDelft

[3] http://www.vision4uav.com/?q=imav12

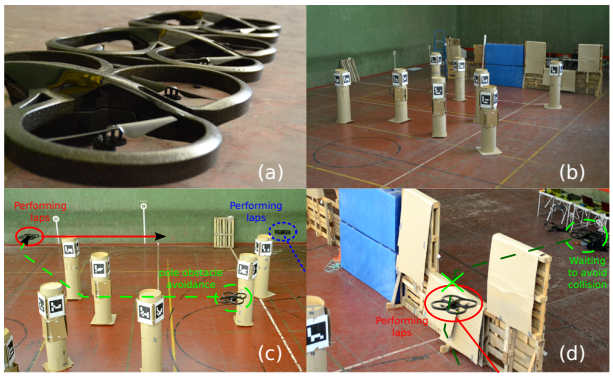

Fig. 1 The presented architecture has been designed for AR Drone 2.0 platforms. Figure (a) shows a picture of three of these quadrotors. The experiments are performed in a replica of the IMAV 2013 Indoor Challenge environment, shown in figure (b). This environment consists of a wall with two windows and eight poles. The position of the wall is previously known but not the location of the windows along it. The wall and the poles represent obstacles that must be avoided by the mUAVs while flying. Only the position of the four corner poles is previously known. All the poles and the windows are stamped using ArUco visual markers, see (b) and (c). Figure (c) shows an experimental flight at the moment when the quadrotors are crossing the unknown poles area. The unknown poles are robustly located in previous laps, when the drone performs laps around the known poles, ensuring a good estimation of their positions. The collision avoidance with other agents is solved at the trajectory planning step. For instance in (d), a drone has to wait until the path to cross the window is free. The flight shown in (c) is described and analyzed in Sec. 5. Videos of this and other flights are available at the website `http://www.vision4uav.com/?q=node/386` .

semi-structured environment with known obstacles, a wall and four fixed poles; and various obstacles located at unknown positions, four poles and two windows. The scoring tasks included: flying through a window, path following and precision landing, flying through an obstacle zone, and target detection and recognition, among others.

The first contribution of this paper is the implementation and successful exhibition of a vision-based quadrotor multi-robot solution for the IMAV 2013 Indoor Autonomy Challenge. The system is reliable, even though it is based on the low-cost AR Drone 2.0. Its weakest point is the dependency on a minimum available WiFi bandwidth to receive the video stream from each drone on their corresponding ground computers. The system was made scalable by leveraging the capabilities of the Robot Operating System (ROS). The second contribution is an analysis of the optimal performance of our multi-robot system to achieve the navigation tasks of the IMAV2013 competition. As a performance metric we decided to determine the maximum number of drones that can fly the mission without being a significant amount of time hovering in position, waiting for their next checkpoint to be free. Several simulations were run for this purpose, and the result of this analysis is presented in section 4. The third contribution is the successful experimental testing of a localization strategy using different kinds of data. In order to solve the localization problem, an EKF based algorithm formulation[31, 32] was adapted to work with multicopters instead of ground robots. This fact also required to adapt it to artificial 6D pose landmarks. The algorithm makes use of an initially known map and enlarges it when new landmarks are observed. In this case the odometry-based estimate is a full 6D pose instead of a horizontal position and a heading, and we use a very different sensing technology: filtered Inertial Measurement Unit, ultrasound altimeter and optical-flow speed measurements. The resulting odometry estimate is integrated with the detection of both a priori known and unknown visual markers to correct the drift of the odometry. The last contribution is that the presented architecture has been made publicly available as open-source[1]. For a more detailed analysis on the general design considerations of our architecture the reader is referred to the paper by Sanchez-Lopez *et al.*[37].

The layout of the paper is the following. In the first place, the state of the art and a description of the architecture are presented in sections 2 and 3. In section 3, the modules of our agent-level architecture are described. Secondly, a discussion about the optimum number of quadrotors that can be flown simultaneously in the IMAV 2013 competition environment using our solution is presented in section 4. Then, a experimental flight in a replica of the IMAV 2013 environment and a performance analysis are presented in section 5. Lastly, the conclusions and the future work are discussed in sections 6 and 7.

2 State Of The Art

Localization in indoor environments is a challenging task for UAVs, especially if a low cost and very lightweight solution is required[23, 36, 17, 9]. In the absence of GPS signal and heavier sensors such as laser or RGB-D sensors, visual approaches based on landmarks or visual markers are very popular[36, 17, 9]. The method presented by Jayatilleke and Zhang[17] requires all the landmark poses to be known a priori and only works in limited areas, making use of quite a simple approach without filtering of any kind. The work by Faig *et al.*[9] presents an interesting approach for local relative localization in swarms of micro UAVs, which requires

[1] `http://www.vision4uav.com/?q=quadrotor_stack`

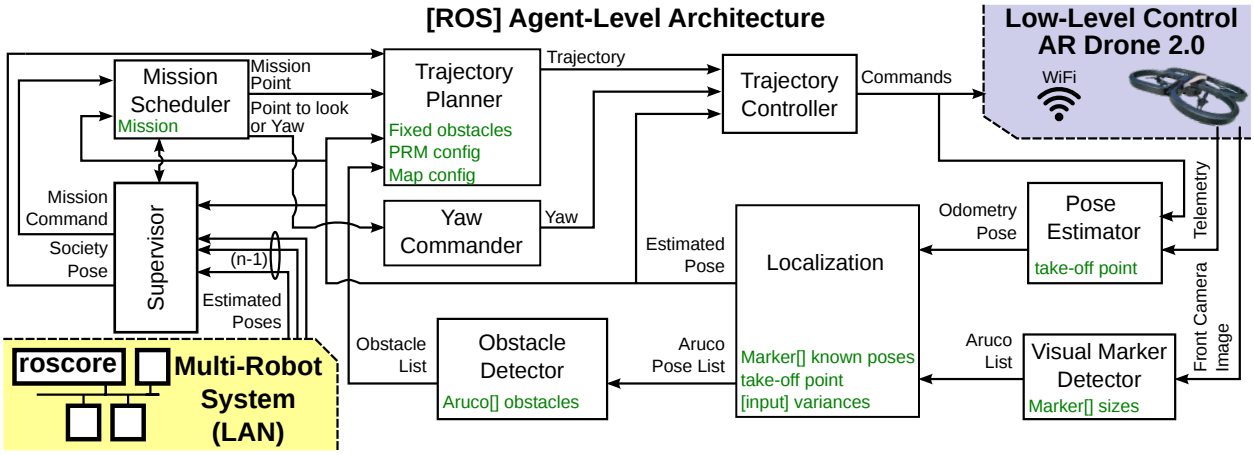

Fig. 2 Software architecture instantiated by each robotic agent. The architecture consists of several modules that communicate using the Robot Operating System (ROS) middleware. Each white box represents a module, and the green text specifies configuration parameters. The localization module fuses the odometry pose estimate with the detected visual markers' information, and broadcasts the estimated pose to the rest of the architecture and to the other robotic agents. The trajectory planner calculates collision-free paths avoiding collisions with obstacles and other members of the multi-robot system. The obstacle detector outputs the location of the unknown positioned poles as their ArUco markers are gradually mapped by the localization module. The trajectory controller receives the calculated paths and executes them or makes the drone hover in position, while making the drone look at an specific pole based on references delivered by the yaw commander. The mission scheduler module monitors the execution of the navigation tasks and delivers a predefined sequence of goal positions as they are attained. The supervisor monitors that all modules are online as well as the quadrotor's battery level. Additionally, it is in charge of the startup and poweroff sequences of the architecture. All the robotic agents' ground computers are connected through a Local Area Network (LAN) and communicate among each other using ROS. The communication between robotic agents is limited to sharing their estimated position, thus keeping the desired swarm behavior.

to keep external markers always visible. Our method was mainly inspired by the work by Rudol[36], but our models and formulation are quite different from those proposed by Conte[6].

Much research work is conducted using motion capture systems such as Vicon [2] to explore small UAV collaboration. The active and dynamic collaboration among a team of quadrotors is a subject of active development at the ETH Zürich Flying Machine Arena [3], where, for instance, they are able to throw a ball using a net held by a team of quadcopters[35] or make two quadrotors play, exchanging a pole[4]. Similar feats have been achieved by the GRASP Lab of the University of Pennsylvania [4] [5]. This Lab actively researches dynamic maneuvering[19] and quadrotor collaboration which is, for instance, demonstrated by having a team of quadrotors build structures[22].

Other research groups have focused on outdoor multi-robot systems, e.g. collaborative localization and mapping with multiple Micro Aerial Vehicles in unstructured environments[12]. The goal of the sFly FP7 European Project [6] was to develop small flying vehicles which can fly autonomously in city-like environments. As part of this project, strategies for coverage and surveillance of a target area were developed[33]. As a result of the sFly project a software framework to control Asctec quadrotors using Visual SLAM and IMU fusion was released as open-source[7][41, 42]. The Pixhawk project advertised a MAVmesh [8][24] concept to communicate quadcopters using reliable decentralized serial link communications, but the experimental testing of the concept has not been published. In a similar line, very recent ground breaking research has achieved decentralized multi-copter flock flight[40] in outdoor environments which is based on GPS usage. The flock can perform stable autonomous outdoor flights with up to 10 flying multirotors.

3 System Description

Based on the analysis of the contest rules, a localization system using visual markers and fusing with IMU and ground optical flow data was estimated to be robust enough against the inherent uncertainty of a competition environment. In addition, a solution that utilizes

[2] http://www.vicon.com/
[3] http://flyingmachinearena.org/
[4] https://www.grasp.upenn.edu/
[5] http://www.kumarrobotics.org/, 2014
[6] http://www.sfly.org/
[7] http://wiki.ros.org/asctec_mav_framework
[8] https://pixhawk.ethz.ch/software/mavmesh/start

low-cost quadrotors could leverage the usage of multiple drones to achieve a better score. Thus, a vision-based quadrotor multi-robot architecture was selected as our best option to participate in the IMAV 2013 competition. A multi-robot system composed by 3 or more relatively simple quadrotors is designed to achieve all navigation missions autonomously. Our solution requires a small number of human operators to start and monitor the whole system; who, in case of malfunction, can stop independent robotic agents.

The multi-robot system is composed by low-cost AR Drone 2.0 quadrotors, see figure 1(a). Each quadrotor is paired through a WiFi link with a ground computer to achieve an autonomous drone or robotic agent that can complete a predefined navigation mission while avoiding collisions with obstacles and other drones of the multi-robot system. The intelligence is implemented at the agent level without a centralized planning approach. The system displays thus a swarming or emerging behavior. All the drones perform self-localization in world coordinates by means of visual markers, and broadcast their estimated pose to the others. By using the other robotic agents' positions each one of them can plan collision-free trajectories during the execution of the mission.

Therefore, our multirobot-system is composed by identical robotic agents, who consist of an instance of the software architecture that commands an AR Drone 2.0 over WiFi, as shown in Fig. 2. The communications between modules on each ground station and also among robotic agents are implemented using the Robot Operating System (ROS) middleware[9], and the interface with the drones is achieved using the ardrone_autonomy ROS package[10]. A thorough description of the AR Drone 2.0 is given in[5]. This quadcopter is a popular platform for prototyping projects, which has been used by other research groups to perform autonomous navigation in unstructured environments[8, 26].

As explained, each drone is paired to a separate ground computer which runs a full-instance of the architecture depicted in Fig. 2. The communication with the drone is performed over WiFi. The communication between agents is achieved over a common Local Area Network (LAN) to which all ground computers are connected. One of the ground computers runs the *roscore*. This ROS program provides the necessary network addressing information that allows all our modules to seamlessly communicate with each other. In ROS, each module can be addressed separately by using a generic name and a *namespace*. We leverage this capability by employing one separate numbered namespace per drone, for instance *drone1*, *drone2* and *drone3*.

[9] Robot Operating System (ROS), `http://www.ros.org/`

[10] ardrone_autonomy ROS package, `https://github.com/AutonomyLab/ardrone_autonomy/`

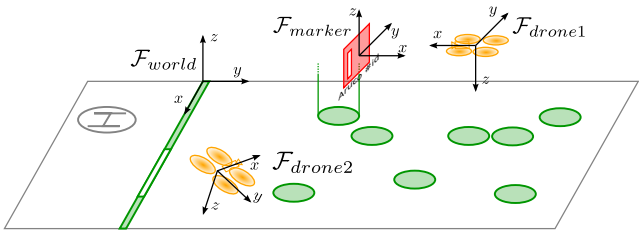

Fig. 3 Relevant coordinate frames of our proposed solution. The figure shows a simplified depiction of the IMAV2013 environment: the initial take-off position is shown in gray; the static obstacles, the wall and columns, in green; two drones in green and a marker in red. The world coordinate frame, $\mathcal{F}_{world}$, is located in one of the corners of the wall. The frames of the drones, $\mathcal{F}_{drone1}$ and $\mathcal{F}_{drone2}$; and the marker, $\mathcal{F}_{marker}$, are also shown.

3.1 Coordinate Frames

The IMAV2013 environment has several elements that need to be located with respect to the world frame for the drone to be able to plan obstacle-free trajectories. These elements are: the two wall sections, the four known and four unknown columns, the known and unknown visual markers and the other drones. The walls and columns are defined by their size and the horizontal location of their center point. The drones and visual markers require to be localized using a full 6D pose, composed of position and attitude. The coordinate frames of these objects is shown in Fig. 3.

In the rest of this article, the coordinate transformation from a frame A to a frame B, which transforms position coordinates $\mathbf{x}_A$ with respect to frame A to coordinates $\mathbf{x}_B$ with respect to frame B, will be denoted as $\mathcal{T}_{B\_A}$. $\mathcal{T}_{B\_A}$ can be represented as a 4 by 4 homogeneous transformation matrix that performs the coordinate transformation, $\mathbf{x}_B = \mathcal{T}_{B\_A}\mathbf{x}_A$. It is noted that pose measurements, which include position and attitude, are also estimates for a relative coordinate transformation between two frames.

3.2 Quadrotor Dynamics Model

Fig. 4 shows the free body diagram of a quadrotor. Other than aerodynamic effects, which are not considered on this model, the principal magnitudes that affect the quadrotor behavior are: the mass properties of the vehicle's body, and the forces and torques generated by the propellers. The following variables and symbols, introduced in Fig. 4, are later used in equations and cited

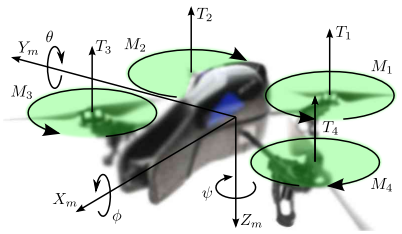

Fig. 4 Free body diagram of a quadrotor. The four propellers, and each of their performed thrusts T_i and torques M_i are labeled 1-4. The axes of the drone coordinate frame are $\{X_m, Y_m, Z_m\}$. The Euler angles of the drone coordinate frame with respect to the world coordinate frame are denoted $\{\phi, \text{roll}\}$, $\{\theta, \text{pitch}\}$ and $\{\psi, \text{yaw}\}$.

in the discussion: the pose, position and attitude, of the drone coordinate frame with respect to the world coordinate are specified by its position vector, denoted by the $\{x, y, z\}$ coordinates, and the attitude of the vehicle, represented by the $\{\psi, \text{yaw}\}$, $\{\theta, \text{pitch}\}$ and $\{\phi, \text{roll}\}$ Euler angles following the $z-y'-x''$ intrinsic rotations convention (also known as nautical angles); the quadrotor rigid body is characterized by its mass m and its three principal mass moments of inertia $\{I_x,\ I_y,\ I_z\}$; each propeller i generates a thrust T_i and a heading torque M_i; the l constant is the distance between the centers of each pair of opposite propellers, such us propellers 1 and 3 in Fig. 4; and each propeller rotates at speed ω_i and generates a torque of magnitude $l/2 \cdot T_i$ with respect to the quadrotor body that can tip the quadrotor body frontward, backward or sideways.

The quadrotor rigid body dynamics model, see Eq. 3.2 and Fig. 4, can be inferred from the laws of motion of the rigid body, as explained in multiple articles[16, 25]. In these equations $\{s\psi,\ s\theta, s\phi\}$ are the sine and $\{c\psi,\ c\theta, c\phi\}$ the cosine of the corresponding Euler angle.

$$\begin{cases} I_x\,\ddot{\phi} = \dot{\psi}\,\dot{\theta}\,(I_y - I_z) + \frac{\sqrt{2}}{2} l\,(+T_1 - T_2 - T_3 + T_4) \\ I_y\,\ddot{\theta} = \dot{\phi}\,\dot{\psi}\,(I_z - I_x) + \frac{\sqrt{2}}{2} l\,(-T_1 - T_2 + T_3 + T_4) \\ I_z\,\ddot{\psi} = \dot{\theta}\,\dot{\phi}\,(I_x - I_z) + (-M_1 + M_2 - M_3 + M_4) \\ m\,\ddot{x} = (-s\phi\,s\psi - c\phi\,s\theta\,c\psi)\,\sum_{i=1}^{4} T_i \\ m\,\ddot{y} = (-c\phi\,s\theta\,s\psi + s\phi\,c\psi)\,\sum_{i=1}^{4} T_i \\ m\,\ddot{z} = mg - c\theta\,c\phi\,\sum_{i=1}^{4} T_i \end{cases} \quad (1)$$

It is generally accepted to approximate T_i and M_i as $T_i = k_T \cdot \omega_i^2$ and $M_i = k_M \cdot \omega_i^2$, where ω_i, k_M and k_T are the propeller rotational speed and two propeller characteristic constants. However, it is important to note that in our solution the low-level stabilization of the attitude and altitude of the drone is achieved by the AR Drone 2.0 itself, and that our trajectory controller provides the corresponding commands directly to this low-level controller. These commands are: altitude speed $\frac{dz}{dt}_c$, pitch θ_c, roll ϕ_c and yaw speed $\frac{d\psi}{dt}_c$.

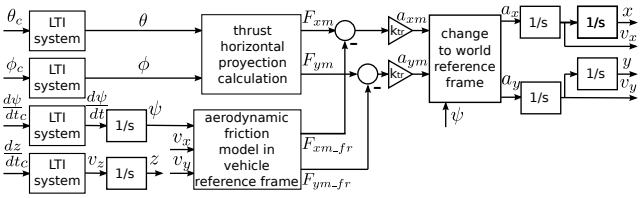

Fig. 5 Simplified model of an AR Drone multicopter proposed in[28], which includes the action of the control commands. F_{xm} and F_{ym} are the projection of the thrust vector on the horizontal plane, which are added to the aerodynamic friction force, F_{xm_fr} and F_{ym_fr} , generated by the movement of the main body of the vehicle. k_{tr} is a characteristic constant that sets the response time of the linear speed of the vehicle's body. $\{a_{xm}, a_{ym}\}$ and $\{a_x, a_y\}$ are the predicted acceleration in the mobile and world frames, respectively.

3.3 Agent-Level Modular Architecture

In this section, the main characteristics and technical aspects of each module of the agent-level architecture, depicted in Fig. 2, are introduced. In this section, *droneI* stands for one of the n_d drones flying on the current mission, and *markerJ* stands for one of the n_m markers placed on the competition environment.

3.3.1 Pose Estimator module

For *droneI*, the pose estimator, which is based on the EKF algorithm, estimates the coordinate transform $\{\mathcal{T}_{world\_droneI}\}_{odom}$, and speed estimates, $\{v_x, v_y, v_z, \frac{d\psi}{dt}\}_{odom}$, for the variables $\{x, y, z, \psi\}_{odom}$. However, these estimates are biased by the inherent drift related to odometry-based estimation. As explained later, the localization module calculates a bias-free estimate of $\{\mathcal{T}_{world\_droneI}\}$ by fusing only the odometry-based pose estimate $\{\mathcal{T}_{world\_droneI}\}_{odom}$ with the detected visual markers.

The employed measurement model is depicted in Fig. 5, and it is fused with the drone commands $\{\theta_c, \phi_c, \frac{dz}{dt}_c, \frac{d\psi}{dt}_c\}$, the filtered IMU measurements $\{\psi_m, \theta_m, \phi_m, \frac{d\psi}{dt}_m\}$, the ultrasound altimeter measurements $\{z_m, v_{zm}\}$, and the ground optical flow horizontal speed measurements $\{v_{xm}, v_{ym}\}$. The altitude and yaw speed measurements $\{v_{zm}, \frac{d\psi}{dt}_m\}$ are obtained by a careful numeric derivation of the altitude and yaw measurements $\{z_m, \psi_m\}$.

This pose estimator requires measurements from the ground optical flow sensor in order to work properly. It differs from other research in that it fuses the high-level commands and the filtered IMU measurements.

– In comparison to the work by Weiss *et al.*[41], they fuse the raw IMU measurements with visual-based pose measurements. These measurements are the direct readings from the three-axis accelerometers and

gyroscopes from the drone's autopilot, and require the estimation of the IMU biases during flight.
– Recent research[1, 21] has shown that the filtered IMU measurements in quadcopters are commonly biased due to the neglect of the propeller aerodynamic frictions which dominate any other aerodynamic friction during normal multicopter flight.
– Arguably, there are better physical grounds to fuse the propeller thrust and torque commands, but with the AR Drone 2.0 only the higher-level commands $\{\theta_c, \phi_c, \frac{dz}{dt}_c, \frac{d\psi}{dt}_c\}$ are known.

In spite of these differences, our odometry-based pose estimator has repeatedly allowed stable and precise quadrotor flight, and our research group has successfully used it in its participation on various International Micro Aerial Vehicle competitions. More details about the pose estimator can be found in the Master's Thesis[28].

3.3.2 Visual Marker Detector module

This module processes the images acquired by the on-board camera of the drone. Given the real size of the fiducial markers located on the environment, see the markers on the images in Fig. 1(b,c,d), it returns the poses of the detected markers with respect to the drone, $\mathcal{T}_{droneI\_markerJ}$. For known markers, these estimations are equivalent to a pose measurement with respect to the world frame, $\mathcal{T}_{world\_droneI}$. For unknown markers the localization module needs to additionally estimate the pose of the fiducial marker, $\mathcal{T}_{world\_markerJ}$. The type of visual markers used in this work are ArUco[14][11], whose implementation is open-source. A comprehensive comparison between various visual marker libraries was performed by Krajník *et al.*[18].

3.3.3 Localization module

Localization is achieved by an Extended Kalman Filter (EKF) Simultaneous Localization and Mapping module that calculates the complete 6 DOF pose of the drone by fusing the odometry based pose estimate and the measured pose with respect to the detected visual external markers. The map is initialized with the known 6D landmarks. The take-off point coordinates in a common global reference frame are used as initial localization pose.

In the prediction stage, an incremental motion model is used. The odometry estimate with respect to the previous iteration is composed with the current localization estimate by means of 6D pose relative transformation composition using homogeneous matrices. The covariance matrix is accordingly modified using the Jacobians of the composition.

In the correction stage, data association is not necessary since the external markers have unique ids. Recognized landmarks are used for setting up the Jacobian matrices to be used for the EKF correction update. A simple observation model in the Euclidean space is used with subsequent angle normalization, a manifold representation[10] could be tested in future work. Once the best current estimate is obtained, it can be used for the addition of new observed landmarks to the map. This way, the localization method benefits from the existence of known landmarks but also incorporates detected visual markers with unknown positions.

Similar methods for ground mobile robots were developed in previous work by de la Puente et al., initially based on the observation of 2D point features with a laser scanner [30] and later based on the extraction of planar features from 3D point clouds generated by a tilting laser scanner [31]. One important difference is that in the work presented here an initial map was already a priori known, so this approach is very close to a pure EKF standard localization algorithm. Other important differences are related to the sensors and data used, and to the less stable and less smooth 3D motion of the vehicle. All modules that are upstream of the Localization module benefit from its drift-free estimation of the pose of the drone, $\mathcal{T}_{world\_droneI}$.

3.3.4 Obstacle Detector module

The unknown markers are located on the four center columns and the windows. Their position is unknown but the identification number of the ArUco markers located on these columns is known. This module reports the position of the mentioned environment elements by calculating the mean position of the markers of the obstacle that have been already localized. The markers are located at known position offsets from the center of the obstacle. These offsets are known and used in the calculations. The produced list of obstacles is reported to the Trajectory Planner module.

3.3.5 Supervisor module

This module performs four basic operations. Firstly, it monitors the system during the startup of the agent-level software architecture. Secondly, it coordinates the actions of other modules during flight mode transitions like take-off, landing or the

[11] ArUco: a minimal library for augmented reality applications based on opencv `http://www.uco.es/investiga/grupos/ava/node/26`

beginning of trajectory following operations. Thirdly, it reports the position of the agent to the other drones of the multi-robot system. And fourthly, it gathers the pose measurements from other drones, $\mathcal{T}_{world\_dronek}$, monitors that the other drones keep reporting their positions and passes all their poses as a single message to the Trajectory Planner module. Other drones are considered as obstacles when they are nearby, nearer than a threshold distance. A drone that stops reporting its pose for a long enough period of time, 5s, is considered as off-line by the rest of the multi-robot system. Off-line drones are not reported to the Trajectory Planner.

3.3.6 Mission Scheduler module

The Mission Scheduler utilized to participate in the IMAV 2013 competition was basically sequential. Thus, the mission was divided in a sequence of tasks. In order to add versatility to the mission definition, its tasks are read from a configuration file. The available tasks included all the navigation related goals of IMAV2013: take-off, land, navigate to a desired waypoint while looking in a given direction or towards a column and sleep for a given time. Each task is monitored to determine its completion by this module. The completion event launches the execution of the next task in the sequence.

3.3.7 Yaw Commander module

This module calculates the yaw reference, with respect to the world frame, depending on the agent's current task. This reference sets the direction at which the drone's on-board camera is looking. Thus, it is set to potentially observe the maximum number of fiducial markers. For the IMAV2013, the looking direction was set to look at a particular position occupied by a column, for instance while moving in the column area; or to look in a given direction, for instance while crossing the window.

3.3.8 Trajectory Planner module

This module computes a free collision 2D trajectory (horizontal coordinates x and y) to reach the current mission point, which is set by the Mission Scheduler. The yaw attitude angle is not considered by the planner. Taking into account the other agents' positions a collision-free trajectory for each drone is generated by using a combination of state of the art trajectory planning algorithms: probabilistic road maps[7], a potential field map algorithm[20] and an A-Star algorithm[27].

This paragraph summarizes the calculations performed by the Trajectory Planner. The node map utilized by the planning algorithm is precalculated and unchanged during mission execution. The potential at each node must be recalculated during execution due to the changing position of unknown obstacles and the other agents. Each of the static obstacles, wall sections and columns, and nearby drones, which are nearer than a threshold distance (2.5m), generate potential and, thus, cost for the obstacle-free path calculation algorithm. The resulting planned path is the optimal solution that minimizes the cost to reach the mission point from the currently estimated pose. This path naturally avoids obstacles, and it is only recalculated when a moving obstacle is present in the currently planned trajectory. Other agents that are further away are not considered as obstacles whatsoever. If this module fails to find a feasible path to the mission point, then an empty trajectory is commanded. This event is triggered only when the agent does not have a feasible obstacle-free path available to his current mission point. In our convention, when the trajectory controller receives an empty trajectory, then it starts to hover on the current estimated pose.

The implemented collision avoidance strategy only uses the current position estimate of the other robotics agents. Given that the agents navigate the competition environment at low speeds, to minimize the blurring of the acquired images, this approach is enough to avoid collisions on our multi-robot system. Relevant related work on this characteristic are approaches that consider the relative speed of the obstacles[11, 3, 2] or the incertitude of their localization[15].

3.3.9 Trajectory Controller module

The Trajectory Controller is composed of a state machine and a high-level controller. The state machine's main role is to set the references for the high-level controller while tracking the completion of consecutive checkpoints, switching between straight-line movement and turning modes, and setting a hovering mode at the end of the trajectory. The high-level controller in turn calculates the commands that are sent through WiFi to the low-level controller of the AR Drone. This controller, depicted in Fig. 6, implements four feedback control loops to command the drone in position, speed and yaw. The horizontal linear position loops are approximately decoupled by taking into account the current yaw of the vehicle. The controller was inspired by prior work by Hoffmann *et al.*[16] and Michael *et al.*[25]. In comparison to the work by Hoffmann *et al.*[16], our

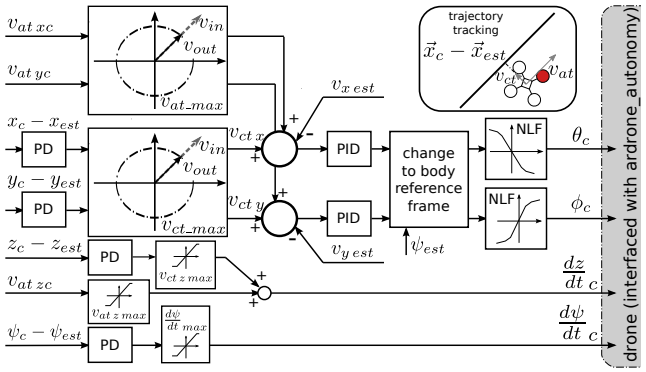

Fig. 6 High-level controller architecture, used in IMAV2012 and IMAV2013. Out is a cascade controller, which consists of an inner speed loop and an outer position loop. The controller includes non-linear laws to model the drone's speed capabilities, and a coordinate transformation to decouple the yaw control from the position control in the horizontal plane. There are many saturation blocks that allow a proper configuration of the controller with regards to maximum velocity limitations on the sending capabilities of the drone.

experiments were carried out in indoor environments and using different sensing solutions: ground optical-flow and LIDAR[29], and, in the current work, ground optical-flow and fiducial markers. Other than the sensing solution, the main difference of our controller when compared to these two prior works is the addition of several saturation laws that can intrinsically take into account speed limitations established by the on-board sensors. In IMAV2012, our drone had to deal with low frame-rate optical-flow measurements[29]. In this work, these saturations were used to configure a desired maximum navigation speed for each agent. These saturations have proved to be the main advantage of this controller as they allow to integrate it in different navigation applications.

As shown in the "trajectory tracking" diagram in Fig. 6, the controller assumes that the navigation is along straight trajectory segments. In this figure, the variables with an *at* subindex stand for "along-track" variables, for instance the x speed command along the trajectory v_{atxc}. Similarly the *ct* subindex stands for "cross-track" referring to commands that are perpendicular to the trajectory. As shown, the controller follows position, yaw, $\{x_c, y_c, z_c, \psi_c\}$; and speed references $\{v_{atxc}, v_{atyc}, v_{atzc}\}$. For this it requires estimates on these same variables, $\{x_{est}, y_{est}, z_{est}, \psi_{est}\}$; and $\{v_{xest}, v_{yest}, v_{zest}\}$. The configuration of the controller consists mainly of PD and PID gains and the saturation values, $\{v_{at\_max}, v_{ct\_max}, v_{atzmax}, v_{ctzmax}, \frac{d\psi}{dt}_{max}\}$. The controller is described in more detail in the article[29] and Master's Thesis[28].

3.4 Multi-Agent-Level Emerging Intelligence

From the design and conventions in the specifications of the Trajectory Planner, see section 3.3.8, and knowing that the drones perform only navigation related missions; the overall expected behavior of the agents during mission execution is:

- Each agent follows a sequence of waypoints given by its mission specification. The mission scheduler executes it sequentially.
- During execution, the trajectory planner will attempt to find trajectories that are collision-free. If it fails, the quadrotor will be controlled to stay in the current position.
- If other robotic agents enter the current trajectory, the trajectory planner will detect it and it will command the trajectory controller, and thus also the quadrotor, to stay in its current position while attempting to find a collision-free path to the current waypoint.
- If another robotic agent is on or enters the current waypoint, then the trajectory planner handles the situation in the same manner. The trajectory controller is commanded to stay in the current position. A new collision-free path will be calculated when the current waypoint is free again.

The interesting characteristic of our multi-robot system is that there is no central mission supervisor. Rather, the synchronization of the drone navigation mission is achieved by the set of rules and conventions specified in the Trajectory Planner and the Trajectory Controller, see sections 3.3.8 and 3.3.9, of which one instance is running on each agent. Arguably, this intelligence does not allow to solve every possible navigation conflict between the robotic agents. However, if the mission is specified correctly the conflicts will not occur often.

The number of drones that can be flown using our architecture is limited by the WiFi bandwidth requirements of the AR Drones. Experimental testing has shown us, that by limiting the quality of the video stream sent by each drone, it is possible to fly up to 6 drones, which share the three available independent WiFi channels in pairs. Otherwise, using drones with an on-board computer running an instance of our agent-level architecture would allow for higher numbers of drones. In this case, the WiFi bandwidth would only be used, during execution, to share the pose estimate of each drone, $\mathcal{T}_{world\_droneI}$, and the information required by the human-supervisor for monitoring purposes. The robotic agents would be able to navigate in a structured area while avoiding collisions with other drones of the proposed

multi-robot system, without the requirement of implementing a complex "Global Mission Planner".

4 Simulation Results

This section shows the final result of a series of simulations that were run to benchmark the designed swarm behavior of the architecture in a simulation of the IMAV 2013 competition environment. Taking advantage of the modularity of our architecture, the simulations are run using most of the actual software architecture. Two simulators were developed for this purpose, see Fig. 2, a quadrotor simulator and a Visual Marker Detector simulator:

- The quadrotor simulator is achieved by means of a state machine that mimics the flying modes of the AR Drone which include: landed, taking-off, hovering, flying, landing and emergency modes. Most of the modes are substituted by simple behaviors. In the flying mode a basic dynamics model explained in[28] is used. This model is not intended to be used for controller tuning or precise quadrotor dynamics simulation. The intent of this quadrotor simulator is to allow the testing of the mission specification and of the logic rules implemented on separate modules that affect module interfaces and module-level internal logic.
- The Visual Marker Detector simulator module implements a simple set of visibility rules, such as the requirement that the visual marker has to be in front of the simulated quadrotor and a range of distances were the visual marker is considered to be detectable by the software. This simulator outputs a noisy measurement of the visible markers pose with respect to the drone, $\mathcal{T}_{droneI\_markerJ}$, whose positions are specified in the map configuration files.

The simulator modules allow the execution of simulations with all the rest of the architecture. An image of the visualization software used to inspect the simulations, or to interpret the behavior of the robotic agents in real flights is shown in Fig. 7. Then, the rest of the modules are the same as during real flights, and thus can be tested and debugged without preparing time-consuming experimental flights. Two examples of these simulations, where five drones fly in a simulated replica of the IMAV 2013 environment are shown in Figs. 8 and 9. In both simulations all the agents were able to accomplish the mission successfully. The launch of the drones was timed and performed at about every 15 seconds. The conflicts during navigation where a drone had to avoid another one are easily perceived,

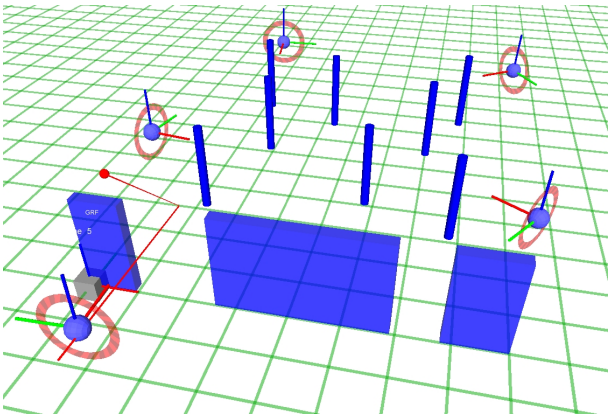

Fig. 7 Visualization in Rviz of a simulated flight performed by five drones in a replica of the IMAV 2013 environment. Drone axes: red x axis, green y axis and blue z axis. The blue cylinders represent the map columns, the blue rectangles represent the wall and its windows, and the green grid represents the floor. The red line shows the current planned trajectory of the selected agent and the red dot is the current waypoint.

because they stand out from the normal execution of the mission.

A second capability of these simulations is that they allow to estimate the limits of our multi-robot solution, except for actual flight dynamics, measurements noise, WiFi bandwidth limitations and other problems related to real flights. In order to determine the maximum number of drones that could simultaneously fly using our solution in the Indoor Autonomy Challenge, a set of simulations in a replica of the IMAV 2013 environment were conducted. Two of such simulations with five drones are shown in Figs. 8 and 9. In order to interpret these figures the modules specifications, which are described in the Agent-Level Modular Architecture section (Sec. 3), have to be taken into account. The expected behavior of the agents, discussed in section 3.4, is mainly determined by the mission definition, provided by the Mission Scheduler's configuration file, and by the design of the Trajectory Planner module, described in section 3.3.8.

In Figs. 8 and 9, the simulated drones are ordered by their launch times and their executed trajectories are shown with lines plots. The mission specification used to run both simulations is the same that was used during the IMAV 2013 competition, which is the following sequence of tasks:

1. take-off and start the whole architecture,
2. move in front of the window and then move to one corner of the poles area (thus, crossing the big window). During this task, the quadrotor is controlled to look at the poles area perpendicularly to the wall, which results in a constant yaw reference,
3. perform laps around the poles area for 5 minutes,

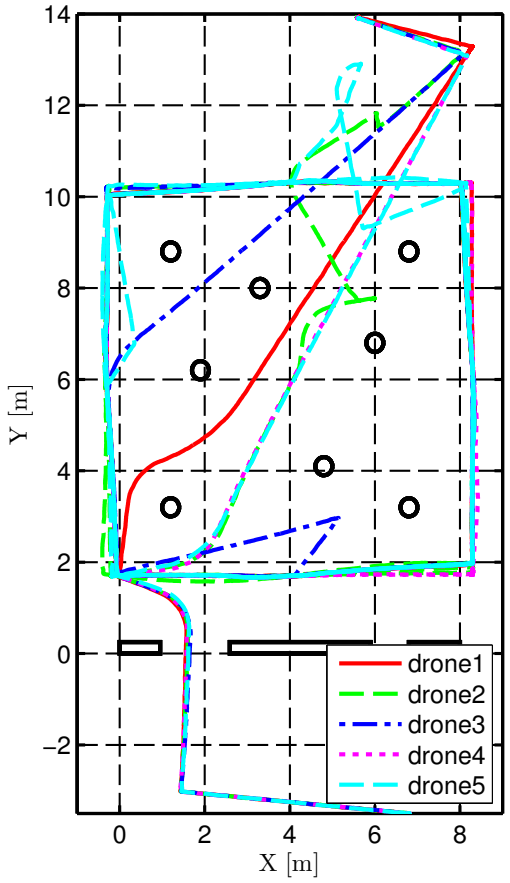

Fig. 8 Simulated flight where five drones flew simultaneously to perform navigation tasks in a replica of the IMAV 2013 environment. The poles are plotted as black circles, and the wall is plotted by three black rectangles, with the two free collision passages represented as windows. The size of the AR Drone 2.0 is too big for the small window, so all the drones have to cross the big window. The trajectory executed by each agent is shown as a distinct line, whose color and line style are specified in the figure's legend. In this simulation, all the agents managed to accomplish their respective missions. The conflicts during navigation when a drone had to avoid another one are easily perceived, because they stand out from the normal execution of the mission. Only two such conflicts occurred during this simulation: the first one resulted in drone3 taking a detour during the laps execution when another drone was traversing the window; and the second one occurred while drone2 was finishing the crossing of the unknown poles area.

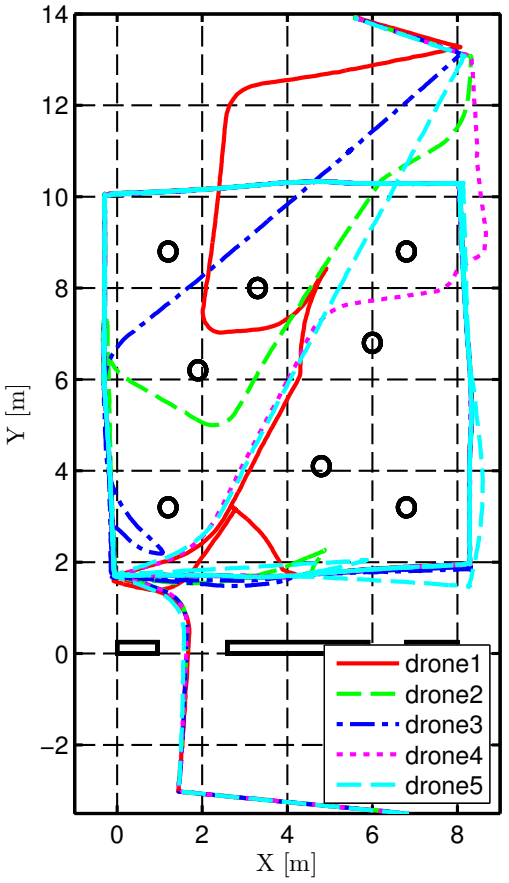

Fig. 9 Simulated flight where five drones flew simultaneously to perform navigation tasks in a replica of the IMAV 2013 environment. The interpretation of this is the same as for Fig. 8. Again, in this second simulation, all the robotic agents managed to accomplish their respective missions. The main difference between both simulations is the higher number of navigation conflicts that occurred during the mission execution.

4. cross the poles area moving towards the upper right corner of the map,
5. navigate to the final scoring location and perform the landing, hovering and landing task specified in the IMAV 2013 rules. During this task, the quadrotor is controlled to look at the poles area perpendicularly to the wall, which results in a constant yaw reference.

The laps and the crossing through the unknown poles area tasks are specified so that all the drones will probably navigate in the same direction. This mission specification requirement minimizes the occurrence of navigation conflicts. During tasks 3 and 4, the quadrotor is controlled to look at a specific corner column. The laps around the poles area are specified by waypoints in the corners and in the middle points of the square lap. The drone switches to looking at the next column in these middle points.

Considering the size of the map, 10x15m, and the size of the AR Drones, around 52x52cm, the competition environment gets relatively crowded when more than 5 drones are navigating around the columns' area. Taking this into account, the results of the simulations for two to seven drones, with the above described mission specification, are the following:

– 2-4 drones: the robotic agents will automatically synchronize during mission execution solving the conflicts when they occur.

– 5-6 drones: the robotic agents do sometimes enter a conflictive situation, because of a lack of initial synchronization. As time goes on, the drones become automatically synchronized and they are able to navigate the laps in a continuous manner. Two of these simulations are shown in Figs. 8 and 9. For six drones, the synchronization time increases due to a higher number of navigation conflicts; also the crossing through the middle area of the map caused many more navigation conflicts than in the case of a five drones simulation.
– 7 drones: once the multi-robot system becomes synchronized to perform the laps, the navigation is not continuous. During lap execution, there are only two agents actually moving at any given time; while the other five agents are staying at their current positions waiting for their next waypoint to be free.

The main factor, more important than localization and control inaccuracies, that affects the selected safety distance between quadrotors is the necessity to avoid interference among the ultrasound altimeters of the AR Drones. In simulation, the safety distance is the main cause limiting the maximum number of drones that may fly in the IMAV 2013 challenge using our solution. In real flights, WiFi bandwidth limitations may also limit the maximum number of drones to around six, in our experience maximally nine, drones flying simultaneously.

From the previous analysis of the simulations, it was concluded that the optimum number of drones for this mission is five. With this group size, the robotic agents tend to execute their individual missions almost with no interruptions, leading to a maximum score for our solution in the IMAV 2013 Indoor Challenge environment.

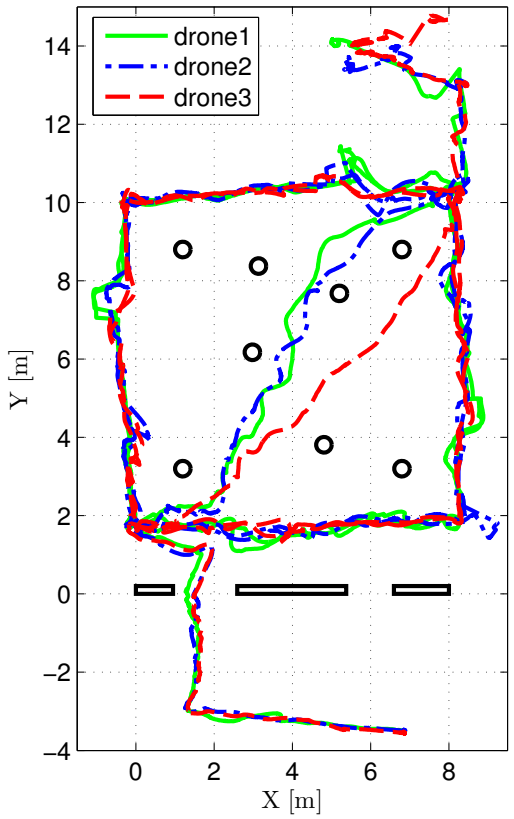

Fig. 10 Experimental flight, three drones fly simultaneously performing navigation tasks. The environment is a replica of the IMAV 2013 Indoor Challenge environment. Figs. 10, 11, 13, 14 and 15 correspond to this experimental flight. The inner unknown pole locations are the estimates of drone1. The flight through this area is usually successful even when the position of the unknown poles is not precisely known, because the agents localize themselves with respect to these poles when they see them, thus planning obstacle-free trajectories with respect to their position estimate. The execution of this flight was successful, with all agents accomplishing their missions. A video of this test is available at the website: `http://www.vision4uav.com/?q=node/386` .

5 Experimental Results

In this section one experimental test with three drones is described. This test shows the performance of our architecture in a replica of the IMAV 2013 competition environment. The mission specification is identical, except for a shorter time of 3 minutes performing laps outside the pole area, to the mission specification described in the Simulation Results section (see the enumerated sequence of tasks written in Sec. 4). Subfig. 1(c) shows the drones attempting to cross the unknown pole area during this flight. And the figures 10 and 11 summarize the experimental flight. In these figures, the obstacles, the wall with two windows and the eight poles, are shown in black. The trajectories executed by the drones, as estimated by the localization module, are shown with lines. The agents know the position of the four corner poles and the wall, and the size of the windows; but need to estimate the position of the four unknown poles and the windows.

A 2D overview plot of the flight is shown in Fig. 10. As shown in the videos the quadrotors are commanded to look at specific known poles to control the odometry position error. At the middle of each side the agents are commanded to look at the next corner pole. At this moment the position estimate might change. These events increase the localization error in specific points of the mission execution and trigger the planning of new trajectories. In this flight they occur repeatedly in the points $[-0.5, 7.75]$ and $[5.5, 10.5]$, see Fig. 10.

The explained emergent synchronized behavior of the drones is shown in Fig. 11. The mission is specified

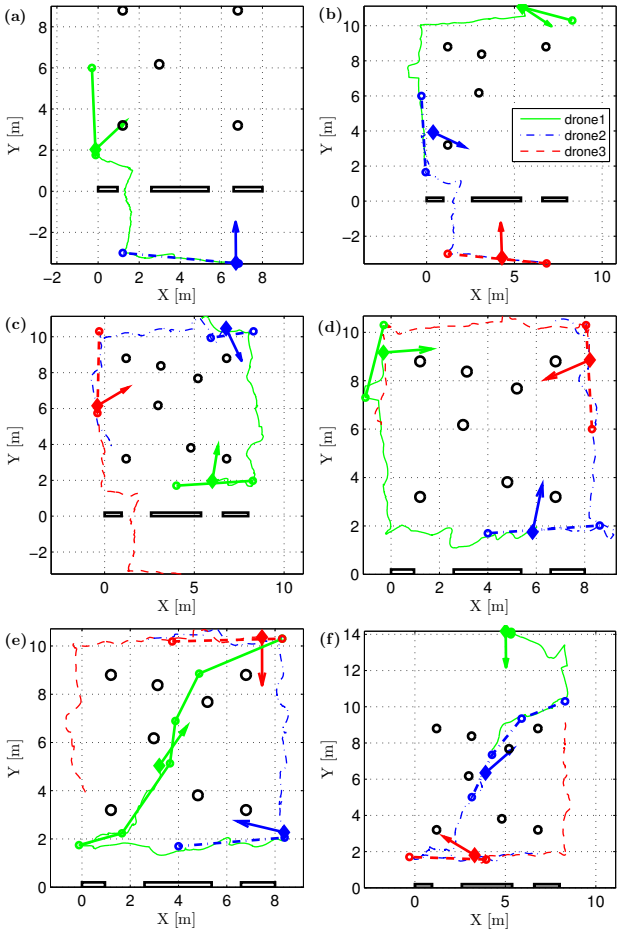

Fig. 11 Same experimental flight as shown in Fig. 10. These figures, labeled a-f, are ordered in time, and each agent's plots have a different color and line style. The drones ordered by their launch times are plotted with: (*drone1*) green-solid lines, (*drone2*) blue-dash dotted lines and (*drone3*) red-dashed lines. The planned trajectory is shown with a thicker width than the actual executed trajectory. The dotted lines are the current trajectory references at the end of the plotted time period. The arrows indicate the direction at which the quadrotor is estimated to be looking in the plotted position. As shown, the quadrotors are commanded to look at one of the four known corner columns of the map. The unknown poles localized by *drone1* are shown in these figures. (a), (b), (c) and (d) show consecutive flight intervals of 30 seconds showing the startup sequence of the multi-robot system. In this period, *drone1* crosses the window and performs 1.25 laps. (e) and (f) show consecutive flight intervals of 30 second of the final stages of the mission. The drones cross the unknown poles area by planning and executing obstacle-free trajectories. A video of this flight can be found in the website `http://www.vision4uav.com/?q=node/386` .

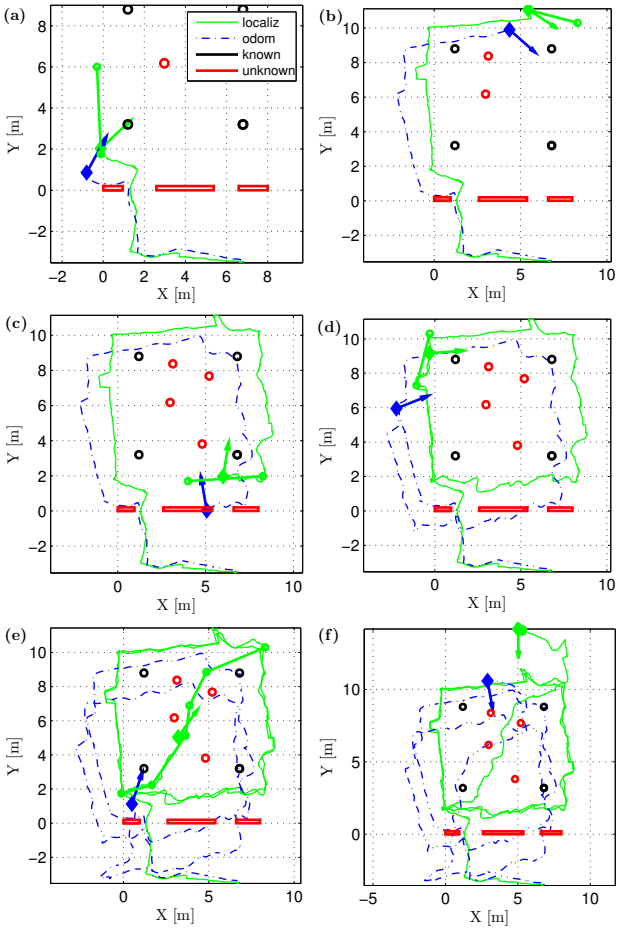

Fig. 12 Same experimental flight as shown in Fig. 10. These figures, labeled a-f, are ordered in time, all the plotted elements are from *drone1*. The timestamps at which these plots were generated are the same as in Fig. 11(a-f). The Localization module drift-free estimate from *drone1* is shown with thin green-solid lines, the odometry-based estimate from the Pose Estimator module is shown with blue-dash dotted lines, the known environment elements are shown with thick black lines and the unknown environment elements are shown with thick red lines. The drone estimated positions are shown with diamonds and the yaw estimates with an arrow of the same color. The thick green line plot, shows the currently commanded trajectory. These figures show how the Localization module is able to calculate a drift-free pose from the odometry-based estimate. The corrections are specially visible after the yaw turns, when the drone can localize with respect to the next corner column. It is noted that in this experiment some known markers were placed on the wall in the taking-off side, but that the windows were localized as unknown elements.

to minimize navigation conflicts and maximize the final achieved competition score. For example, as described in the Simulation Results section (Sec. 4), all the agents perform laps in the same direction.

In Fig. 11 the current estimated position and orientation of each quadrotor is shown with a marker and an arrow, the last 30 seconds of the executed trajectory are shown by a line and the currently planned trajectory is shown by a segmented line with markers. Subfigs. 11(a)-(d) show the startup sequence, where the drones cross the big window and the automatic synchronization is achieved. Subfigs. 11(e)-(f) show the final steps of the mission execution, where the drones cross the unknown pole area and land in the final position. More details of

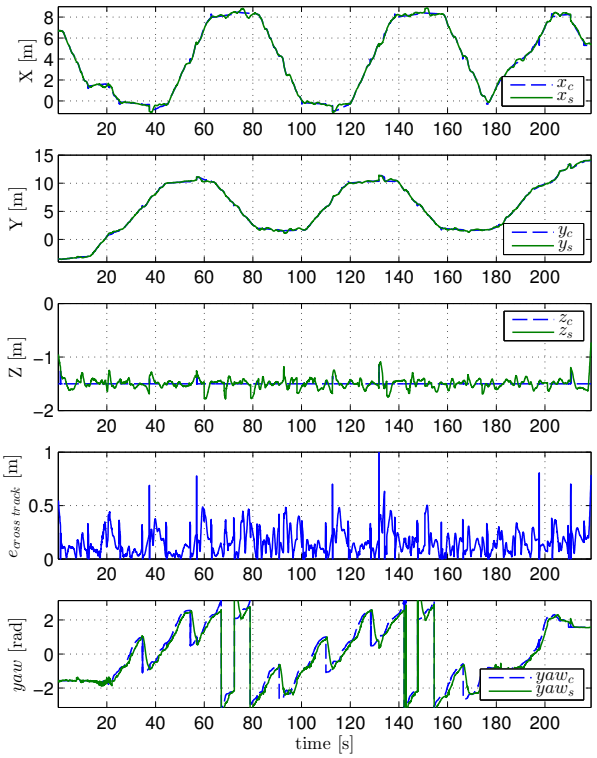

Fig. 13 Flight performance of *drone1*, same experimental flight as shown in Figs. 10 and 11. The plots show the estimated position and yaw of the quadrotor and the inputs given to the position and speed control loops inside the trajectory controller module. The $e_{cross\,track}$ plot shows the absolute position control error based on these commands and the estimated position, as calculated by the localization module. The yaw command only changes sharply when the drone is commanded to look at a different column. The rest of the time when a sharp change is shown it has just evolved to the opposite side of the $[-\pi, +\pi]$ interval.

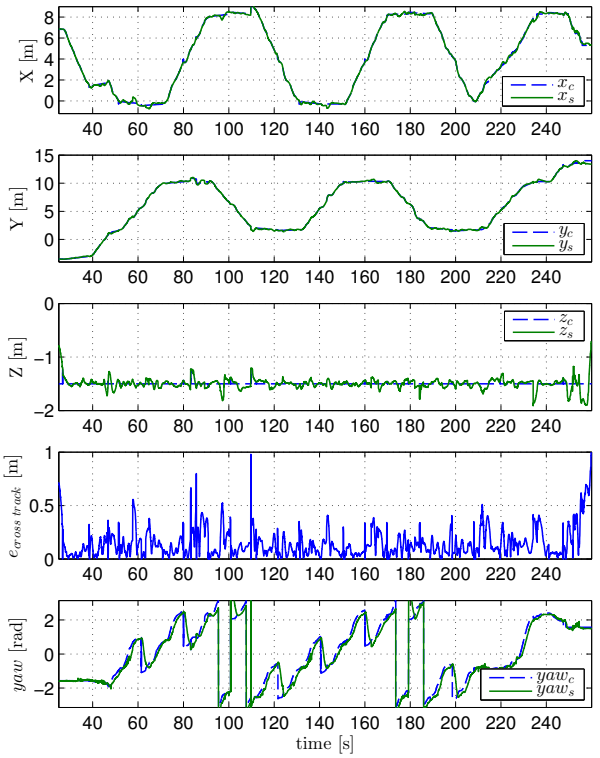

Fig. 14 Flight performance of *drone2*, same experimental flight as shown in Figs. 10 and 11. The interpretation of this figure is similar to that of Fig. 13.

the execution of this mission can be read in the caption of the experiment figures 10 and 11.

Additionally, Fig. 12 shows, for *drone1* only, plots of the drift-free (green) and the odometry-based (blue) estimates. The known environment elements are shown in black and the progressively mapped unknown elements are shown in red. This figure shows how the Localization module is able to calculate a drift-free pose from the odometry-based estimate.

The performance in terms of trajectory following capabilities, and tracking error of each drone are shown in Figs. 13, 14 and 15. In each of the figures the control references and the estimated positions, output of the localization module, are shown. There is no ground truth available, so these estimates are utilized to measure the performance of the system. $\{X, Y, Z, yaw\}$ are expressed in the same reference system as Fig. 10. $e_{cross\,track}$ is the trajectory following position error, including horizontal and

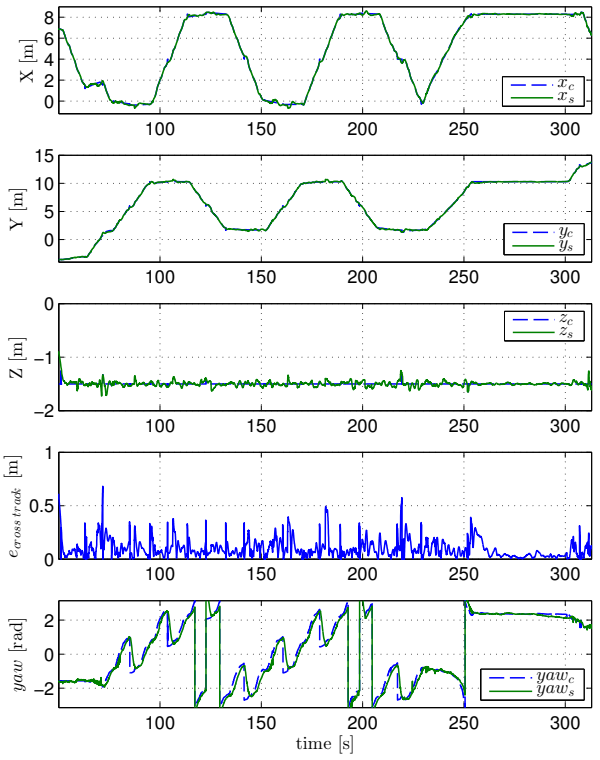

Fig. 15 Flight performance of *drone3*, same experimental flight as shown in Figs. 10 and 11. The interpretation of this figure is similar to that of Fig. 13.

	drone1	drone2	drone3
e_{ct} [m]	0.1971	0.1986	0.1290
$horiz_{ct}$ [m]	0.1723	0.1678	0.1166
$vert_{ct}$ [m]	0.0957	0.1063	0.0551
yaw_{error} [deg]	26.97	26.40	24.37

Table 1 Flight performance of the drones in the experimental flight shown in Figs. 10 and 11. The values are the RMS error over the whole mission execution shown in Figs. 13, 14 and 15. e_{ct}, $horiz_{ct}$, $vert_{ct}$ are the total, horizontal and vertical position errors with respect to the trajectory reference respectively. The yaw_{error} is the error with respect to the commanded yaw, which corresponds to looking to one of the map columns or in an specific direction (which happens only in the first and last tasks).

vertical error. In these figures it can be appreciated that the tracking error is less than 0.5 m most of the time and that yaw is commanded slowly to limit the blur in the acquired video stream, so as to improve the performance of the marker detector module.

The overall performance of the navigation tasks is summarized by the RMS error of $\{X, Y, Z, yaw\}$ during trajectory following. These values are shown in Table 1. The RMS horizontal tracking error is around 10-15cm, which is between a fourth and a third of the size of the drone. The RMS horizontal tracking error is around 10-15cm, which is between a fourth and a third of the size of the drone. The RMS yaw error is between 2deg and 25deg depending on the current mission task that the agent is performing. The lowest value corresponds to a constant yaw reference, during the first task before crossing the window, and the highest values correspond to the maximum yaw reference variations, during the navigation task around the columns. The reason for this surprisingly high RMS yaw error is that its controller is tuned with a very low proportional gain, so that the blurring on the acquired images is minimized. This was required to accommodate the image acquisition for the lowest light conditions that could occur in the indoor environment of the competition.

6 Conclusions

An overview of an autonomous quadrotor multi-robot architecture designed to participate in the indoor challenge of the IMAV 2013 competition has been presented. All the robotic agents have access to the global position of all the agents of the team. Our multi-robot solution consists of low-cost AR Drone 2.0 platforms and their corresponding ground computers and WiFi links which communicate using the ROS middleware. Its deployment is quite fast thanks to the fact that only a limited number of visual markers must be placed.

The presented work is a continuation of[30]. On top of this prior work, firstly, our solution to IMAV 2013 has been analyzed in simulation to determine the architecture capabilities and the optimal number of quadrotors that could be flown in the competition environment based on our solution. And secondly, the experimental performance of the presented architecture has been analyzed to determine how precise the MAVs can navigate using this version of our localization and controller modules.

The first contribution of this paper is the presentation of our vision-based quadrotor multi-robot solution for the IMAV 2013 Indoor Autonomy Challenge, which was awarded with the First Prize in this challenge. Our architecture was robust enough to work properly during the competition, but encountered problems due to its dependency on WiFi communication links. The second contribution is an analysis of the optimal performance of our multi-robot system to achieve the navigation tasks of the IMAV2013 competition. The third contribution is the successful experimental testing of a localization strategy using different kinds of data. An EKF based algorithm formulation[31, 32] was adapted to work with multicopters instead of ground robots. The last contribution is that the presented architecture has been made publicly available as open-source[12]. To the authors knowledge this is the only indoor visual-based quadrotor multi-robot solution whose code is publicly available. The authors hope that this open-source software will benefit other developers in their research.

7 Future Work

The first work to be conducted by the authors is to accommodate the developed architectures to bigger MAVs that can carry an on-board computer, thus continuing our research group's, the CVG-UPM's, efforts towards the research of MAVs and Computer Vision for civilian applications.

The main drawbacks of the presented system are the off-board computation, and the dependency on visual markers. The work by other groups present tested approaches to perform autonomous navigation in indoor and outdoor unstructured environments using only on-board computation in low-end computers[42, 41, 34, 38] and higher-end ones[13, 39].

In order to move in that direction higher-end quadrotors with an on-board computer should be used. Many modules of the system can be improved taking into account this literature to obtain pose estimates at a high frequency based on the IMU data and precise

[12] `http://www.vision4uav.com/?q=quadrotor_stack`

timestamping of all sensor data and the use of faster and more precise controllers. The trajectory planning can be improved to perform 3D trajectories and also highly aggresive maneuvers as developed by[34, 38].

On the side of the multi-robot system development, it would be beneficial to improve the system to use only the visual markers with no previously known positions and also a visual quadrotor detection system. This would be an intermediate step to move to a full visual SLAM based system, where the MAVs would need to figure out whether they are looking at the same scene as the others and where they could avoid collisions and collaborate. Additionally, other multi-robot advanced behaviors or active missions, such as collaborative 3D reconstruction, could be tested.

References

1. Abeywardena D, Kodagoda S, Dissanayake G, Munasinghe R (2013) Improved state estimation in quadrotor mavs: A novel drift-free velocity estimator
2. van den Berg J, Lin MC, Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation 2008 (ICRA 2008), IEEE, pp 1928–1935
3. Berg J, Guy S, Lin M, Manocha D (2011) Reciprocal n-body collision avoidance. In: Pradalier C, Siegwart R, Hirzinger G (eds) Robotics Research, Springer Tracts in Advanced Robotics, vol 70, Springer Berlin Heidelberg, pp 3–19, DOI 10.1007/978-3-642-19457-3_1, URL `http://dx.doi.org/10.1007/978-3-642-19457-3_1`
4. Brescianini D, Hehn M, D'Andrea R (2013) Quadrocopter pole acrobatics. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, pp 3472–3479
5. Bristeau PJ, Callou F, Vissière D, Petit N, et al (2011) The navigation and control technology inside the ar. drone micro uav. In: 18th IFAC world congress, 1, pp 1477–1484
6. Conte G (2009) Vision-Based Localization and Guidance for Unmanned Aerial Vehicles. PhD thesis, Linkopings universitet
7. D Hsu RM JC Latombe (1997) Path planning in expansive configuration spaces. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 27192726
8. Engel J, Sturm J, Cremers D (2012) Camera-based navigation of a low-cost quadrocopter. In: Proc. of the International Conference on Intelligent Robot Systems (IROS)
9. Faigl J, Krajník T, Chudoba J, Saska M, Přeučil L (2013) Low-Cost Embedded System for Relative Localization in Robotic Swarms. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA), IEEE
10. Fernandez-Madrigal JA, Blanco-Claraco JL (2012) Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods, chapter 10: Advanced SLAM Techniques. IGI Global
11. Fiorini P, Shillert Z (1998) Motion planning in dynamic environments using velocity obstacles. International Journal of Robotics Research 17:760–772
12. Forster C, Lynen S, Kneip L, Scaramuzza D (2013) Collaborative monocular slam with multiple micro aerial vehicles. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, IEEE, pp 3962–3970
13. Fraundorfer F, Heng L, Honegger D, Lee GH, Meier L, Tanskanen P, Pollefeys M (2012) Vision-based autonomous mapping and exploration using a quadrotor mav. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, pp 4557–4564
14. Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas FJ, Marín-Jiménez MJ (2014) Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition 47(6):2280–2292
15. Hennes D, Claes D, Meeussen W, Tuyls K (2012) Multi-robot collision avoidance with localization uncertainty. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, pp 147–154
16. Hoffmann GM, Waslander SL, Tomlin CJ (2008) Quadrotor Helicopter Trajectory Tracking Control. Electrical Engineering (2008) pp 1–14
17. Jayatilleke L, Zhang N (2013) Landmark-based localization for unmanned aerial vehicles. In: IEEE International Systems Conference (SysCon'13), pp 448–451
18. Krajník T, Nitsche M, Faigl J, Vaněk P, Saska M, Přeučil L, Duckett T, Mejail M (2014) A practical multirobot localization system. Journal of Intelligent & Robotic Systems 76(3-4):539–562, DOI 10.1007/s10846-014-0041-x
19. Kushleyev A, Mellinger D, Powers C, Kumar V (2013) Towards a swarm of agile micro quadrotors. Autonomous Robots 35(4):287–300
20. Latombe JC (1991) Robot Motion Planning. Kluwer Academic

21. Leishman RC, Macdonald JC, Beard RW, McLain TW (2014) Quadrotors and accelerometers: State estimation with an improved dynamic model. Control Systems, IEEE 34(1):28–41
22. Lindsey Q, Mellinger D, Kumar V (2012) Construction with quadrotor teams. Autonomous Robots 33(3):323–336
23. Mao G, Drake S, Anderson BDO (2007) Design of an Extended Kalman Filter for UAV Localization. In: Information, Decision and Control, 2007 (IDC'07), pp 224–229
24. Meier L, Tanskanen P, Heng L, Lee GH, Fraundorfer F, Pollefeys M (2012) Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision. Autonomous Robots 33(1-2):21–39
25. Michael BYN, Mellinger D, Lindsey Q (2010) The GRASP Multiple Micro UAV Testbed. IEEE Robotics & Automation Magazine (September):56–65
26. Mostegel C, Wendel A, Bischof H (2014) Active monocular localization: Towards autonomous monocular exploration for multirotor mavs. In: Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)
27. P E Hart BR N J Nilsson (1968) A formal basus for the heuristic determination of minimum cost paths. IEEE Transactions on SYstems Science and Cybernetics 4(2):100–107
28. Pestana J (2012) On-board control algorithms for Quadrotors and indoors navigation. Master's thesis, Universidad Politécnica de Madrid, Spain
29. Pestana J, Mellado-Bataller I, Sanchez-Lopez JL, Fu C, Mondragón IF, Campoy P (2014) A general purpose configurable controller for indoors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles. Journal of Intelligent & Robotic Systems 73(1-4):387–400
30. Pestana J, Sanchez-Lopez JL, de la Puente P, Carrio A, Campoy P (2014) A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, IEEE, pp 617–622
31. de la Puente P, Rodriguez-Losada D, Pedraza L, Matia F (2008) Robot goes back home despite all the people. In: Proc. 5th. Conference on Informatics in Control, Automation and Robotics ICINCO 2008 Funchal, Portugal, pp 208–213
32. de la Puente P, Rodriguez-Losada D, Valero A (2009) 3D Mapping: testing algorithms and discovering new ideas with USARSim. In: USARSim workshop, IEEE Int. Conf. on Intelligent Robots and Systems (IROS)
33. Renzaglia A, Doitsidis L, Martinelli A, Kosmatopoulos EB (2012) Multi-robot three dimensional coverage of unknown areas. The International Journal of Robotics Research p 0278364912439332
34. Richter C, Bry A, Roy N (2013) Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In: Proceedings of the International Symposium on Robotics Research (ISRR)
35. Ritz R, Muller M, Hehn M, D'Andrea R (2012) Cooperative quadrocopter ball throwing and catching. In: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, pp 4972–4978
36. Rudol P (2011) Increasing autonomy of unmanned aircraft systems through the use of imaging sensors. Master's thesis, Linkoping Institute of Technology
37. Sanchez-Lopez JL, Pestana J, de la Puente P, Carrio A, Campoy P (2015, under review) A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation. Journal of Intelligent & Robotic Systems
38. Shen S, Mulgaonkar Y, Michael N, Kumar V (2013) Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In: Robotics: Science and Systems, Citeseer
39. Shen S, Mulgaonkar Y, Michael N, Kumar V (2013) Vision-based state estimation for autonomous rotorcraft mavs in complex environments. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, pp 1758–1764
40. Vásárhelyi G, Virágh C, Somorjai G, Tarcai N, Szörényi T, Nepusz T, Vicsek T (2014) Outdoor flocking and formation flight with autonomous aerial robots. Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on
41. Weiss S, Achtelik MW, Chli M, Siegwart R (2012) Versatile distributed pose estimation and sensor self-calibration for an autonomous mav. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, pp 31–38
42. Weiss S, Achtelik MW, Lynen S, Chli M, Siegwart R (2012) Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, pp 957–964