A Vision-based Quadrotor Swarm for the participation in the 2013 International Micro Air Vehicle Competition

Jesús Pestana[1] and Jose Luis Sanchez-Lopez[1] and Paloma de la Puente[2]
and Adrian Carrio [1] and Pascual Campoy[1]

***Abstract*— This paper presents a completely autonomous solution to participate in the 2013 International Micro Air Vehicle Indoor Flight Competition (IMAV2013). Our proposal is a modular multi-robot swarm architecture, based on the Robot Operating System (ROS) software framework, where the only information shared among swarm agents is each robot's position. Each swarm agent consists of an AR Drone 2.0 quadrotor connected to a laptop which runs the software architecture. In order to present a completely visual-based solution the localization problem is simplified by the usage of ArUco visual markers. These visual markers are used to sense and map obstacles and to improve the pose estimation based on the IMU and optical data flow by means of an Extended Kalman Filter localization and mapping method. Taking into account the other swarm agents' positions a free-collision trajectory for each drone is generated by using a combination of state of the art trajectory planning algorithms: probabilistic road maps, a potential field map algorithm and an A-Star algorithm. The last element of our autonomous agent is a robust mid-level controller which executes the generated trajectory commands. This paper also presents a discussion of the performance of our architecture on various simulated and experimental flights on a replica of the IMAV2013 environment. The presented solution and the performance of the CVG_UPM team were awarded with the First Prize in the Indoors Autonomy Challenge of the IMAV2013 competition.**

I. Introduction

The motivation of this work is the design of a solution to participate in the 2013 edition of the International Micro Air Vehicle Flight Competition (IMAV2013). The IMAV Flight Competition is the most relevant European competition in the fields of Autonomous Aerial Robotics and Small Remotely Piloted Air Systems (sRPAS). Our research group, the Computer Vision Group (CVG), was awarded[3] for its performance in the 2012 edition of the IMAV competition[21] showing the potential of our group in the development of autonomous Unmanned Aerial Systems (UAS). The learning experience obtained from the indoor dynamics competition encouraged us to keep working in the same direction and also to try a swarming approach in the 2013 edition. Our motivation for participating in such competitions is to develop autonomous systems which can be later modified to perform civilian applications. The 2013 edition's rules are significantly different with respect to former edition's. In IMAV2013 there was only one indoor competition (see [3]) which requires a high level of autonomy. The scenario has some fixed and previously known obstacles (a wall and four fixed poles) and several obstacles located at unknown positions (two windows and four pole obstacles). The indoor competition includes various challenges, including flying through a window, flying through an obstacle zone, target detection and recognition, path following and precision landing, among others.

The second motivation for this paper is that there is a large variety of applications which require a robotic system to densely navigate within a wide area. Such applications can benefit from a swarming approach for the required data gathering of the problem at task, taking benefit from a multi-robot system. For instance, such an approach could be applied to security and surveillance tasks of middle sized areas.

After a deep analysis of the contest characteristics, a

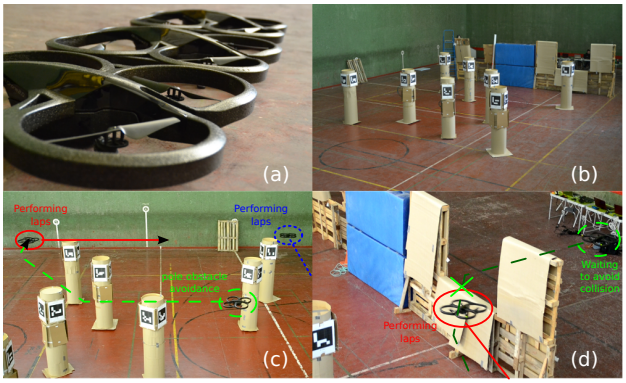

Fig. 1. The presented solution has been designed using AR Drone 2.0 quadrotors, see (a), in a replica of the IMAV2013 indoors challenge environment where the map and the obstacles were marked using ArUco markers [2], see (b). This environment, used during experimental flights, consists of a small window, a big window and 8 poles. The position of the wall is previously known except for the positions of the windows along it are unknown. The positions of the 4 corner poles are previously known, however, the positions of the 4 poles in the middle are unknown. The windows and poles represent obstacles that must be avoided during flight. The subfigure (c) shows a experimental flight where one of the drones is crossing the unknown poles area. The unknown poles are robustly located on previous laps, where the drone performs laps around the known poles, ensuring a good estimation of their positions. Our framework also allows to test partner collision avoidance during experimental flights, as shown in (d), where a drone is waiting until the path to cross the big window is clear. The flights shown in (c) and (d) are both explained in detail on Sec. V. Videos and more information about our experimental flights can be found in the website `http://www.vision4uav.com/?q=node/386` .

*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.

[1]Computer Vision Group, Centre for Automation and Robotics, CSIC-UPM. {jesus.pestana, jl.sanchez, adrian.carrio, pascual.campoy}@upm.es. www.vision4uav.com

[2]Institut of Automation and Control, Vienna University of Technology. pdelapuente@acin.tuwien.ac.at

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

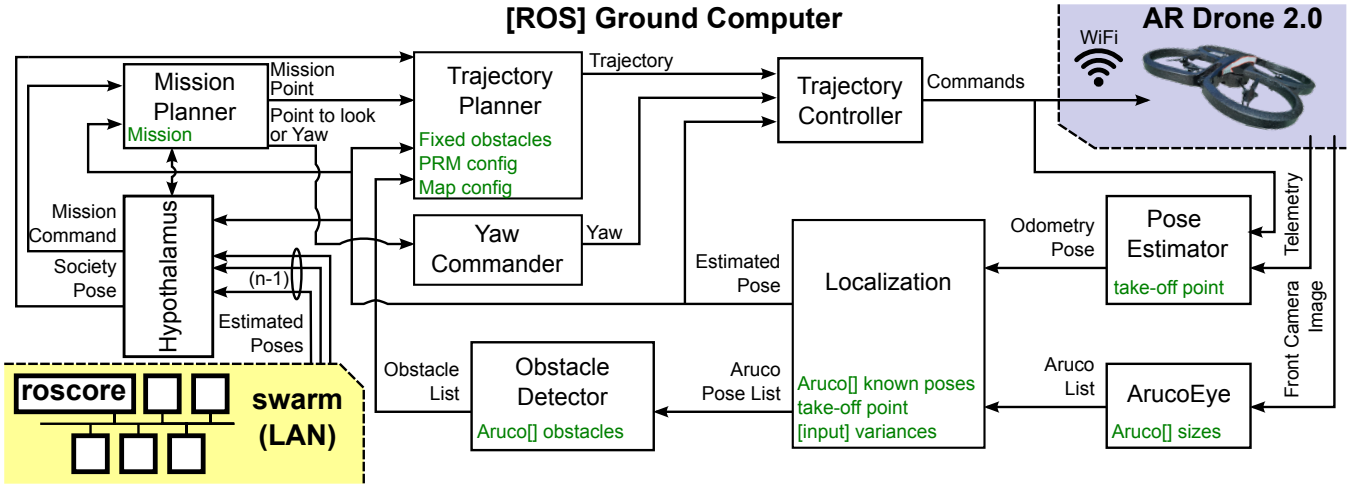

Fig. 2. Robotic swarm agent software architecture. The architecture is implemented using the Robot Operating System (ROS) framework. The architecture is modular, where the modules or ROS nodes of each agent are executed on a ground station which commands its corresponding AR Drone via WiFi. Each white box represents a module, its configuration parameters are written in green. The localization module is implemented using an EKF which fuses the odometry based estimation with the visual markers feedback. This module broadcasts the estimated pose to the mission and trajectory planning modules, to the controller module and to the other robotic agents. The hypothalamus module receives the estimated position of the other robots and communicates it to the trajectory planner.

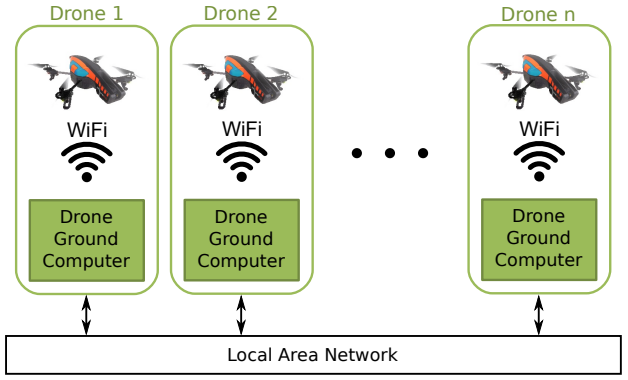

Fig. 3. The swarm is composed by identical robotic agents, which consist of an AR Drone 2.0 and an instance of the software architecture. The drone is commanded via WiFi from a ground station. The ground stations can communicate with each other through a Local Area Network (LAN). The communications between modules and swarm agents is implemented creating a single ROS network.

vision-based Quadrotor Swarm was selected as the best option to join the IMAV2013 indoor flight competition. A swarm composed by a significant number, 3 or more drones, of relatively simple quadrotors is used to achieve all navigation missions. Additionally, as we decided to work with a vision-based swarm, external visual markers are used to simplify the localization problem. Our swarm is fully autonomous, and thus, the level of autonomy is categorized as "Autonomous Mission Control", requiring a small number of operators to start and monitor the whole system. In case something goes wrong, an operator can stop independent agents of the swarm to prevent further malfunctioning.

The layout of the paper is the following. First, the modules of our architecture are described in sections II & III. Second, a discussion about the optimum number of drones that can be flown simultaneously in the IMAV2013 competition environment is presented in section IV. Third, two experimental flights in a replica of the competition map are presented in section V. And lastly, the future work and the conclusions are discussed respectively in the sections VI & VII.

II. System Description

The system is composed by a swarm of autonomous Unmanned Aerial Vehicles (UAVs), which in the case of our proposal for IMAV2013 are AR Drone 2.0s, see Fig. 1(a). Each drone is autonomous and can complete a previously defined navigation mission avoiding obstacles and collisions with the other drones of the swarm. There is no high-level intelligence that controls or synchronizes the drones. Therefore, the system has a "swarming" or cooperative behavior. All the drones in the swarm share their pose with the rest, so that the partner detection problem is simplified and solved.

Each robotic agent is composed of the quadrotor platform and a ground station, see Fig. 3. The agent runs several software modules, depicted in Fig. 2, which communicate using the Robot Operating System (ROS) software framework, see [13]. The characteristics of the AR Drone 2.0 are thoroughly explained in [5]. The ground computer communicates with the drone via Wi-Fi using the ardrone_autonomy ROS package [1]. On the other side, all drones' ground computers are connected through a Local Area Network (LAN) and communicate using ROS.

III. Architecture Software Modules

In this section the main modules of each robotic agent in the swarm are described. The modules are depicted in figure 2. The "Pose Estimator" and the "Trajectory Controller" modules are explained in the following article and Master's Thesis [21], [20]. The controller is able to follow yaw commands while executing a trajectory. The "yaw commander"

module, see Fig. 2, calculates de yaw reference depending on the current swarm agent's mission. This module could potentially decide in which direction to look in order to explore or to get the most localization information from the environment. These possibilities will be explored in future work. Previous works of the authors in this architecture are described in [23].

A. Drone's ArUco Eye

The localization of the drone in the map is firstly estimated using the Pose Estimator module. Since this measure has drift it has to be corrected with absolute measurements. For this purpose, we use external ArUco visual markers, which are shown in subfigures 1b & 1c. These visual markers are open-source and the software library can be downloaded from [2]. This library computes the current 3D pose of the camera with respect to the ArUco markers that are visible in the current frame. The visual markers help us solve two problems at the same time in quite a straight forward manner: the problem of sensing the various obstacles, which is a very hard task using only computer vision techniques, and we also avoid the visual localization problem in a general environment, which has not been entirely solved yet. A remarkable work where visual localization is achieved using AR Drones is described in [11].

B. Drone's Localization and Mapping

Localization in indoor environments is a challenging task for UAVs, especially if a low cost and very lightweight solution is required [18], [22], [15], [12]. In the absence of GPS and laser sensors, visual approaches are very popular [22], [15], [12].

In the proposed system, the global localization of each drone is based on IMU and optical flow data for the pose estimation, calculated by the Pose Estimator module. However, this measure has some drift which may be significant, so it should be corrected with more reliable information from the environment when available. ArUco visual markers (see section III-A) are used for this purpose. This library provides the 3D pose of the camera with respect to each ArUco visual marker in a simple and convenient manner. The input of the localization node are hence the pose estimation result (similar to odometry) and the relative observations of the ArUco markers, received in terms of ROS messages.

Since the environment can be partially known in advance, some fixed landmarks are employed. ArUco landmarks with previously known global poses are attached to the known poles. Other markers are placed on the wall and the unknown obstacles. Simple and easy-to-use accessories for a fast arrangement of the ArUco markers in this environment were designed and built.

Localization with visual external aids for UAVs has been recently proposed in other works [22], [15], [12]. The method presented by Jayatilleke and Zhang [15] 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.* presents an interesting approach for local relative localization in swarms of micro UAVs, that requires to keep external markers always visible. Our method was mainly inspired by the work by Rudol [22], but our models and formulation are quite different from those proposed by Conte [6].

We designed and implemented an Extended Kalman Filter (EKF) that allows the complete 6 DOF pose of the drone to be corrected by integrating the odometry data and the information from the visual external markers detection. The localization method benefits from the existence of known landmarks, but it also incorporates unknown detected features, using a Maximum Incremental Probability approach for building a map of 6 DOF poses corresponding to ArUco markers positioned in the environment. 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 [8] and later based on the extraction of planar features from 3D point clouds generated by a tilting laser scanner [10], [9].

In this work, the data association problem does not have to be addressed, since the ArUco readings provide unique ids for the observations and the landmarks. This way, loop closure is facilitated and enhanced robustness can be achieved with a not very cumbersome algorithm which showed nice empirical results in our initial tests.

Non linear state and observation models are used. At each iteration k, the prediction of the pose state x (6 DOF) is given by:

$$\tilde{\mathbf{x}}_\mathbf{k} = f(\mathbf{x}, \mathbf{u})_{\hat{\mathbf{x}}_{\mathbf{k-1}}, \mathbf{u_k}} = \hat{\mathbf{x}}_{\mathbf{k-1}} \oplus \mathbf{u}, \tag{1}$$

where the $\oplus$ operator corresponds to the composition of relative transformations in the 6D space. The noise in the odometry measurements is considered as Gaussian white noise (as required to apply the EKF), and the odometry increment $\mathbf{u}$ is represented as $\mathbf{u} \sim N(\hat{\mathbf{u}}, Q)$.

The observation model is defined by the following innovation vector for an association of observation $\mathbf{o_i}$ and map landmark $\mathbf{l_j}$:

$$\mathbf{h_{i,i+5}} = \tilde{\mathbf{x}} \oplus \mathbf{o_i} - \mathbf{l_j} \tag{2}$$

The correction of the pose state is obtained by the update equation:

$$\hat{\mathbf{x}}_\mathbf{k} = \tilde{\mathbf{x}}_\mathbf{k} - W\mathbf{h_k} \tag{3}$$

where W is the Kalman gain matrix of the system. The covariance matrices are updated at each stage of the filter as required [24].

The environment is assumed to be static except for the presence of other drones. The accumulation of drift error if the drone is not able to detect ArUco markers all the time may require the incorporation of a forgetting mechanism so that the drone can navigate safely with local maps. In our tests thus far this has not been necessary due to the addition of extra ArUco markers over the floor, but this should be further investigated.

The input parameters of the algorithm (initial pose, covariance values, global poses and ids of the known landmarks) are read from an XML file, by means of the pugixml library

[16]. The corrected absolute pose of the drone and the list of global poses of the landmarks belonging to the map are obtained as output of this module. Other nodes subscribe to these topics, as shown in Fig. 2.

C. Drone's obstacle generator

Once the position of the unknown ArUco landmarks is obtained, they are processed in order to obtain higher level geometrical features in 2D to be used as obstacles by the trajectory planner. The map of obstacles is rebuilt at every iteration.

To do so, some prior information is required. Each of the obstacles is given a unique id and the ids of the ArUco markers belonging to it are provided. The default radius of the poles is known. The poles are modeled with circles given by the coordinates of their center and their radius $c(x_c, y_c, r)$, while the walls are modeled with rectangles given by the coordinates of the center, the width and the length $R(x_c, y_c, w, l)$.

Given the observation of a landmark $\mathbf{l_j}$ belonging to pole i, an initial estimate of the circle i is very easily obtained:

$$(x_{c_i}, y_{c_i}) = \mathbf{l_j} + r\mathbf{dir} \tag{4}$$

with $\mathbf{dir} = (cos(yaw), sin(yaw))$. This initial estimate is further refined by the mean value of incorporating subsequent landmarks belonging to the same pole.

The distribution of the ArUcos of the windows has to be known more precisely. Currently, two different options are supported: the first solution is to place the ArUco markers at the corners of each window (with a predefined order) and the second solution is to place one ArUco marker at each side of each window and two ArUco markers below each window (also with a predefined order). The second option seems to work best due to the fact that the AR Drone presents a horizontal field of view wider than the vertical field of view. Basic geometry is applied in order to obtain the rectangle models of the wall.

D. Drone's Trajectory Planner and Collision Avoidance

This module computes a free collision 2D trajectory (horizontal coordinates x and y) to reach the current mission point.

The module works as follows: a free of obstacles Probabilistic Road Map (PRM) [7] of the 2D map is generated off-line. The advantage of using a PRM instead of a fixed-cell decomposition is that you can select the number of nodes in the graph and their neighborhood. Also, if the robot is moving through a zone with a lot of obstacles, new nodes can be added.

Once the free of obstacles graph is created, an A-Star algorithm [19] searches the path using a potential field map function as cost function. This potential field map is computed as a sum of one component that attracts the drone to the end of the obstacle zone and another component that repels the drone from any obstacle. The usage of a search algorithm (A-Star) instead of the potential field map alone [17], avoids the problem of the local minimum.

Three kinds of obstacles are considered. The first type of obstacles are the fixed and previously known obstacles which are set during the module startup and are obstacles and never change their previously known position. The second type are the fixed and unknown obstacles that are received from the obstacle generator whose position could change over time, depending on how precisely the ArUcos' pose is determined by the localization module. The last type are the unknown and moving obstacles that are other drones and are only considered in the path planning if they are close to the drone. Other drones' positions are received through the hypothalamus module.

Once the path is calculated using the A-Star algorithm, it is post-processed in order to obtain a shorter and more direct path, avoiding the noise produced by moving the robot from node to node of the PRM. The post-processing is done using the value of the potential field map. Fig. 4 shows a sample trajectory obtained by employing the exposed algorithm.

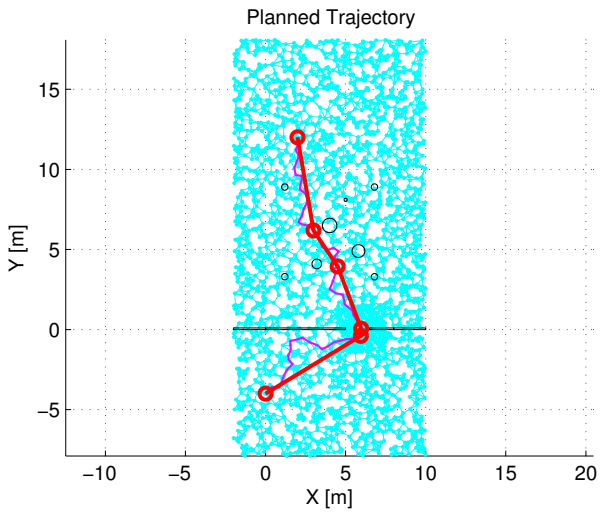

Fig. 4. Planned trajectory. In black, obstacles; in blue, the PRM; in magenta, the solution of the A-Star; in red, the post-processed trajectory

When new drones' poses or new obstacles' positions are received, the planner checks if the new obstacles are outside the planned trajectory and if the drone is following the path. Otherwise, the trajectory is re-planned.

With this algorithm we solve the problem of the path planning and the collision avoidance, being able to navigate safely in the map using the Trajectory Controller module. An important work in this field is described in [14].

E. Drone's Mission Planner

The mission planner allows the operator to define a mission as a set of separate tasks; which are, in turn, fully described by a set of numeric parameters. The mission definition requires a xml file where the mission is defined. It has different available tasks such as: take off, land, hover, sleep or move.

This module interacts with the trajectory planner module, the localization module when moving and with the hypothalamus module.

F. Drone's Hypothalamus

This module implements low-level intelligence such as: monitoring the state of the other modules of the swarm agent and presenting a simple interface between the mission planner and other modules. Some of the commands that the mission planner can achieve through the hypothalamus module are: setting up the whole system to an active flying behavior (including the start-up of all other modules and the trajectory controller); and also simpler commands such as take-off, land or hover.

This module also implements the communication between its swarm agent and the rest of the swarm. Currently the communication is limited to sharing each drone's current pose.

IV. Simulation Results

This section shows the final result of a series of simulations that were run to determine the maximum number of drones that could be flown simultaneously on the IMAV2013 competition environment. Taking advantage of the modularity of our architecture, which was eased by the usage of ROS during design, the simulations were run on most of the actual software architecture.

In order to run such simulations, we developed an "AR Drone 2.0 simulator module" and a "Drone's Aruco Eye simulator module":

- The drone 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. Most of the modes are substituted by simple behaviors, except for the flying mode where a dynamic model explained in [20] is used.
- The "Aruco Eye simulator" implementation mimics the Aruco Eye module's interface. It implements a simple set of visibility rules, such as the requirement that the Aruco has to be in front of the simulated drone and a range of distances were the visual marker is considered to be detectable by the software. A part from these, the "Aruco Eye simulator" just outputs a noisy measurement of the drone's pose with respect to the known visual markers, which position is 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 he behavior of drones in real flights is shown in Fig. 5. Then, the rest of the modules are the same as during flights, and thus can be tested and debugged without preparing experimental flights. Two examples of this simulations, where 5 drones fly in a simulated replica of the IMAV2013 environment are shown in Figs. 6 & 7. In both simulations all the drones were able to accomplish the mission successfully, the small number of conflicts during execution was achieved by timing the launch of the drones to one launch about every 15 seconds. The conflicts during navigation where a drone had to avoid another one are easily perceived, because they stand out from the normal execution of the mission

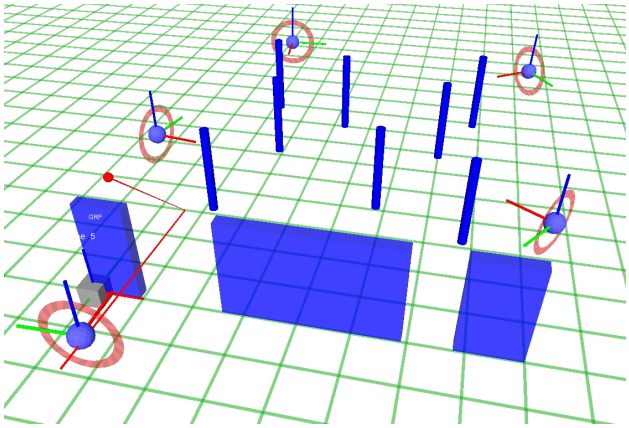

Fig. 5. Visualization in Rviz of a simulated flight of 5 drone in a replica of the IMAV2013 environment. Drone axes: red x axis, green y axis and blue z axis. The blue cylinders represent the map columns, the blue rectangles represent the wall and its windows, and the green grid represents the floor. The red line shows the current planned trajectory for the selected drone and the red dot is the current mission point.

A second capability of these simulations is that they allow to estimate the limits of our swarm solution, except for actual flight dynamics, measurement noise and other problems related to real flights. In order to determine the maximum number of drones that could simultaneously fly in the Indoors Autonomy Challenge, a set of simulations on a replica of the IMAV2013 environment was realized. Two of such simulations with a swarm of five drone are shown in Figs. 6 & 7. In order to interpret these figures, the modules specifications have to be taken into account, which are described in the Arhictecture Software Modules section (Sec. II). From them, the overall expected behavior of the drones is:

- Each drone follows a sequence of waypoints given by its mission specification. The mission planner executes it sequentially.
- During execution, the planner will attempt to find trajectories that are collision-free. If it fails, the drone will be controlled to stay in the current position.
- If other swarm agents enter the current trajectory, the planner will detect it and will stop the drone and attempt to find a collision-free trajectory.
- If another swarm agent is on the current goal waypoint, the drone is commanded to stay in the current position. A new trajectory is planned when the goal position is again free.

This intelligence does not allow to solve every possible navigation conflict between the swarm agents. However, if the mission is specified correctly the conflicts will not occur often. In Figs. 6 & 7, 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 IMAV2013 competition, which is the following sequence of tasks:

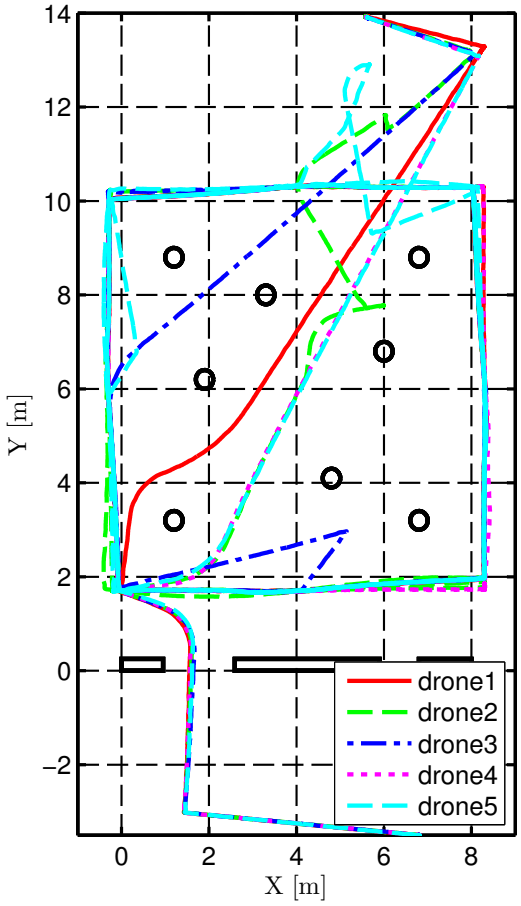

Fig. 6. Simulated flight where five drones flew simultaneously to perform navigation tasks in a replica of the IMAV2013 environment. The poles are plotted as black circles, and the wall is plotted by three black rectangles, with the two free collision passages represented as windows. The size of the 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 drone is shown as a distinct line plot, which color and line style are specified in the figure's legend. In this simulation, all the drones managed to accomplish their respective missions. The conflicts during navigation were 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 on drone3 making a detour during 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. A video of the visualization of this simulated flight can be found in the website `http://www.vision4uav.com/?q=node/386` .

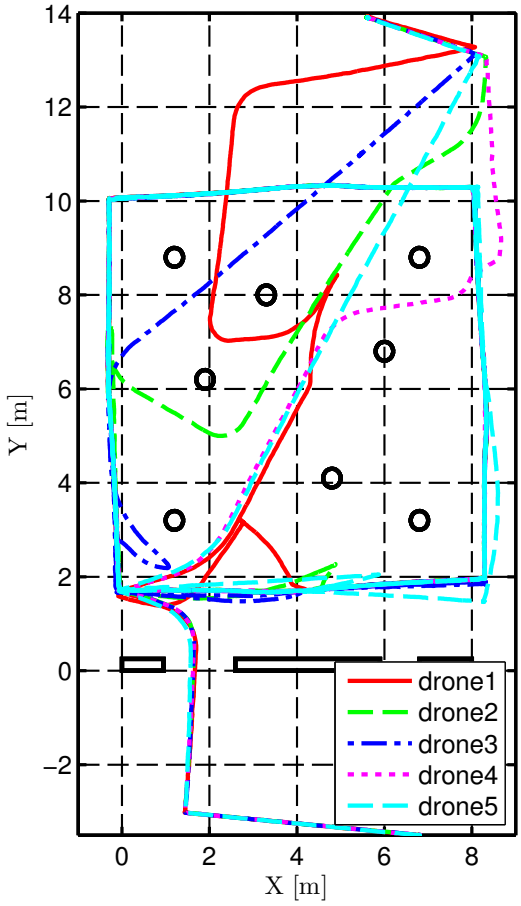

Fig. 7. Simulated flight where five drones flew simultaneously to perform navigation tasks in a replica of the IMAV2013 environment. The interpretation of this is the same as for Fig. 6. Again, in this second simulation, all the drones managed to accomplish their respective missions. The main difference between both simulations is the number of navigation conflicts that occurred that occurred during the mission execution.

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),
3) perform laps around the poles area for 5 minutes,
4) cross the poles area moving towards the upper right corner of the map,
5) move to a final location and land.
6) 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.

The laps around the poles area are specified by checkpoints in the corners and the middle points of the square lap. The results of the simulations for 1 to 7 seven drones, with the above described mission specification, are the following:

- 2-4 drones: the drones will usually synchronize during mission execution solving the conflicts when they occur.
- 5-6 drones: the drones do sometimes enter a conflictive situation, because of a lack of initial synchronization. While the time advances, the drones become synchronized and they are able to navigate the laps in a continuous manner. Two of these simulations are shown in Figs. 6 & 7. 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 much more navigation conflicts than in the case of a five drones simulation.
- 7 drones: once the swarm is synchronized to perform the laps, the navigation is not continuous. During lap execution there are only two drones navigating at any given time, the other 5 drones are staying on their current positions waiting for their checkpoint to be free.

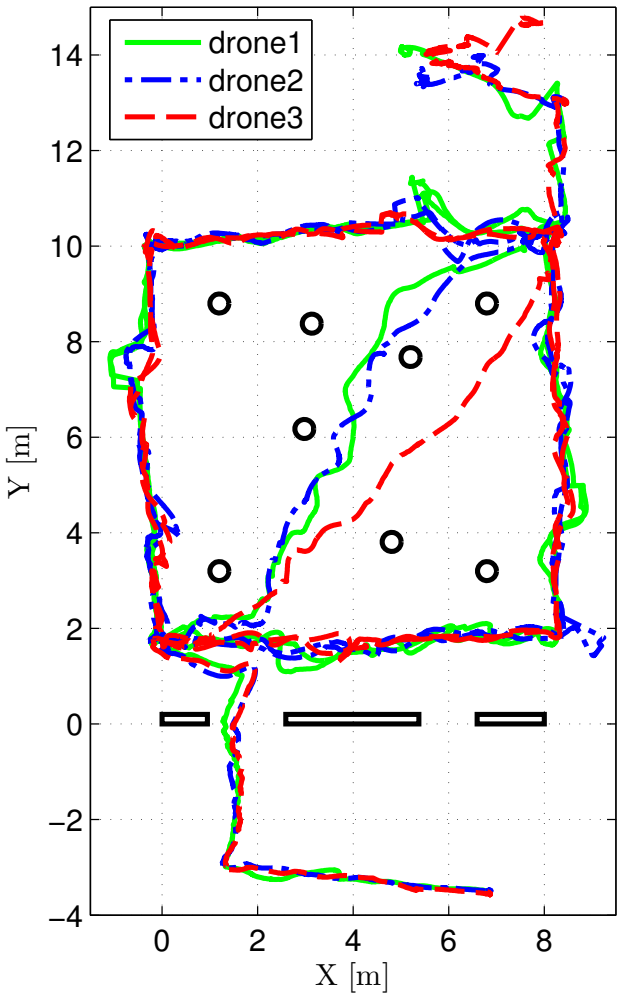

Fig. 8. Experimental flight where three drones flew simultaneously to perform navigation tasks in a replica of the IMAV2013 environment. The same flight is shown in Figs. 8 & 9. The poles are plotted as black circles, and the wall is plotted by three black rectangles, with the two free collision passages represented as windows. The 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 drone is shown as a distinct line plot, which color and line style are specified in the figure's legend. In this experimental flight, all the drones managed to accomplish their respective missions. No navigation conflicts occurred thanks to the 15 seconds timing between the launches of each drone. A video of this flight can be found in the website `http://www.vision4uav.com/?q=node/386` .

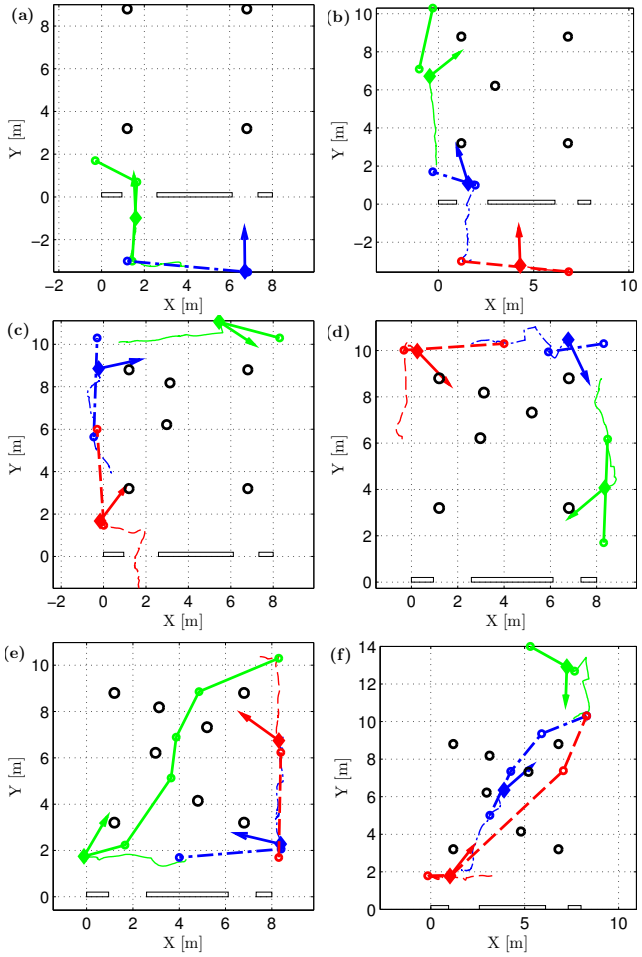

Fig. 9. Experimental flight where three drones flew simultaneously to perform navigation tasks in a replica of the IMAV2013 environment. The same flight is shown in Figs. 8 & 9. These subfigures, labeled a-f, are ordered in time, and each drone's plots have a different color and line style. The drones ordered by their launch times are plotted with: (first drone) green-solid lines, (second) blue-dash dotted lines and (third) 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 where the drone is looking in the plotted position. As it is shown the drone is usually commanded to look at one of the four known corner columns of the map. (a), (b), (c) and (d) show how the first drone was able to cross the big window, and then performed laps around the poles. The second and third drones, initialized 15 and 30 seconds later in (a) and (b), performed the same navigation missions. The three drones are not synchronized by a high level intelligence/planner and a timed sequence of take-offs is beneficial for the overall mission execution. The unknown poles localized by the first drone are shown in these figures. (e) and (f) show the final stages of the mission were the drones finish their last lap and, afterwards, they have to cross the unknown obstacle area. They have to plan and execute an obstacle-free trajectory through this area. A video of this flight can be found in the website `http://www.vision4uav.com/?q=node/386` .

From the previous analysis of the simulations, it was concluded that the optimum number of drones for this mission is 5. With this swarm size, the drones tend to execute their individual missions almost with no interruptions, leading to a maximum score for our swarm solution in the IMAV2013 Indoors Challenge environment.

V. Experimental Results

In this section we describe two experimental flights to showcase the capabilities of our solution, which was demonstrated in the IMAV2013 competition. The mission specification for all the presented experimental flights is almost identical to the mission specification described in the Simulation Results section (see the enumerated sequence of tasks written in Sec. IV). The figures show the IMAV2013 environment in black. The estimated poses or executed trajectories and the current planned trajectories of each drone are shown with line plots.

The image shown in Subfig. 1(c) corresponds to the first experimental flight, which is shown in Figs. 8 & 9. More specifically the photo corresponds to the part of the flight shown between Subfigs. 9(e) & 9(f), when the drones are

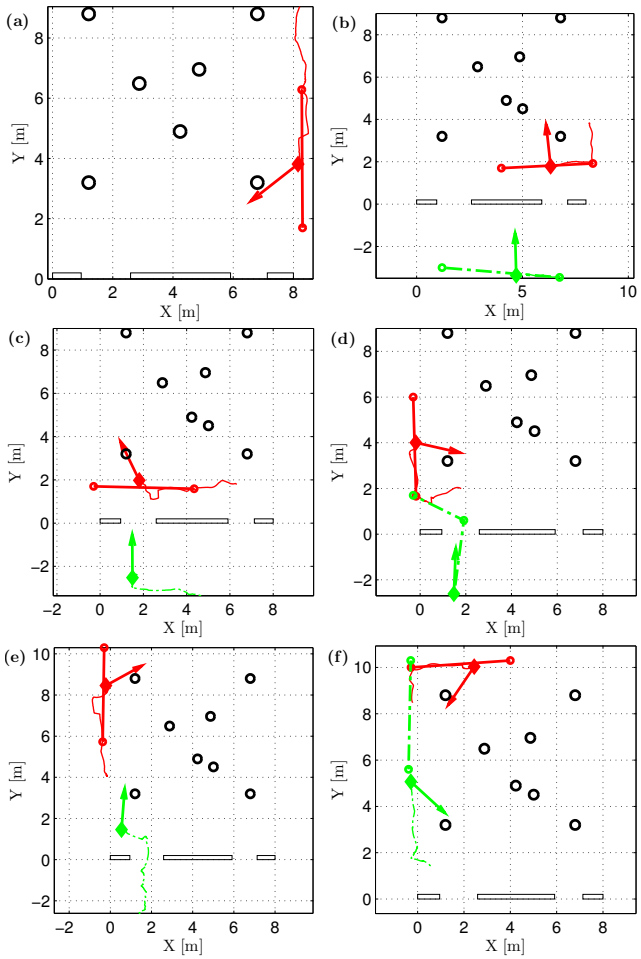

Fig. 10. Experimental flight where three drones flew simultaneously to perform navigation tasks . These subfigures, labeled a-f, are ordered in time, and each drone's plots have a different color and line style. The drones ordered by their launch times are plotted with: (first drone) red-solid lines and (second) green-dash dotted lines. The planned trajectory is shown with a thicker width than the actual executed trajectory. All subplots show how the first drone is performing laps over time. The second drone is launched on subfigure (b) has started the mission late. In Subfigs. (c) & (d) the first drone is blocking the window passage to the second drone, so it has to wait until the way is clear. The planner detects the first drone as an obstacle and thus, does not command a trajectory until the first drone is out of the way. As shown in Subfigs. (e) & (f), when the window passage is clear, the second drone crosses the window and starts performing laps. A video of this flight can be found in the website `http://www.vision4uav.com/?q=node/386` .

attempting to cross the unknown poles area. In the experiment, three swarm agents fly simultaneously performing various navigation tasks in a replica of the IMAV2013 environment. The launch of the drones was timed so that they performed their respective mission without any navigation conflicts. In this case all the drones were able to perform the full mission successfully, including the localization of the unknown elements of the environment and the obstacle avoidance tasks. This experiment was designed to showcase the capability of the swarm agents to fly relatively close to each other and perform the navigation mission when they are launched following the nominal schedule for the realization of the IMAV2013 Indoors Autonomy Challenge. More details of the execution of this mission can be read on the caption of the experiment figures.

The second experiment, shown in Fig.10, showcases the capability of each swarm agent to avoid collisions with other agents. Since there is no central intelligence in our swarm architecture, this capability is what allows the swarm agents to synchronize their navigation around the poles area and successfully accomplish the mission. A photo of this experiment is shown in Subfig. 1(d). As shown in Fig.10, the second drone is launched too late and enters a navigation conflict when it attempts to traverse the big window, because at that the time the first drone is performing a lap just in front of the window. Then, the trajectory planner of the second drone determines that there is no free-collision path to cross the window and commands the drone to stay in position. Finally, when the first drone is no longer on the way, the second drone can proceed with mission execution when its planner determines a new collision-free trajectory to traverse the window. This intelligence is not able to perform well in every situation, however, the mission was specified so that the swarm agents would be able to perform the mission successfully most of the time. For example, as described in the Simulation Results section (Sec. IV), all the drones perform laps in the same direction.

Our experiments show that the swarm is able to navigate in areas with known obstacles, cross windows and navigate near obstacles. In addition, the swarm agents can cross parts of the environment where unknown poles were located, more specifically the middle area of the IMAV2013 environment, see Figs. 1 & 9. Also, as showcased in the experiment shown in Fig. 10, the drones know each other's position, and they are capable of mutual obstacle avoidance without the requirement of a high-level synchronization.

VI. Future Work

Some ways to improve the presented architecture are the following:

- Design interface modules for other multirotors, which could carry more sensors other than a front-facing camera. This modules would have to comply with part of the interface offered by the AR Drone such as its flying modes: take-off, taking-off, hovering, flying, landing, landed and emergency.
- Solve the localization problem using other visual-based techniques, such as stereo vision, mono-camera SLAM or SLAM based on the utilization of laser rangefinders.
- Design, implement and test different swarming behaviors.
- Design, implement and test more complex mission planners, which could detect better solutions to navigation conflicts during mission execution or that could handle different missions other than pure navigation.

VII. Conclusions

This paper presented an overview of a whole swarm system designed to autonomously complete the Indoors Autonomy Challenge of the IMAV2013 competition. The system is

low-cost -employing Parrot ArDrone 2.0 quadrotors without any extra sensors- and the deployment and setup are quite easy and straightforward due to the fact that only a limited number of known external ArUco visual markers has to be put in place.

The ArUco markers are used for localization and mapping, improving the pose estimation obtained from IMU data and optical flow by means of an EKF based method. The resulting map of ArUco markers is converted to higher level 2D geometrical obstacles used by a trajectory planner combining probabilistic roadmaps, the potential field map algorithm and the A-Star algorithm. All the drones have access to the global position of every other drone in the team. The corresponding obstacles are incorporated to obtain a safe trajectory. A robust mid-level controller employs the target global position given by the trajectory planner and the corrected pose of each drone in order to drive them to their respective goals, defined by a mission planner module. The system design and implementation is based on ROS, which makes code sharing and module reuse easier.

This paper has two main contributions. The first contribution is to present our vision-based quadrotor swarm solution for the 2013 International Micro Air Vehicle Indoor Flight Competition, which was awarded with the First Prize in the Indoors Autonomy Challenge. Our solution allowed to simplify the complexity of some modules such as the localization and partner detection capabilities. It was robust enough to work properly during the competition, but encountered problems due to its dependency on the WiFi links of the AR Drones 2.0. The second contribution is that the presented architecture has been made publicly available in the website: `http://www.vision4uav.com/?q=quadrotor_stack`. The modularity of the architecture allows to focus on the development of specific functionalities and test them along with the rest of the architecture. The authors hope that the public stack will benefit other developers in their research on swarming behaviors for small UAS among other topics.

REFERENCES

[1] ardrone autonomy ros stack. `https://github.com/AutonomyLab/ardrone_autonomy/`.

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

[3] Imav 2013 flight competition rules. `http://www.imav2013.org/index.php/information`.

[4] Parrot ardrone 2.0 web. `http://ardrone2.parrot.com/`.

[5] *The Navigation and Control Technology Inside the AR.Drone Micro UAV*, Milano, Italy, 2011.

[6] G. Conte. *Vision-Based Localization and Guidance for Unmanned Aerial Vehicles*. PhD thesis, Linkopings universitet, 2009.

[7] R. Motwani D. Hsu, J.C. Latombe. Path planning in expansive configuration spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation, pp. 27192726*, 1997.

[8] P. de la Puente, D. Rodriguez-Losada, L. Pedraza, and F. Matia. Robot goes back home despite all the people. In *Proc. 5th. Conference on Informatics in Control, Automation and Robotics ICINCO 2008 Funchal, Portugal*, pages 208–213, 2008.

[9] P. de la Puente, D. Rodriguez-Losada, and A. Valero. 3D Mapping: testing algorithms and discovering new ideas with USARSim. In *USARSim workshop, IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

[10] P. de la Puente, D. Rodriguez-Losada, A. Valero, and F. Mata. 3D feature based mapping towards mobile robots enhanced performance in rescue missions. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2009.

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

[12] J. Faigl, T. Krajník, J. Chudoba, M. Saska, and L. Přeučil. Low-Cost Embedded System for Relative Localization in Robotic Swarms. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2013.

[13] Willow Garage. Ros: Robot operating system. `www.ros.org/`.

[14] Daniel Hennes, Daniel Claes, Wim Meeussen, and Karl Tuyls. Multi-robot collision avoidance with localization uncertainty. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 147–154. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[15] L. Jayatilleke and N. Zhang. Landmark-based localization for unmanned aerial vehicles. In *IEEE International Systems Conference (SysCon'13)*, pages 448–451, 2013.

[16] A. Kapoulkine. pugixml. `http://pugixml.org/`.

[17] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1991.

[18] G. Mao, S. Drake, and B. D. O. Anderson. Design of an Extended Kalman Filter for UAV Localization. In *Information, Decision and Control, 2007 (IDC'07)*, pages 224–229, 2007.

[19] B. Raphael P. E. Hart, N. J. Nilsson. A formal basus for the heuristic determination of minimum cost paths. *IEEE Transactions on SYstems Science and Cybernetics*, 4(2):100–107, 1968.

[20] Jesús Pestana. On-board control algorithms for Quadrotors and indoors navigation. Master's thesis, Universidad Politécnica de Madrid, Spain, 2012.

[21] Jesús Pestana, Ignacio Mellado-Bataller, Jose Luis Sanchez-Lopez, Changhong Fu, Iván F Mondragón, and Pascual Campoy. 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, 2014.

[22] P. Rudol. Increasing autonomy of unmanned aircraft systems through the use of imaging sensors. Master's thesis, Linkoping Institute of Technology, 2011.

[23] Jose Luis Sanchez-Lopez, Jesús Pestana, Paloma de la Puente, Adrian Carrio, and Pascual Campoy. Visual quadrotor swarm for the imav 2013 indoor competition. In Manuel A. Armada, Alberto Sanfeliu, and Manuel Ferre, editors, *ROBOT2013: First Iberian Robotics Conference*, volume 253 of *Advances in Intelligent Systems and Computing*, pages 55–63. Springer, 2013.

[24] J. De Schutter, J. De Geeter, T. Lefebvre, and H. Bruyninckx. *Kalman Filters: A Tutorial*, 1999.