Visual Quadrotor Swarm for IMAV 2013 Indoor Competition

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

***Abstract*— This paper presents a low-cost framework for visual quadrotor swarm prototyping which will be utilized to participate in the 2013 International Micro Air Vehicle Indoor Flight Competition. The testbed facilitates the swarm design problem by utilizing a cost-efficient quadrotor platform, the Parrot AR Drone 2.0; by using markers to simplify the visual localization problem, and by broadcoasting the estimated location of the swarm members to obviate the partner dectection problem. The development team can then focus their attention on the design of a succesful swarming behaviour for the problem at hand. ArUco Codes [2] are used to sense and map obstacles and to improve the pose estimation based on the IMU data and optical flow by means of an Extended Kalman Filter localization and mapping method. A free-collision trajectory for each drone is generated by using a combination of well-known trajectory planning algorithms: probabilistic road maps, the potential field map algorithm and the A-Star algorithm. The control loop of each drone of the swarm is closed by a robust mid-level controller. A very modular design for integration within the Robot Operating System (ROS) [13] is proposed.**

I. Introduction

The motivation of the presented work is to design a low-cost framework for quadrotor swarm prototyping. The framework is designed to ease the main issues related to working with a multirobot system, which are obstacle avoidance and partner detection. The focus is to allow the team of designers to try various swarming behaviours on a real robotic swarm, so that the advantages of the proposed strategy can be experimentally demonstrated on an early stage of the development process. In order to obtain a cost-effective testbed the AR Drone 2.0 quadrotor was selected as the aerial swarm agent, as it is a low-cost but competitive platform. The second motivation is to participate in international robotics competitions using simple, but swarming, visual aerial robotic agents. More specifically, the presented work has been implemented to participate in the 2013 edition of the International Micro Air Vehicle (IMAV) Flight Competition.

There exists a large variety of applications which require a robotic system to densely navigate a wide area. Such applications can benefit from a swarming approach for the required data gathering of the problem at task, taking benefit from the robotic population. For instance, such an approach could be applied to security and surveillance tasks of middle sized areas.

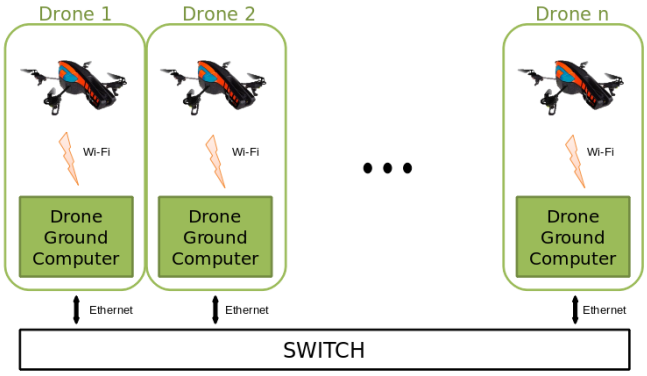

Fig. 1. The swarm is composed by identical robotic agents, which are composed of an AR Drone 2.0 which is commanded via Wifi from a ground station. The ground stations can communicate with each other under ROS via LAN, where one of them is running the roscore.

The IMAV Flight Competition is one of the most relevant competitions in Europe in the field of Autonomous Aerial Robotics and in the field of Small Remotely Piloted Air Systems (sRPASs). The Computer Vision Group (CVG) participated last year [20], in the 2012 edition, showing the potential and the research experience of the group. The learning experience obtained from the indoor dynamics competition encouraged us to keep working in the same line 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. This year's rules are significantly different with respect to 2012 edition. In the IMAV 2013 edition there is 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 obstacle poles). The indoor competition includes various challenges, among others: flying through a window, flying through the obstacle zone, target detection and recognition, path following and precision landing.

After a deep analysis of the contest characteristics, a Visual Quadrotor Swarm was selected as the best option to join the 2013 IMAV indoor flight competition. A swarm composed by a significant number (more than 5) of relatively simple quadrotors (see section II) is going to be used to

*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, Centro de Automática y Robótica, CSIC-UPM jesus.pestana@upm.es, pascual.campoy@upm.es, www.vision4uav.com

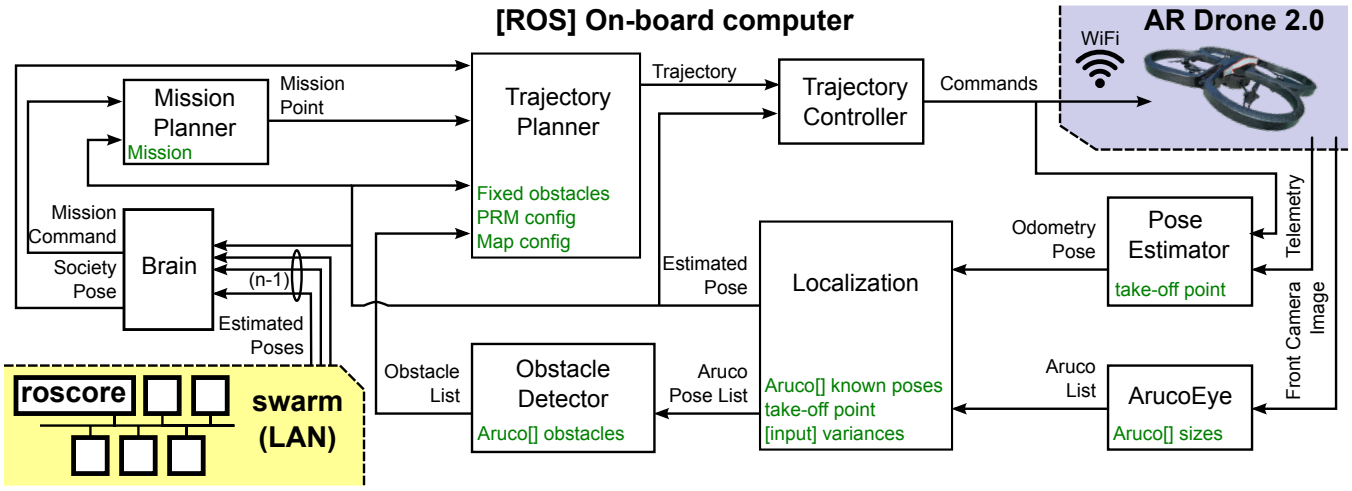

Fig. 2. Robotic agent software architecture. The ROS nodes of each agent are executed on a ground station which commands one AR Drone via WiFi. Each white box represents a module, and the green text inside it are configuration files. The localization module is implemented using an EKF which fuses the odometry based estimation with the ArUco visual feedback. This modules broadcasts the estimated pose to the mission and trajectory planning modules, to the controller modules and to the other robotic agents. The brain module receives the estimated position of the other robots and communicates it to the trajectory planner.

achieve all missions except for the dropping one (mission 6). Additionally, as we decided to work with a visual swarm, a external visual aid described in section III-B is needed to solve the localization problem. Our swarm is going to be fully automated, and thus the level of autonomy is going to be "Autonomous Mission Control", requiring only one operator to start up and supervise/monitor the whole system (just in case something goes wrong, the operator could stop independent elements of the swarm, or the whole system).

II. System Description

The system is composed by a swarm of autonomous Unmanned Aerial Vehicles (UAVs). Each drone is autonomous enough to complete a previously defined mission avoiding obstacles and collisions with the other drones of the swarm. There is no high-level intelligence that controls or synchronizes the drones, the system then being a pure swarm. All the drones of the swarm share its pose in order to avoid the partner detection problem.

Each robotic agent is composed of the quadrotor platform and a ground station, see Fig. 1. The agent is composed of separate modules which work using the Robotics Operating System (ROS), see [13]. The Parrot AR Drone 2 [4] was selected as robotic platform, the characteristics of this platform 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 the drone's ground computers are communicated via ethernet.

III. Drone's Modules

In this section the main modules, see figure 2, of each robotic agent of the swarm are described. The Pose Estimator and the Trajectory Controller modules are explained somewhere else [20], [19].

A. Drone's Aruco Eye

The localization of the drone in the map is firstly estimated using the Pose Estimatmor module. Since this measure has drift, we need to correct it with an absolute measure when available using the module described in section III-B. For this purpose, we use a visual external aid: ArUco Codes (figure 3, see [2]). This library allows to calculate the 3D pose of the camera with respect to each ArUco code.

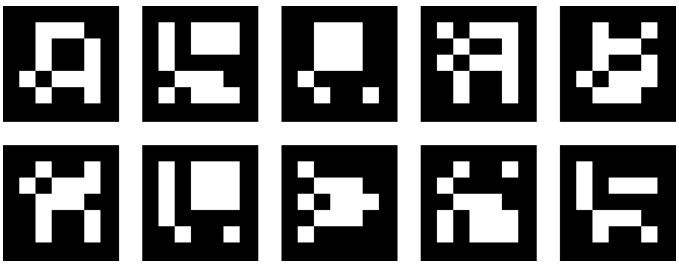

Fig. 3. Samples of ArUco Codes, which are markers used to simplify the obstacle detection and localization problems.

There are two kind of ArUco codes defined in our system. The ones whose position is previously known and the ones whose position is unknown. Some of these ArUco codes are attached to all the obstacles (fixed known obstacles and fixed unknown obstacles).

With this distribution of codes, we are able to locate the drone with respect to the known ArUco codes, and to sense and locate the unknown obstacles with respect to the drone camera.

This visual aid helps us to solve two problems at the same in quite a straight forward manner: we solved the problem of sensing all the obstacles, that is a very hard task using only computer vision, and we also avoid the visual localization problem in a general environment, which is not solved precisely yet (even though there are some approaches

like [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 [17], [21], [14], [12]. In the absence of GPS and laser sensors, visual approaches are very popular [21], [14], [12].

In the proposed system, the global localization of each drone is based on the IMU data and optical flow for the pose estimation, calculated by the Pose Estimator module. However, this measure has a drift which may be significant, so it should be corrected with more reliable information from the environment when available. For this purpose, we decided to use visual external aids, the ArUco codes [2] previously described. This library provides the 3D pose of the camera with respect to each ArUco code 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 Codes, received by means of ROS messages.

Since the environment can be partially known a priori, some fixed landmarks are employed. ArUco codes with a priori known global poses are attached to the known poles. Other ArUco codes are placed on the wall and the unknown obstacles, and others are distributed over the floor, with an inclination angle of 45° and mounted on top of a platform 15 cm high. The latter are used so as to improve the visibility of ArUco codes when navigating among the poles; several tests were performed to qualitatively select an appropiate inclination working properly for the ArDrone's camera field of view. Simple and easy-to-use accessories for a fast arrangement of the ArUco codes in the environment were designed and built.

Localization with visual external aids for UAVs has been recently proposed in other works [21], [14], [12]. The method presented by Jayatilleke and Zhang [14] requires all the landmark poses to be known a priori and only works in limited areas, employing 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, with the external marks always within the field of view. Our method was mainly inspired by the work by Rudol [21], 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 external visual aids 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 codes 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 $\mathbf{l_j}$:

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

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 [22].

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 codes 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 codes 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 [15]. The output is the corrected absolute pose of the drone and the list of global poses of the landmarks belonging to the map. 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 codes belonging to it are provided. The radius of the poles is

known. The poles are modeled with circles given by the coordinates of their center and the 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 codes at the corners of each window (with a predefined order) and the second solution is to place one ArUco code at each side of each window and two ArUco codes below each window (also with a predefined order). The second option seems to work best due to the fact that the ArDrone 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

The objective of this module is the creation of a free collision 2D trajectory (horizontal coordinates x and y) to achieve a mission.

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

Once the free-obstacle graph is created, then an A-Star algorithm [18] searches the path using a potential field map function as the cost of the algorithm. This potential field map is built as a sum of a 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 [16], avoids the problem of the local minima.

There are three kinds of obstacle considered. The first type of obstacles are the fixed and previously known obstacles which are set during the start of the module and are obstacles that never change its previously known position. The second type are the fixed and unknown obstacles that are received from the module described in III-C whose position could change over time, depending on how precisely are the ArUcos pose determined by the module III-B. The last type are the unknown and moving obstacles that are other drones and are only considered in the path planning if they are near to the drone. The positions of other drones are received through the brain (see section III-F).

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.

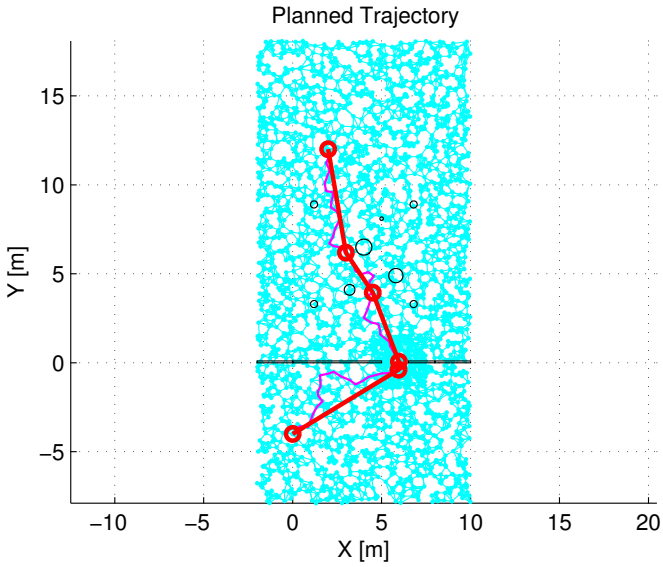

Fig. 4. Planned trajectory. In black, obstacles; in blue, the PRM; in magenta, the solution of the A-Star; in red, the post-processed trajectory

When a new pose of the drone or new positions of the obstacles 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.

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 described. It has available different tasks like take off, land, hover, sleep or move.

This module interacts with the trajectory planner (section III-D) and with the localizer (section III-B) when moving or with the brain (section III-F) otherwise.

F. Drone's Brain

This is the very high-level module of each drone. It sends high-level commands to the drone like take-off, land or hover. It also communicates with other drones, receiving the pose of each other drone of the swarm.

The drone's brain communicates with all the modules, being in charge of monitoring each module state, activating or deactivating them.

IV. Conclusions

This paper presented an overview of a whole swarm system designed to autonomously complete the indoor mission of the IMAV 2013 competition. The system is low-cost -employing Parrot ArDrone 2.0 quadrotors without any extra sensors- and deployment and setup are quite easy and straightforward due to the fact that only a limited number of known external ArUco codes is required.

The ArUco codes 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 codes is converted to higher level 2D geometrical obstacles used by a fast 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, reutilization and monitoring easier.

This paper presents two additional contributions. First, a low-cost framework for visual quadrotor swarm prototyping was described. The framework allows to separate the complexity of some modules such as the localization and obstacle detection capabilities, from the swarm behaviour development and experimental testing. Second, the framework is used to prototype an aerial visual robotic swarm approach to security and surveillance applications, which is then used to participate on the IMAV2013 competition.

Some initial tests were carried out in simple environments, showing good performance results. We are currently building a real environment like the one shown in the competition rules in order to conduct more complex and realistic experiments and validations. Future work also includes designing and implementing a good graphical user interface for mission management. If the visibility of ArUco codes is not good in the whole area, the localization method may be modified so that old unknown obstacles get removed from the map. The trajectory planning algorithm may be updated with novel ideas.

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

[15] A. Kapoulkine. pugixml. http://pugixml.org/.

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

[17] 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.

[18] 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.

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

[20] Jesus Pestana, Ignacio Mellado-Bataller, Changhong Fu, Jose Luis Sanchez-Lopez, Ivan Fernando Mondragon, and Pascual Campoy. A general purpose configurable navigation controller for micro aerial multirotor vehicles. In *ICUAS 2013*.

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

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