A Control Architecture for a Micro Aerial Vehicle Intended for Vessel Visual Inspection

Emilio Garcia-Fidalgo, Francisco Bonnin-Pascual and Alberto Ortiz[1]

Abstract— **Large-tonnage vessels need to be revised periodically in order to detect defective situations, such as cracks, coating breakdown or corrosion that could lead to a catastrophe. The EU-FP7 project MINOAS is designed to develop a fleet of robotic platforms to automate this inspection process. This paper presents a Micro Aerial Vehicle platform to be used as part of this fleet. The control architecture adopted for the MAV and the key challenges that have guided us towards this solution are described and discussed, while hardware, software and network configurations are also exposed. Finally, experimental results proving the suitability of the design are reported.**

Keywords— **Micro Aerial Vehicle, UAV, Vessel Inspection, Control Architecture**

I. Introduction

MARITIME transport is one of the most cost effective ways for carrying goods around the world. Sometimes, these vessels suffer accidents and, from time to time, they imply catastrophic consequences in personal, environmental and financial terms. The major cause of shipwrecks is structural failure. For this reason, *Classification Societies* impose periodic inspections in order to assess the structural integrity of these ships.

Nowadays, to perform a complete hull inspection, a vessel has to be emptied and situated in a dockyard, where scaffolding, lifts and movable platforms need to be installed to allow the workers for close-up inspection of all the different metallic surfaces and structures. Taking into account the huge dimensions of some ships, this process can mean the visual assessment of more than 600,000 m^2 of steel. Besides, surveyors are working in environments whose access is usually difficult and the operational conditions turn out to be sometimes extreme for human operation.

The main goal of the EU-funded FP7 MINOAS project is to develop a set of robotic platforms with different locomotion capabilities in order to facilitate inspection and maintenance tasks to both surveyors and owners. Within this context, a semi-autonomous *Micro Aerial Vehicle* (MAV) has been developed to provide a first overview of the state of the hull, taking pictures at different positions along the vessel structure. These pictures are processed on a base station using algorithms for corrosion and crack detection, also developed within the context of this project [1], [2], [3].

[1]This work is partially supported by FP7 project SCP8-GA-2009-233715, FSE and CAIB through PhD scholarships. Department of Mathematics and Computer Science, University of Balearic Islands, Spain. {emilio.garcia, xisco.bonnin, alberto.ortiz}@uib.es.

This work presents the configuration and control architecture of the MAV to be adopted as part of the re-engineered MINOAS inspection process. Due to its inherent properties for flying indoors and close to other structures, quadrotors have resulted in the platform of election for this application. Experimental results towards fulfilling the intended inspection mission are as well provided and discussed.

The rest of the paper is organized as follows: Section II presents some previous MAV systems, Section III explains the key challenges for an autonomous micro aerial vehicle, Sections IV, V and VI describe the hardware and software configuration used in the design of the platform, Section VII reports some experimental results, and, finally, Section VIII summarizes the contributions of this work.

II. Related Work

MAVs have increased their popularity as a robotic platform in recent years. Their development is driven by commercial, research, government and military purposes. This kind of vehicles allows the access to hazardous environments, usually difficult to reach by humans or ground vehicles, becoming these robots an adequate solution for inspection tasks.

In order to achieve the autonomy with these platforms, a full navigation solution is required. *Navigation* involves platform stabilization, self-localization, mapping and obstacle avoidance, among others. Some recent approaches can be found in this regard in the related literature, which particularly differ in the sensor used to solve these tasks, the amount of processing that is performed onboard/offboard and the assumptions made about the environment.

Laser scanners are the most commonly used sensors, due to its accuracy and speed. In this regard, Grzonka et al [4] present an open-source solution which enables a small sized flying vehicle to operate indoors. Dryanovski et al [5] design a system also based in open-source components, showing experimental results for SLAM and 3D mapping tasks. He et al [6] present a solution for planning vehicle trajectories using an unscented Kalman filter and a laser range finder. Bachrach et at [7] expose another approach using the same kind of sensing devices.

Infrared or ultrasounds are other possible sensors to be used for navigation tasks. Although they present less noise tolerance and accuracy, several researchers [8], [9], [10] have used them to perform navigation tasks in indoor environments, being a cheaper option than laser scanners.

Vision cameras have also been used. They are the cheapest option, although at a higher computational

cost. Some examples of this approach can be found in [11], [12], [13] and [14].

Some authors have combined different sources in order to improve the accuracy of the localization process. For instance, Achtelik et al [15] present a platform which combines laser scanners and vision cameras to autonomously navigate in indoor environments.

In our case, a laser-based solution has been adopted for self-localization and obstacle avoidance, and all critical processing is performed onboard in order to reduce the use of wireless datalinks, which are not favoured within vessel holds due to the surrounding metallic structures.

III. Micro Aerial Robotics Challenges

Many algorithms have been developed along the years for navigation tasks in ground robotics. However, these algorithms are valid for slow moving robots. Unfortunately, they can not be performed directly in a MAV, because of the faster dynamics of these platforms. This section summarizes the key challenges that micro aerial robotics presents [15]. In our case, they have determined the design of both the control software of the platform and the payload sensors.

Due to its dimensions and the power of its thrusters, a MAV can not be equipped with heavy sensors, such as SICK laser scanners or high-fidelity IMUs. Then, this kind of robots have to rely on noisier sensors. Since a MAV cannot directly count wheel turns, these sensors are the only option to estimate the robot motion, which presents more uncertainty compared to their ground equivalent.

Navigation algorithms are typically computationally demanding, while the amount of computation that we can perform on the class of embedded computer systems that can be mounted in a MAV is limited. These algorithms are, thus, impossible to be executed onboard. This fact implies that a base station is needed to perform part of the navigation tasks, such as SLAM or path-planning; thus, datalinks result to be essential. In this case, wireless communications become critical and can be a bottleneck, specially for image transmission.

Another problem is related to the MAV fast dynamics. Usually, data fusion techniques, such as Kalman filters, are used to improve the vehicle state estimation from noisy sources, adding a delay to the process. While this delay is not significant in ground vehicles, it is amplified in aerial robotics due to the MAV dynamics, and, thus, has to be taken into account.

MAVs are in constan motion, so that, sensor uncertainties must be properly considered in order to avoid the platform behaviour degrade seriously.

IV. The Micro Aerial Vehicle

Among the different kinds of helicopter designs that have been proposed so far, multi-rotor configurations present several advantages over comparably scale helicopters [16]. Within these configurations, the four-rotor, or *quadrotor*, is emerging as the most popular design. Our MAV prototype belongs to this class of vehicle. Being more precise, it is based on the well-known Pelican platform from Ascending Technologies (see Fig. 1). This is a 50 cm-diameter platform with 10-inch propellers, able to carry a payload of 500g, and equipped with a barometric pressure sensor for height estimation, a GPS receiver and an inertial measuring unit (IMU), comprising a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer. Attitude stabilization control loops making use of those sensors run over an ARM7 microcontroller as part of the platform firmware; the manufacturer leaves almost free an additional secondary ARM7 microcontroller, so that higher-level control loops (e.g. a position controller) can also be run onboard.

Fig. 1. The Micro Aerial Vehicle.

The MAV has been equipped with a Hokuyo URG-UTM-30LX laser, which is able to detect a range of approximately 30 meters. This sensor is used to estimate the platform motion by matching consecutive laser scans. The device is also used, by deflection of lateral beams using mirrors, to estimate distance to the floor as well as to the ceiling. This method has been found more adequate for the application instead of using the barometric pressure sensor or the GPS, which tend to show large variations around the true height, making height stabilization difficult when the platform navigates indoors or relatively close to other objects.

The vehicle also features a set of uEye 1226-LE-C cameras in order to provide visual information during the flight: one fixed bottom-looking device allows obtaining pictures from the ground, and two additional units can be attached to a carbon fiber tube mounted at the top of the platform. This last set of cameras can adopt a number of configurations depending on the inspection mission to be performed: two front-looking cameras forming a stereo vision system, one camera looking to the front and the other looking to the ceiling, or, to save weight, a single camera looking to the front. An analog video camera operating at

5.8GHz has also been attached to provide real-time information about the state of the vessel.

Finally, the vehicle carries an additional processing board which avoids sending sensor data to a base station, but process it onboard avoiding communications latency inside critical control loops. This processor will be referred to as the high-level processor from now on. (The configuration shown in Fig. 1 includes a CoreExpress pITX-SP board equipped with an Intel Atom 1.6GHz processor and 1GB RAM).

V. Platform Communications

As discussed previously, communications result to be critical for controlling the MAV. In our configuration, a cluster of laptops is used to perform all the offboard operations, so that information exchange between laptops is performed by wire and the wireless datalink is left only for the communications with the quadrotor. This configuration can be seen in Fig. 2. Moreover, only one laptop talks directly with the MAV in order to reduce multiple point-to-point communications for the same data, but they are republished by this laptop to provide the rest of computers of the cluster with the information. This configuration permits us adding new computers to the cluster as needed, ensuring there is no extra wireless communications with the vehicle.

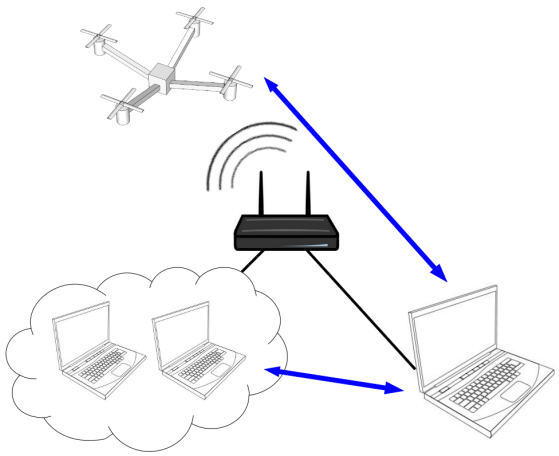

Fig. 2. Network configuration. Only one computer exchange data with the MAV, reducing the load of wireless communications. The remaining computers in the cluster talks directly with this machine in a wired way.

Sending images over a wireless network turns out to be expensive in bandwidth and time terms. For this reason, pictures are stored in the MAV during an inspection mission. In a first approach, visual odometry techniques were employed to estimate the robot motion. These algorithms required a lot of computation and needed to be performed offboard, which implied the corresponding delay produced by wireless communications. On this account, they were discarded and the laser-based solution was adopted.

The wireless device attached to the vehicle is connected to the atom board using a dedicated PCI Express port, avoiding wireless communications from sharing USB bandwidth.

VI. Control Software Architecture

As introduced in previous section, the control software architecture comprises at least two physically separated agents: the MAV itself and a base/ground station. More specifically, the different computational resources of the MAV run the control algorithms as detailed next (either as firmware or as software): (1) as it is well known in the Pelican, the main ARM7 controller essentially runs the low-level software taking care of attitude stabilization and direct motor control [17] (in Fig. 3, it appears as the low-level controller); (2) the secondary ARM7 controller runs the position controller described in [14] (in Fig. 3, it appears as the high-level controller); and (3) the high-level processor executes, on top of the Robot Operating System (ROS [18]) running over Linux Ubuntu, ROS nodes providing platform motion estimation as well as platform safety, interaction with the onboard platform controllers and WiFi communication with the base station. Apart from this, the base station supporting the MAV also runs ROS over Linux Ubuntu. For this configuration, ROS becomes particularly relevant as it supplies the middleware functionality for transparent messages exchange between processes. Those that can tolerate latency in the communications are executed on the base station, while critical control loops run onboard the vehicle in order to ensure minimum delay, a requirement also reported by other authors [15] to permit autonomous flying.

As well as the self-localization and mapping solution described in [5], our MAV control software has been designed around open-source components and following modularity and software reutilization principles. In this way, adapting the platform for different missions involving different payloads, or the selective activation of software modules, can be performed in a fast and reliable way. In this regard, ROS has also proved to be specially useful and, in fact, has guided the control software modularization.

As metioned above, the standard way of organizing algorithms in ROS is using *nodes*. Nodes can share information (raw or processed data) exchanging messages through a publish/subscribe mechanism implemented as *topics*. The ROS middleware comprises both data structure serialization and data messages deserialization at, respectively, the sender and receiver sides. This functionality allows a transparent message exchange among nodes that are executed in different machines.

Nevertheless, in order to save the serialization/deserialization time, tasks running onboard the robotic platform have been implemented as *nodelets*. Unlike nodes, nodelets are algorithms are compiled as plugins that are loaded into a nodelet manager, so that they all are threads of a single process and all the information is exchanged via shared memory.

Next sections comment on the details of the control architecture, whose top-level logical components are enumerated in Fig. 3.

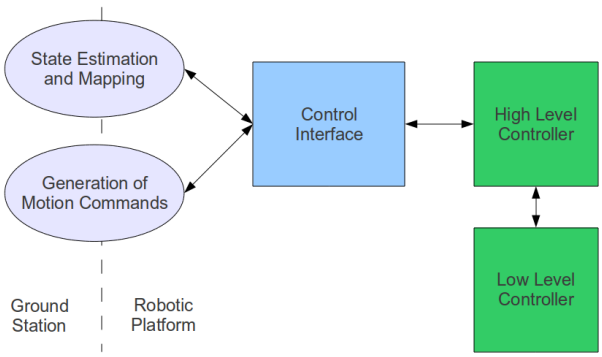

Fig. 3. Vehicle control architecture.

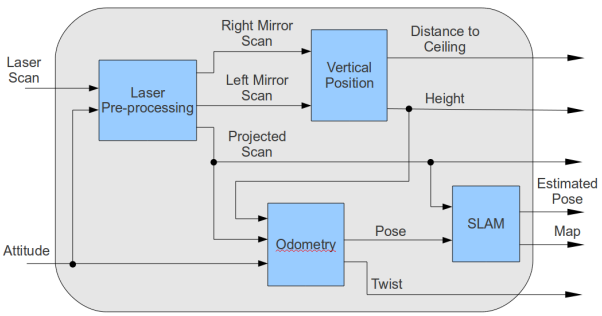

Fig. 4. State estimation and mapping.

A. State Estimation and Mapping

Fig. 4 depicts the pose estimation system. It receives scan data and attitude angles (ϕ, θ, ψ) from, respectively, laser and IMU sensors and estimates the 3D pose of the vehicle.

Within this system, the *Laser Pre-processing* module prepares raw laser scans for the rest of the system. More precisely, it is in charge of: (1) filtering the laser beams that are of no interest; (2) projecting the scans comprising the surviving beams onto the ground, using the attitude information provided by the IMU; and (3) splitting the laser scans into several fragments, so that beams reflected by the lateral mirrors are separated from beams providing information on detected obstacles ahead.

The *Vertical Position* module estimates the distance of the robot to the ground and to the ceiling. It uses, respectively, the laser beams which are deflected by the down-looking and up-looking mirrors. Apart from being useful for obstacle detection above and below the platform, depending on the mission and the environment, one or the other measurement feeds the vehicle height controller while flying, keeping constant the desired altitude or distance to the ceiling.

The *Odometry* component estimates the MAV 3D pose. First, the projected laser data is passed onto a scan matcher, which computes the platform roto-translation between consecutive scans and estimates a new 2D pose (x, y, ψ), using the yaw angle provided by the IMU as initial estimate for ψ. The 2D pose so obtained is then combined with the height and the roll and pitch angles (ϕ, θ), provided by, respectively, the laser altimeter and the IMU, to obtain a 3D pose for the vehicle. Robot speed is also estimated by this module.

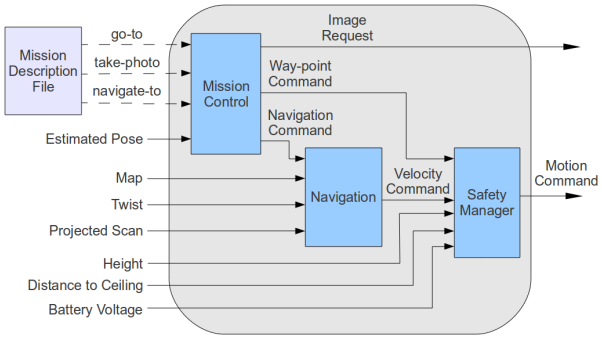

Fig. 5. Generation of motion commands.

In order to compensate the drift in the estimations produced by the scan matcher, a *Simultaneous Localization and Mapping* process is executed as part of the *SLAM* module. It receives projected laser scans and provides 2D corrections of the robot position and the environment map. Due to its high computational needs, this process runs on the base station. A further component within the *SLAM* module, responsible for monitoring the corrections provided by the SLAM, is executed onboard. This component limits the SLAM corrections to prevent the robot from moving in a too aggressive way within close environments. Furthermore, if the connection with the base station is suddenly lost, it keeps the platform operating with the last correction received. The public ROS package *gmapping*, based on [19], provides the SLAM functionality.

B. Generation of Motion Commands

This system generates the motion commands to be executed by the robot in accordance with the actions described in a mission specification file. See Fig. 5 for an overview.

The *Mission Control* component is executed on the base station and it is in charge of the accomplishment of the mission. The mission is described in an XML file as a sequence of actions. These actions can be: (1) *go-to*, which specifies a 3D pose to be attained by the vehicle, together with maximum speeds; (2) *navigate-to*, similar to the *go-to* action, but avoiding obstacles; and (3) *take-photo*, which requests for a picture to be taken by the robot using one of the attached cameras.

Just by way of illustration, the mission specification shown in Fig. 6 makes the vehicle take off at a height of 1 meter and hover for 5 seconds (line 2), take a picture and hover for 5 more seconds (line 3), move 2 meters in x and hover for 5 seconds (line 4), take a picture and hover for 5 more seconds (line 5), go home and hover for 5 seconds (line 6), and, finally, land (line 7).

A client program, which is inside the *Mission Control* module, parses the mission specification file and invokes the corresponding tasks. A new action is sent if the previous one succeeds before a specified

```
1  <mission>
2  <goto x="0.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
3  <takephoto camera="1" path="picture1.jpg" stay_time="5.0"/>
4  <goto x="2.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
5  <takephoto camera="1" path="picture2.jpg" stay_time="5.0" />
6  <goto x="0.0" y="0.0" z="1.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.1" accori="0.0" timeout="30" stay_time="5.0" />
7  <goto x="0.0" y="0.0" z="0.0" spx="0.2" spy="0.2" spz="0.2" yaw="0.79" accpos="0.3" accori="0.0" timeout="30" />
8  </mission>
```

Fig. 6. Example of a mission specification file.

timeout. Otherwise, the mission is aborted and the vehicle hovers at the attained position. This module is also responsible for handling *go-to* and *take-photo* actions. Actions of the first kind are directly processed by the *Safety Manager* module, which filters out motion commands towards the high-level controller, while actions of the second kind are sent to the camera driver for image grabbing.

Navigate-to actions are handled by the 2D *Navigation* module, which includes functionalities such as reactive obstacle avoidance and path planning. Given a two-dimensional map and a waypoint, it generates the velocity commands needed to achieve the waypoint preventing collisions. These commands are also sent to the *Safety Manager* module. The *Navigation* module is a public ROS stack [20] that has been configured for using a DWA local planner [21].

Finally, the *Safety Manager* is in charge of filtering all control commands before sending them to the *Control Interface*. Currently, it implements three safety behaviors: (1) it prevents the robot from flying too high or too close to the ceiling, (2) it monitors the battery voltage and sends commands for landing when this is lower than a safety threshold, and (3) it sends commands to make the robot hover when the wireless connection with the base station is lost.

VII. Experimental Results

This section presents some experimental results to test the suitability of the platform for the inspection tasks described in previous sections. Two experiments have been defined to represent the kind of mission to be performed by the MAV onboard a real ship for providing the surveyor with an overall view of the state of a certain hull area. More precisely, the first mission describes a sweeping task, consisting in achieving a collection of waypoints along a zig-zag-like path. The experiment, whose results can be found in Fig. 7, has been executed several times to compare results between consecutive executions. In this regard, the small differences which can be observed are due to, among other factors, the accuracy parameter that determines when an action has succeeded or not (set to 0.1 meters in this experiment).

The second experiment also consists of a sweeping task but, in this case, the robot first has to achieve the area to be inspected. To go from the home position to the place where is performed the sweeping (and viceversa), the robot navigates avoiding the obstacles that have been spread throughout the room to make the global and local planners take part and

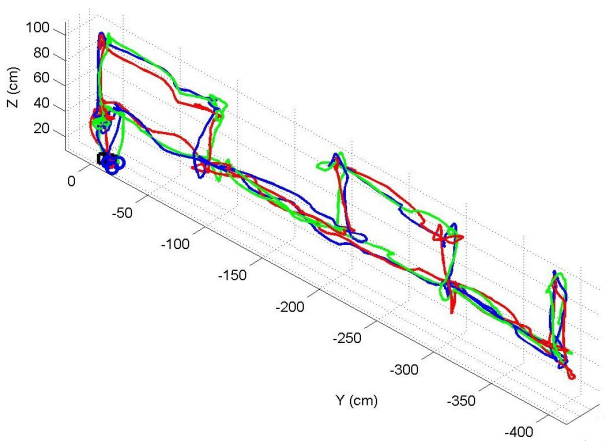

Fig. 7. Three executions of a sweeping mission, shown in different colours. The robot keeps the x coordinate almost constant during all the mission.

Fig. 8. Images of the environment used to perform the last experiment. There are several obstacles which the vehicle has to avoid in order to, first, reach the initial point of the sweeping, and then reach the home location again, once the inspection has finished.

produce safe paths towards the targets. The scenario is shown in Fig. 8, while a graphical description of the mission can be found in Fig. 9(left). As can be observed: first, the vehicle is instructed to reach a point at the other end of the room; then, a sweeping task is performed, and the corresponding images are taken; after the finalization of the inspection, the robot returns to the home position. Fig. 9(right) shows, for this experiment, the different paths followed by the vehicle during three executions of the experiment, while examples of the photos taken at the same location for two of the executions of the mission can be found in Fig. 10. The slight viewpoint changes that can be observed in the pictures are again due to the accuracy parameter, set to 6 degrees on this occasion.

VIII. Conclusions

A Micro Aerial Vehicle intended to assist human surveyors during visual inspections of vessels has

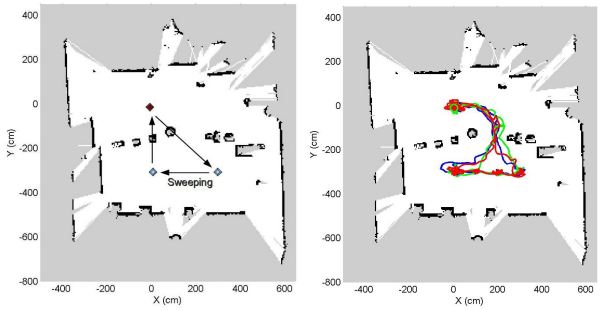

Fig. 9. (left) A map of the scenario for the third experiment. The red point indicates the beginning and the end of the experiment, the blue points show the locations to be reached by the vehicle, and arrows represent the direction of the robot. (right) Paths of three executions in different colours.

Fig. 10. Pairs of images taken at the same location in two different executions of the last experiment. The slight deviations in viewpoint are due to the orientation accuracy parameter, set to 6 degrees in the mission specification file. The drawing at the bottom shows the positions where the photos are to be taken during the inspection mission. The pairs above correspond to, respectively, locations 2, 4 and 6.

been described. It is based on a commercial platform which integrates a control architecture intended to cover the requirements imposed by the inspection missions. The details and organization of the control software have been described and discussed and results showing the suitability of the platform for the problem at hand have been reported as well.

References

[1] F. Bonnin, "Detection of Cracks and Corrosion for Automated Vessels Visual Inspection," Master's thesis, University of Balearic Islands (Spain), 2010.

[2] F. Bonnin-Pascual and A. Ortiz, "Combination of weak classifiers for metallic corrosion detection and guided crack location," in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pp. 1 –4, sept. 2010.

[3] F. Bonnin-Pascual and A. Ortiz, "An adaboost-based approach for coating breakdown detection in metallic surfaces," in *Control Automation (MED), 2011 19th Mediterranean Conference on*, pp. 1206 –1211, june 2011.

[4] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 2878–2883, 2009.

[5] I. Dryanovski, W. Morris, and J. Xiao, "An Open-Source Pose Estimation System for Micro-Air Vehicles," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 4449–4454, 2011.

[6] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a GPS-denied environment," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 1814–1820, 2008.

[7] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice, and N. Roy, "RANGE robust autonomous navigation in GPS-denied environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 1096–1097, 2010.

[8] A. Matsue, W. Hirosue, H. Tokutake, S. Sunada, and A. Ohkura, "Navigation of Small and Lightweight Helicopter," *Transactions of the Japan Society for Aeronautical and Space Sciences*, vol. 48, no. 161, pp. 177–179, 2005.

[9] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "Quadrotor Using Minimal Sensing For Autonomous Indoor Flight," in *Proc. European Micro Air Vehicle Conf. and Flight Competition*, 2007.

[10] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Towards Autonomous Indoor Micro VTOL," *Autonomous Robots*, vol. 18, pp. 171–183, 2005.

[11] G. Buskey, J. Roberts, P. Corke, and G. Wyeth, "Helicopter automation using a low-cost sensing system," *Computing Control Engineering Journal*, vol. 15, no. 2, pp. 8–9, 2004.

[12] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 21–28, 2010.

[13] S. Hrabar and G. Sukhatme, "Vision-based navigation through urban canyons," *Journal of Field Robotics*, vol. 26, no. 5, pp. 431–452, 2009.

[14] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and Monocular Vision Based Control for MAVs in Unknown In- and Outdoor Environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 3056–3063, 2011.

[15] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments," *International Aerial Robotics Competition*, pp. 582–586, 2009.

[16] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010.

[17] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 khz," in *Proc. IEEE Intl. Conf. on Robotics and Automation*, pp. 361–366, 2007.

[18] "ROS: an open-source Robot Operating System," in *Proc. of ICRA Workshop on Open Source Software*, 2009.

[19] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[20] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2010.

[21] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.