2014 International Conference on Unmanned Aircraft Systems (ICUAS)
May 27-30, 2014. Orlando, FL, USA

# Unmanned Ground and Aerial Vehicles in Extended Range Indoor and Outdoor Missions

Francesco Cocchioni[1], Valerio Pierfelice, Alessandro Benini, Adriano Mancini,
Emanuele Frontoni, Primo Zingaretti, Gianluca Ippoliti and Sauro Longhi

*Abstract*— **The capability to instantiate a cooperation among heterogeneous agents is a fundamental feature in mobile robotics. In this paper we focus on the interaction between Unmanned Ground Vehicle (UGV) and Unmanned Aerial Vehicle (UAV) to extend the endurance of UAV, thanks to a novel landing/recharging platform. The UGV acts as a docking station and hosts the UAV during the indoor/outdoor transition and vice-versa. We designed a platform and a robust landing target to automate the fast recharge of UAV. The synchronization and coordination of cooperation is managed by a Ground Control Station (GCS) developed using a versatile software toolchain based on the integration of Stateflow, auto-generation of C-code and ROS. All the software components of UAV, UGV and GCS have been developed using ROS. The obtained results show that the UAV is able to land over the UGV with high accuracy (<5cm for both x and y axis) thanks to a visual position estimation algorithm, also in presence of wind (with gust up to 20-25km/h), recharging its batteries in a short time to extend its endurance.**

## I. Introduction

Autonomous systems represent a promising evolving area, in particular with respect to research in artificial and embedded systems. In the European industry, many solutions of autonomous systems focused on solving a very specific problem are available, but the majority of these solutions, however, is proprietary. While maintaining the property of a particular technology ensures a position in the market, on the other hand this becomes very costly especially in the long term. Due to the increase in number of proprietary solutions, the robotic industry presents a high degree of fragmentation, which cause a slowdown in the development of new systems, especially nowadays, where the number of applications of robotic systems is increasing even more. Most of the issues that must be addressed in the development of autonomous systems are common to all areas of applications: these problems include, for example, the localization in indoor/outdoor environments, the ability to avoid obstacles, the ability to make simple decisions based on the occurrence of certain events, and so on.

### A. Motivations and Research Objectives of R3-COP Project

In order to overcome or at least alleviate these challenges, the R3-COP project [1] plans to progress the current state-of-the-art of autonomous systems in two main directions: Technology and Methodology. With respect to technology, R3-COP will develop a new fault tolerant high performance embedded hardware platform based on a multi-core architecture taking into account functional and non-functional requirements of autonomous systems. In addition, R3-COP will provide innovative system components for robust perception of the environment, including sensor fusion, for reasoning and reliable action control, and for communication and cooperation among autonomous vehicles. With respect to methodology, R3-COP will address the first challenge by establishing a methodology-based development framework for autonomous systems. This will be built upon an extensible knowledge base comprising all state-of-the-art information about the enabling technologies (hardware, algorithms, etc.) for robust autonomous systems. This will include a reference framework for reasoning.

The application of this methodology will be supported by a tool platform, which will cover design, implementation, and verification/testing [2], [3]. This will make possible efficient development of reference platforms for various robotic application domains as well as dedicated solutions, with strong means for avoiding mistakes due to experience-based methodology behind [4]. Further, testing tools will support automatic test case generation for cooperating autonomous systems with complex behaviour using multidimensional and multimodal sensing. As a consequence, this platform tool will both help to reduce cost and improve dependability. Aiming to support the development of cross-domain reference platforms, long-term aspects like maintainability or reuse will also be positively influenced. The above mentioned achievements have been applied in a series of ground-based, airborne and underwater demonstrators. This paper focuses on the airborne demonstrator where two unmanned systems, a quadrotor and a four wheeled robot, cooperate in an indoor and outdoor mission.

### B. Context of this Work

The work carried out in this paper aims to overcome some of the problems addressed by the R3-COP project, with particular focus on the localization and navigation of autonomous vehicles and the capability to extend the endurance of an unmanned aerial vehicle. The obtained results have been tested and validated in the final airborne demonstrator of the R3-COP project.

In particular, the airborne demonstrator in the R3-COP project is oriented to demonstrate some examples of cooperation between the UAV and UGV during a building inspection. A more detailed description of UGV and UAV can be found in Section II and Section III. The battery recharging system,

*All authors are with Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Ancona, ITALY.

[1]Contact author: francesco.cocchioni@gmail.com

978-1-4799-2376-2/14/$31.00 ©2014 IEEE

besides being an interesting cooperating task, provides an efficient way to extent UAV endurance, and consequently its mission. A landing platform installed on the unmanned ground vehicle hosts the UAV after the landing, recharging its batteries. The platform described in the next session is simple and does not provide a swap of battery which requires complex mechanisms [5], [6].

The main aim of this paper is to present a platform installed on a UGV to recharge a battery of a UAV extending the endurance; the UAV and the UGV cooperate to accomplish an indoor/outdoor mission. The overall architecture presented in this paper is based on ROS and auto-generated code. The idea to automate the recharge of an unmanned system is not recent and for ground robots is well established (e.g., cleaning robots). The problem is still open for unmanned aerial systems especially for fixed-wing ones that cannot hover and is known as Automated Aerial Refueling (AAR) mostly Air-to-Air [7].

The main problem when dealing with AAR in the case of small rotary wing UAVs is the choice of the strategy: replace or recharge. In the first case the replace strategy allows a fast hot-swap but it requires complex mechatronic devices [5], [8], [9]. The recharging strategy is characterized by longer time to recharge battery (time depends on the maximum charging rate of battery pack). In our case, a series of electrical contacts ensures the continuity between the battery pack and the charger.

Other approaches are based on inductive power transfer but the efficiency is still low [10].

Our platform is designed to provide a passive recharge mechanism that allows to recharge the battery within 20 minutes (no full charge). We also developed a robust marker, which is installed on the landing platform allowing a precise and accurate estimation of UAV pose and attitude. A marker enables a safe landing of the UAV over the UGV also in GPS denied environment, as in [11]–[13].

The paper is organized as follows. In section II a description of the UGV is presented. Section III provides a description of UAV, while section IV describes the GCS. Section V gives details about the system architecture. Section VI outlines conclusions and future works.

## II. Unmanned Ground Vehicle

The adopted UGV is based on a customization of the well known robotic platform P3-AT. In Fig. 1 the UGV is shown. It is equipped with the following sensors/devices:

- Radio-modem Aerocomm AC4868-250
- Landing Platform for UAV Battery Recharge
- Thermal Camera FLIR Navigator
- Inertial Measurement Unit Microstrain 3DM-GX1
- 2D Laser Scanner SICK LMS200
- Li-Po Battery Charger Groupner Ultramat 18
- Geiger Sensor North Optic J305$\beta$

### A. Localization

The UGV is equipped with the AC4868-250M radio-modem produced by Aerocomm (now Laird technologies). The UGV communicates using the 868MHz radio link with a total station to get its updated position (expressed as North - East - Up). The total station (Topcon QS5) is able to track a prism (see Fig. 2) installed on the landing/recharging platform of the UGV. If the speed is low or zero the UGV requires to the total station the new estimation of position.

The system requires a start-up calibration procedure in order to calculate the roto-translation transformation from the local reference frame of the total station to the absolute reference frame. The calibration procedure consists of calculating the position of two reference points (the absolute position is well known) setting the zero (East direction) of the total station on the first reference point. The required time to perform the calibration is typically 10-15 minutes.

The total station is required in order to precisely and accurately calculate the position of UGV in the outdoor area. A wide set of tests using a differential GPS receiver has been also conducted on the same area, but the presence of dense trees/canopy and metallic facades degrade the quality of GPS positioning. The outdoor area represents an ”urban canyon”,

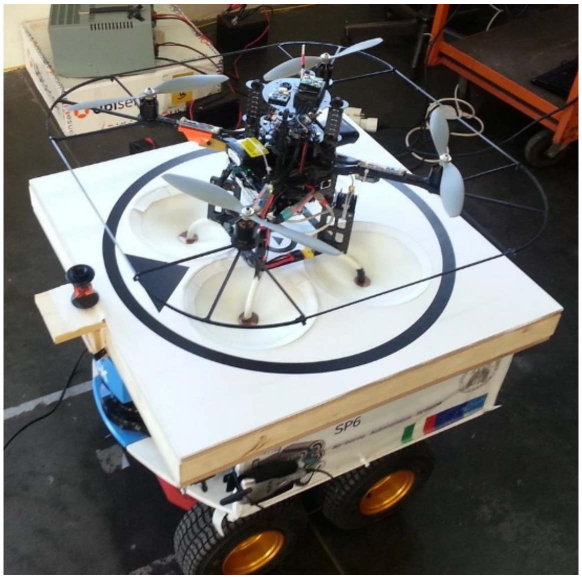

Fig. 1. Customized UGV platform with the UAV on the landing platform

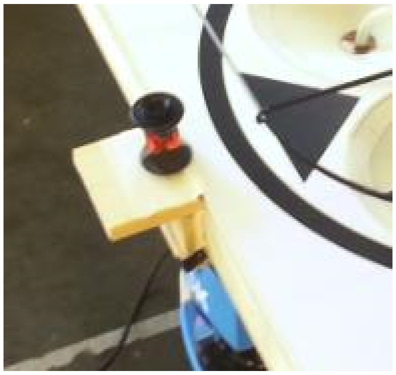

Fig. 2. Details of the installed reflector to track the position of UGV in outdoor GPS-denied environment

a GPS-denied environment. The localization and navigation algorithms are derived from the works of [14]–[16].

### B. Landing Platform

The landing platform provides both a secure place for landing, and the mechanism to recharge UAV battery using electrical contacts. The UAV will not be able to land on the desired ideal position considering the error of the estimated position and the control action. To avoid the use of large exposed electrical surface, that compensate those errors, a passive centering system brings the UAV in a fixed position where the electrical contact (small in size) are located.

The landing surface is composed of four top-down hollow cones, with their centers located at the vertices of a square of the same size of the UAV base. The radius of each cone base is upper bounded to half the value of the UAV base square side. Larger radius implies larger admissible errors which the system can compensate, either along X-Y axis and with respect to a fixed heading. The height of the cones defines, with a fixed base radius, the slope of the internal sliding surface. Electrical contacts are placed at the bottom of two adjacent cones, avoiding any possibility of accidentally short circuit or polarity inversion. Fig. 3 shows a 3D model of our landing platform.

On the UAV the electrical contacts are installed at the bottom of specific feet, designed to enter inside the landing surface (cone) and slip on it. Since the Asctec Pelican is not equipped with them, we designed a feet extension which can be installed on the Asctec Pelican base without any structural change; for this purpose we used a 3D object printer (rapid prototyping). These feet also extend the Pelican base square, allowing the use of cones with a larger radius. With these feet, the Asctec Pelican base creates a square of 20cm side; Table I shows, for different error along x and y axis, the maximum error in degree that the centering system (with cone of radius 10cm) is able to compensate with respect to one of the four principal heading (0, 90, 180, 270 degree).

### C. Recharging Interface

The recharging interface has been introduced to enable the fast re-charge of batteries at 3C rate. As discussed in the previous subsections, the landing platform hosts four surfaces. The adopted strategy is to enable the recharge without taking into account the balancing of Lipo batteries (3s3p - 6000mAh). This choice is safe if the number of consecutive recharges is low, to avoid drift of the cell's voltages.

A possible improvement of the designed platform is to add additional points of contact enabling the balancing. However, the balancing of cells during fast recharge (considering charging rate and time) is not so critical. The charger is powered with a dedicated battery and it is not necessary to use an inverter. It is possible to directly power the charger with a +12VDC input. A specific circuit allows to opto-isolate the charger from the main ground of the PC installed on-board the UGV, using a classical opto-isolator (the 4N37). When the signal from the parallel port is low the charger is disabled; it is enabled by sending a proper set of activation pulses from the parallel port. Fig. 4 shows a detail of the landing platform recharge system, with the fuse installed for safety on the drone to avoid accidental short circuit of its battery.

## III. Unmanned Aerial Vehicle

The UAV used for this work is a quadrotor manufactured by Ascending Technology: the Asctec Pelican. The sophisticated design of the drone allows to install different kind of sensors and processing unit, up to a maximum suggested payload of 650g. It is equipped with:

- Processing Unit Asctec Mastermind
- Thermocamera Flir Quark 336

TABLE I
Maximum landing error along X and Y axis for a landing platform with cone of 10cm radius

| X and Y Axis Error | Maximum Rotational Error |
|---|---|
| 0cm | 40° |
| 1cm | 33° |
| 2cm | 27° |
| 3cm | 22° |
| 4cm | 15° |
| 5cm | 10° |
| 6cm | 4° |

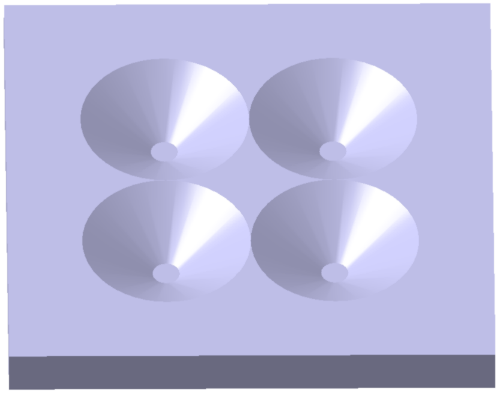

Fig. 3. 3D model of the landing platform developed.

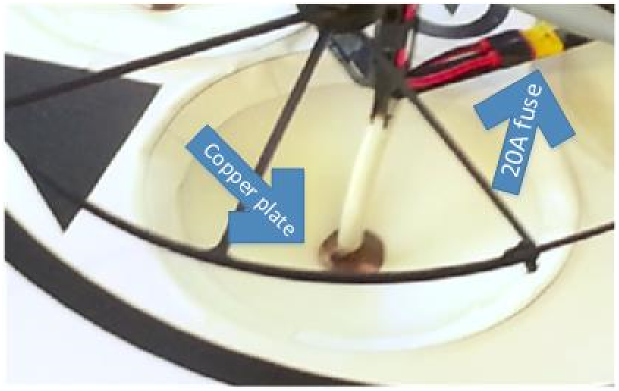

Fig. 4. Details of the landing platform recharge system.

- Global Positioning System
- Inertial Measurement Unit
- Visible Camera Matrix Vision BlueFox MLC200wC
- Ultrasonic Sensor Maxbotix MB1320

The Flir Quark 336 thermal camera (with 13mm lens) video stream is sent to the ground control station via a dedicated video-link 1.2/1.3GHz. The thermal image is processed by the ground control station to identify anomalous hot-spots in the area under inspection. In Fig. 5 an example of thermal image acquired by the UAV themocamera is shown.

In this section the developed vision system for the UAV is presented. It estimates the position of the drone with respect to a visual target installed on the landing platform, and allow the navigation of the drone in indoor and outdoor environments thanks to an optical flow algorithm which estimates the position of the quadrotor without the use of a priori know visual landmark.

### A. *Landing Target*

The choice of the landing target is fundamental to achieve a safe and robust landing. The landing target was designed to fulfil three requirements: it must fit the planar space left by the four cone on the landing surface, it must be easy to recognise from different altitudes, and it must provide the 6 DOF of the drone.

These requirements were fulfilled using rings and upside down triangle. The rings provide the estimation of the 3D position, roll and pitch angle of camera while the triangle is necessary to estimate the yaw. To make the target easily visible from different altitudes we duplicated the same design adding a larger ring and triangle. when the UAV flies at very low altitude to perform the landing, only the inner ring/triangle pair is visible by the camera (equipped with a 2.8mm lens), while at higher altitude the outer pair provide more accuracy due to its larger dimension. Fig. 6 shows the final developed landing platform with the target on its top. The smaller and larger rings/triangle pairs are, respectively, placed in the inner and outer planar surface left by the four cone. The triangles are drawn upside down to best fit the space left on the outer surface maximizing its dimensions (the inner triangle is upside down for symmetry).

### B. *Pose Estimation from Visual Target*

The designed landing platform sets the following affordable constraint: the drone should be able to land with a translation error with respect to the center of the landing platform which is $\leq$ 5cm along both x and y axis, and a maximum rotation error of 10 degree. To compensate the fast dynamics of the UAV also providing an accurate landing, the navigation control must rely on accurate and fast measurements of drone position with respect to the landing platform.

The measurements have to be available in every flight scenario, indoor or outdoor, without a priori structured environment. For these reasons, a fast and accurate as possible dedicated vision algorithm has been developed to compute the UAV position, with respect to the visual target installed on the landing platform.

The developed algorithm is written in C++ as ROS package using OpenCV library and it processes the images acquired by the Matrix Vision BlueFox MLC200wC camera (configured to acquire mono8 image at 90Hz with automatic exposure time calculation) which is installed on the lower side of the drone facing down towards the ground. In order to achieve maximum performance every image is processed considering the minimum set of operations, using efficient implementation for each ones.

The gray-scale image is binarized with a specifically written adaptive threshold algorithm, based on the one developed by Wellner [17]; with our optimization the algorithm is able

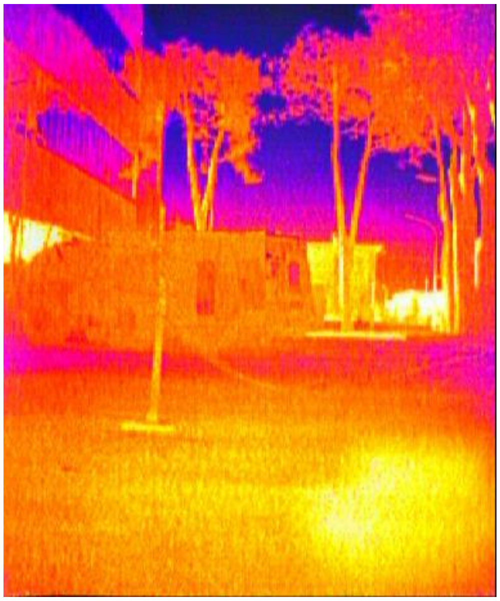

Fig. 5. Image acquired from the thermocamera installed on the UAV.

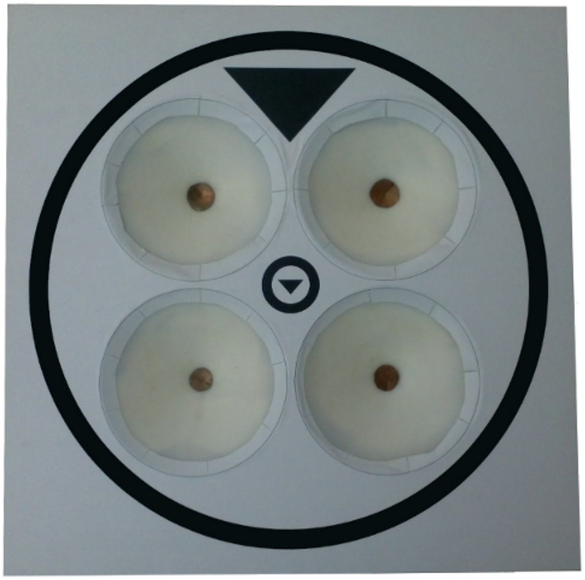

Fig. 6. Landing target on its final design.

to process a 752x480 pixel image in 2.156ms, on the onboard computer (Asctec Mastermind with a Intel Core2Duo SL9400 1.86 GHz), providing good binarization results with the same set of configuration parameters for image taken at different heights, even in presence of different light and shadow conditions (contrary to the OpenCV adaptive threshold algorithm).

The binarized image is then processed using Suzuki algorithm [18], searching for contours. All the contours are classified with a tree hierarchy, which is used to find the target and its different elements; the algorithm is executed on average in 2.402ms. Considering the geometric design of the target, most of contours can be ignored, like the ones which have not both a parent and a child contour, or the ones whose area is too small. With a series of different filtering, and no more time consuming operation (thanks to contours hierarchy), the contours associated with the target, and the respective ellipse which interpolate them are extrapolated; the contours associated to the largest visible triangle is also found.

The overall algorithm is able to process an image with an average time of 5.5ms in the worst situation, with the camera at such altitude that both the entire target and part of the background are in its field of view. Our algorithm is able to process all the images provided by the camera at its full speed (90Hz); it could be able to process even further image per seconds, if a faster image acquisition system is available. Fig. 7 shows the robustness of our algorithm to external light/shadow disturbance.

The 6DOFs of the camera with respect to the landing platform are estimated from the largest visible circle and triangle. The estimation of drone pose (with exception of yaw angle) is based on the work proposed by Chen [19]; the estimation of yaw angle is performed using the larger visible triangle (finding the angle generated by its unique top vertex).

The accuracy of the estimated pose has been verified with a series of different static tests. The results are summarized in Table II and III.

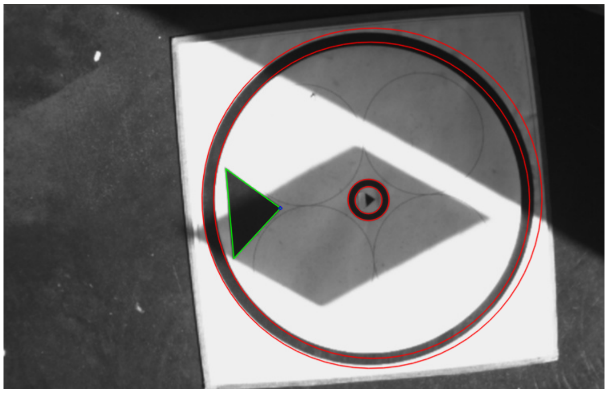

Fig. 7. Landing target detection at high altitude with external light/shadow disturbance.

### C. Position Estimation for Navigation

The estimated position of the drone with respect to the landing platform can not be used for its the navigation, since the landing vision algorithm requires that the visual target is always in the field of view of the camera. To estimate the position of the drone without using the landing target an additional vision algorithm has been developed. This algorithm is based on the optical flow to estimate the odometry of the drone. A set of features are tracked between temporally close images to estimate the movement of camera. This vision algorithm has been developed in C++ using OpenCV library sharing the same ROS application as the landing vision algorithm. The resulting moving vector defines in the image plane the movement sensed by the camera. The movement must be converted from pixel to a real distance unit, using the pin hole model and a measurement of camera distance from the ground (in our case provided by the ultrasonic sensor).

Since the camera is fixed on the drone frame, an oscillation of the drone during its normal operation produces in the image plane a movement vector which is not associated to a real camera translation movement. For this reason, the vector provided by the filtering algorithm must be processed removing the component introduced by the rotational movement of the camera. This is performed using the solution proposed by Rondon [20]. It was not possible to calculate the accuracy for the position estimated by the navigation vision algorithm, since there was no ground truth localization system. A work around test has been used to validate the position estimated.

The drone has been commanded to autonomously fly in a closed path, using the control algorithm and the high-level logic discussed in the following of this paper, starting over the landing target which defines the origin of our reference system. Once the drone returns in sight of the landing target a comparison between the accurate value provide by the landing vision algorithm and the one provided by the navigation vision algorithm is compared, calculating the error for x and y axis. Table IV shows the results for three different autonomous flights done on three paths of different length,

TABLE II
AVERAGE POSITION ESTIMATION ERROR, USING THE LANDING TARGET AT DIFFERENT HEIGHT.

| Distance | X Axis Error | Y Axis Error | Z Axis Error |
|---|---|---|---|
| 25cm | 0.13cm | 0.08cm | 0.35cm |
| 50cm | 0.42cm | 0.30cm | 1.94cm |
| 100cm | 1.32cm | 1.49cm | 3.74cm |
| 200cm | 1.81cm | 1.98cm | 5.86cm |

TABLE III
AVERAGE ANGLE ESTIMATION ERROR USING THE LANDING TARGET.

| Angle | Error |
|---|---|
| Roll | 2.66° |
| Pitch | 1.45° |
| Yaw | 1.27° |

which are performed, respectively from the shortest to the longest, indoor, outdoor and both indoor and outdoor.

In Table V the evaluation of endurance performed on several real runs is shown. The success rate of autonomous landing and correct positioning of UAV over the copper contact is $> 90\%$.

## IV. Ground Control Station

The ground control station is responsible for the execution of a building inspection mission involving the coordination of a couple of autonomous vehicles, consisting of the UAV and the UGV. The mission plan is assigned by an operator and consists of two sets of objectives, one per vehicle.

Generally, the plan involves the surveillance of an indoor and outdoor area, including the transition from one area to another one. During the surveillance phase, the ground control station is responsible for the synchronous coordination of the autonomous systems. It prevents the takeoff of the UAV from the recharge platform while the UGV is moving, or the UAV to reach the landing point before the UGV reached the rendezvous point.

These conditions are defined in the mission plan by the operator and are generally expressed as ”Wait for UGV (or UAV) to reach objective X before moving to objective Y”, as shown in Fig. 8. The transition phase requires the coordination of UAV and UGV to sequentially land the UAV on the recharge platform, activate the recharge station, move the UGV along the transition path, disable the recharge and takeoff the UAV when the UGV reaches the first objective of the new area. Once landed, the UAV enters in a sleep mode, turning off the motors, and must be awakened once the UGV ends the transition phase.

In order to extend the endurance of the mission, the ground control station must interrupt the execution of the mission allowing the UAV to recharge the batteries on the UGV recharge platform. This action is performed by stopping the UAV and UGV from reaching their current goals and redirecting the UAV to UGV’s current position. Once landed, the ground control station requests to the UGV the activation of the recharge system. When the recharge is complete the ground control station deactivates the recharge system and resumes the mission as planned.

TABLE IV
Position estimation error using the navigation vision algorithm, after closed path of different length.

| Path Lenght | X Axis Error | Y Axis Error |
|---|---|---|
| 13m | 0.49cm | 6.07cm |
| 37m | 38.8cm | 21.6cm |
| 63m | 136cm | 57.3cm |

TABLE V
Evaluation of UAV endurance

| Phase | Time [minutes] |
|---|---|
| Mission | 13 |
| Landing | 1 |
| Reserve | 1 |
| Charging (20A) | 20 |

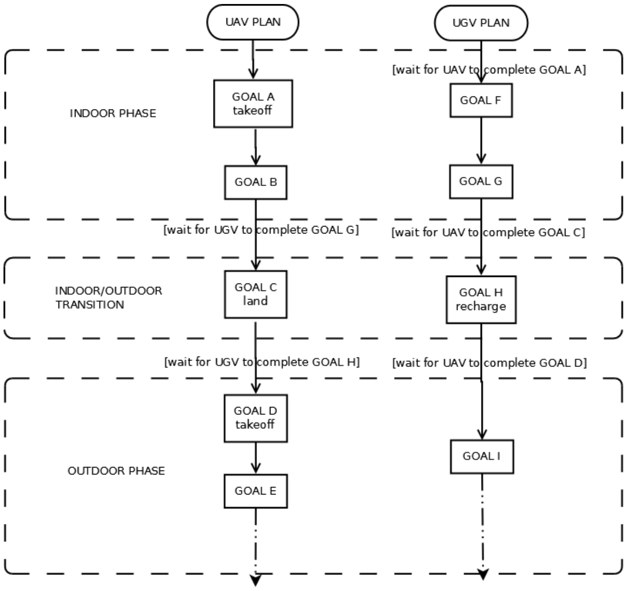
UAV PLAN
UGV PLAN
INDOOR PHASE
[wait for UAV to complete GOAL A]
GOAL A takeoff
GOAL F
GOAL B
GOAL G
[wait for UGV to complete GOAL G]
[wait for UAV to complete GOAL C]
INDOOR/OUTDOOR TRANSITION
GOAL C land
GOAL H recharge
[wait for UGV to complete GOAL H]
[wait for UAV to complete GOAL D]
GOAL D takeoff
OUTDOOR PHASE
GOAL E
GOAL I

Fig. 8. Example: definition of a mission plan

## V. System Overview

### A. Architecture

To achieve the goal of a loosely coupled architecture, the high-level control system has been divided into a number of modules that communicate with each other over a 802.11n WiFi Network, as shown in Fig. 9 .

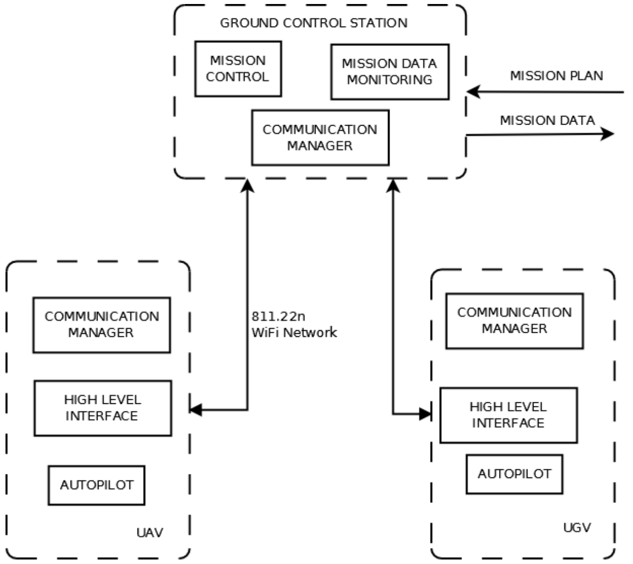

Fig. 9. Conceptual design of system architecture

In order to successfully interact with the ground control station, each vehicle runs three modules on-board:

- ”Autopilot” module
- ”High Level Interface” module

- ”Communication Manager” module

*a) The Autopilot:* this module acts as a waypoint server to handle each ground control station request; it notifies to the ground control station the status of each goal received and fulfills it by implementing an interface with the vehicle’s on-board position controller. It is responsible for the autonomous navigation of the UAV and the UGV.

*b) The High Level Interface:* this module implements an interface to control and observe each vehicle’s sub component: UAV motors, UGV recharge system, battery status, and so on. At the same time, it prevents the misuse of critical safety compliants components (e.g. it must prevent the sent to UAV of the turn off command while it is flying).

*c) The Communication Manager:* this module implements the zero-configuration protocol [21] by which each module publishes itself over the local network as a zero-configuration service, enabling the others to automatically discover it, and vice versa. Then it is possible to implement a mechanism to detect network failures and grant a robust communication among GCS, UAV and UGV. The communication manager is also responsible for the rerouting of all public signals over the local network; it redirects all the control signals from the ground control station to UAV/UGV and, vice versa, redirects the interface implemented by the ”Autopilot” and ”High level interface” to the ground control station.

### B. *Software*

In order to provide a portability of our code avoiding potential (binary) driver issues, we installed Ubuntu Linux 12.04 on each on-board computer and on the ground control station, which makes tedious cross-compiling unnecessary. Since we are running a couple of different subsystems that need to communicate with each other, we use the ROS [22] framework as a middleware. The necessity to maximize the re-use of software avoiding the code re-writing is a critical aspect especially for heterogeneous and cooperating robots.

Inter process/robot communication (IPRC) architecture serves as a framework to implement TCP communication between remote nodes, by exploiting each local ROS master node interface in order to register local topics and nodes over the local network on remote master nodes, as shown in Fig. 10. This way, a failure in the network connection does not stop the local nodes standard execution, because they depend only on the local master node.

Control logic for the ”Autopilot”, ”High Level Interface” and ”Mission Control” modules is designed as a set of finite state automa, implemented with Matlab Stateflow/Simulink software, as shown in Fig. 13. The auto-generation code is performed by the Embedded Coder of Matlab/Simulink, which automatically translates the stateflow/Simulink diagram into executable ANSI C code.

A stub code has been also developed to link the C struct generated from the Embedded Coder to ROS messages, in order to run each control module as a standalone ROS node, as shown in Fig. 11 and 12.

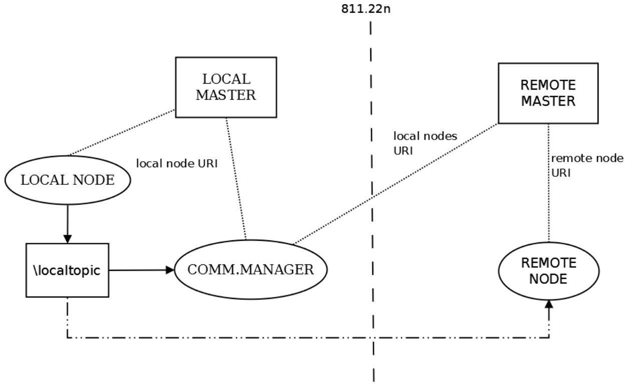

Fig. 10. Example of ROS *Master* interface exploitation to achieve loosely coupled communication between nodes distributed over the local network

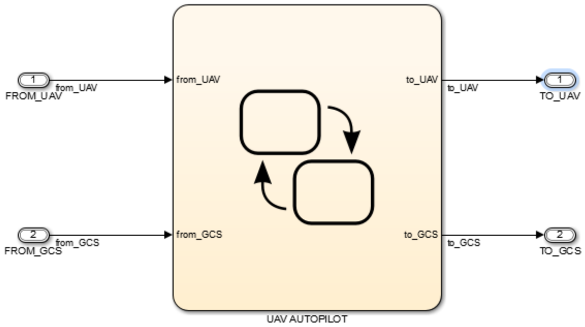

Fig. 11. Simulink implementation of the UAV Autopilot and High level interface Stateflow diagrams

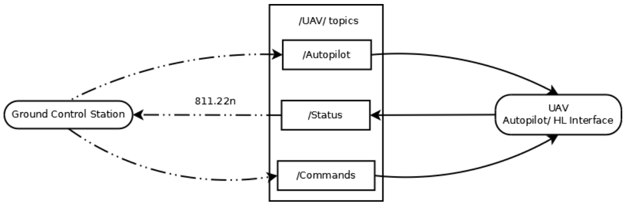

Fig. 12. ROS implementation of the UAV Autopilot and Interface modules with signals stubbed as topic messages

## VI. Conclusions and Future Works

In this paper a heterogeneous robotic system formed by UAV and UGV coordinated by a centralized GCS has been presented. The paper presented a working approach to recharge batteries of an unmanned aerial vehicles to extend its endurance. The UGV acts as a docking station that hosts the UAV after landing. The set-up has been successfully tested on real mixed indoor/outdoor scenario in the context of the EU Project R3-COP.

The architecture is based on the use of ROS that has been adopted for UAV, UGV and GCS. The auto-generation of C code stubbed into ROS has been developed in order to generate safe and verified/validated code.

The UAV is able to quickly detect the landing target even in presence of critical environmental conditions (e.g., shadows and changes in the illumination) thanks to the developed thresholding algorithm, and is able to navigate both in indoor and outdoor areas using a position estimation based on optical flow.

The video in [23] shows our UAV and UGV implementing a cooperation from the take-off to the landing.

Future works will be steered to integrate additional sensors for the indoor positioning in order to improve the performance of visual odometry with particular reference to the Ultra Wide Band (UWB) Real Time Location systems [24].

## ACKNOWLEDGEMENT

This work has been developed in the framework of the EU ARTEMIS-JU R3-COP (Resilient Reasoning Robotic Cooperating Systems) Project. The authors would like to thank Riccardo Minutolo, Francesco Barcio, Francesco Monai and Mauro Montanari at Thales Italia S.p.A. for their support.

## REFERENCES

[1] "R3COP," Apr. 2014. [Online]. Available: http://www.r3-cop.eu

[2] A. Benini, A. Mancini, R. Minutolo, S. Longhi, and M. Montanari, "A modular framework for fast prototyping of cooperative unmanned aerial vehicle," *Journal of Intelligent and Robotic Systems*, vol. 65, no. 1-4, pp. 507–520, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10846-011-9577-1

[3] A. Benini, A. Mancini, E. Frontoni, P. Zingaretti, and S. Longhi, "A simulation framework for coalition formation of unmanned aerial vehicles," in *Control Automation (MED), 2011 19th Mediterranean Conference on*, 2011, pp. 406–411.

[4] E. Schoitsch, W. Herzner, C. Alonso-Montes, P. Chmelar, and L. Dalgaard, "Towards composable robotics: The r3-cop knowledge-base driven technology platform," in *Computer Safety, Reliability, and Security*, ser. Lecture Notes in Computer Science, F. Ortmeier and P. Daniel, Eds. Springer Berlin Heidelberg, 2012, vol. 7613, pp. 427–435. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33675-1_40

[5] K. Swieringa, C. Hanson, J. Richardson, J. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 3335–3340.

[6] T. Toksoz, J. Redding, M. Michini, B. Michini, J. P. How, M. Vavrina, and J. Vian, "Automated Battery Swap and Recharge to Enable Persistent UAV Missions," in *AIAA Infotech@Aerospace Conference*, March 2011, (AIAA-2011-1405).

[7] C. Martinez, T. Richardson, and P. Campoy, "Towards autonomous air-to-air refuelling for uavs using visual information," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 5756–5762.

[8] N. Ure, G. Chowdhary, T. Toksoz, J. How, M. Vavrina, and J. Vian, "An automated battery management system to enable persistent missions with multiple aerial vehicles," vol. PP, no. 99, 2014, pp. 1–12.

[9] M. Saska, T. Krajnik, and L. Pfeucil, "Cooperative uuav-ugv autonomous indoor surveillance," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*, March 2012, pp. 1–6.

[10] S. Jung, T. Lee, T. Mina, and K. Ariyur, "Inductive or magnetic recharging for small uavs," in *SAE 2012 Aerospace Electronics and Avionics Systems Conference*, October 2012.

[11] S. Lange, N. Sunderhauf, and P. Protzel, "A vision based onboard approach for landing and position control of an autonomous multirotor uav in gps-denied environments," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, June 2009, pp. 1–6.

[12] N. Frietsch, O. Meister, C. Schlaile, and G. Trommer, "Teaming of an ugv with a vtol-uav in urban environments," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*, May 2008, pp. 1278–1285.

[13] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi, "A single-camera feature-based vision system for helicopter autonomous landing," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, June 2009, pp. 1–6.

[14] L. Armesto, G. Ippoliti, S. Longhi, and J. Tornero, "Probabilistic self-localization and mapping - an asynchronous multirate approach," *IEEE Robotics and Automation Magazine*, vol. 15, no. 2, pp. 77–88, 2008.

[15] G. Ippoliti, L. Jetto, A. La Manna, and S. Longhi, "Improving the robustness properties of robot localization procedures with respect to environment features uncertainties," vol. 2005, 2005, pp. 1451–1458.

[16] C. Fulgenzi, G. Ippoliti, and S. Longhi, "Experimental validation of fastslam algorithm integrated with a linear features based map," *Mechatronics*, vol. 19, no. 5, pp. 609–616, 2009.

[17] P. D. Wellner, "Adaptive Thresholding For The DigitalDesk," Tech. Rep., 1993. [Online]. Available: http://xrce.fr/content/download/16283/117540/file/EPC-1993-110.pdf

[18] S. Satoshi and A. Keiichi, "Topological Structural Analysis Of Digitized Binary Images By Border Following," 1985. [Online]. Available: http://dblp.uni-trier.de/db/journals/cvgip/cvgip30.html#SuzukiA85

[19] C. Qian, W. Haiyuan, and W. Toshikazu, "Camera calibration with two arbitrary coplanar circles," 2004. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24672-5_41

[20] E. Rondon, I. Fantoni-Coichot, A. Sanchez, and G. Sanahuja, "Optical Flow-Based Controller For Reactive And Relative Navigation Dedicated To A Four Rotor Rotorcraft," 2009. [Online]. Available: http://dx.doi.org/10.1109/IROS.2009.5354483

[21] E. Guttman, "Autoconfiguration for ip networking: Enabling local communication," *IEEE Internet Computing 5 (3)*, vol. 5, no. 3, pp. 81–86, 2001. [Online]. Available: http://dx.doi.org/10.1109/4236.935181

[22] "ROS," Apr. 2014. [Online]. Available: http://wiki.ros.org

[23] "Video Of The Developed System," Apr. 2014. [Online]. Available: http://www.youtube.com/watch?v=Q5WWU84TGaI

[24] A. Benini, A. Mancini, and S. Longhi, "An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15.4a wireless sensor network," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 461–476, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10846-012-9742-1

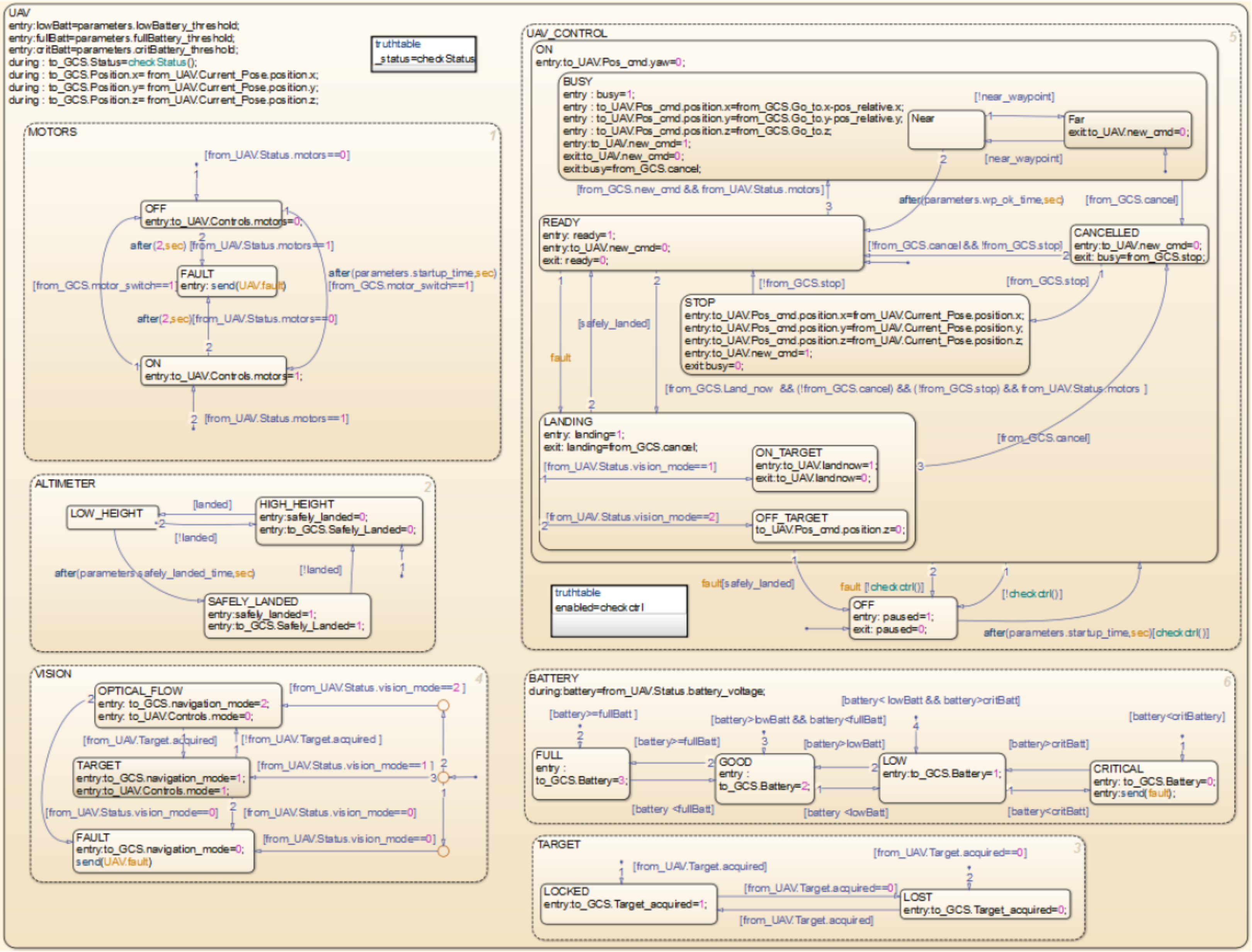
UAV
entry: lowBatt=parameters.lowBattery_threshold;
entry: fullBatt=parameters.fullBattery_threshold;
entry: critBatt=parameters.critBattery_threshold;
during : to_GCS.Status=checkStatus();
during : to_GCS.Position.x= from_UAV.Current_Pose.position.x;
during : to_GCS.Position.y= from_UAV.Current_Pose.position.y;
during : to_GCS.Position.z= from_UAV.Current_Pose.position.z;
truthtable
_status=checkStatus
MOTORS
[from_UAV.Status.motors==0]
OFF
entry:to_UAV.Controls.motors=0;
after(2,sec) [from_UAV.Status.motors==1]
FAULT
entry: send(UAV.fault)
[from_GCS.motor_switch==1]
after(parameters.startup_time,sec)
[from_GCS.motor_switch==1]
after(2,sec)[from_UAV.Status.motors==0]
ON
entry:to_UAV.Controls.motors=1;
[from_UAV.Status.motors==1]
ALTIMETER
LOW_HEIGHT
[landed]
[!landed]
HIGH_HEIGHT
entry:safely_landed=0;
entry:to_GCS.Safely_Landed=0;
after(parameters.safely_landed_time,sec)
[!landed]
SAFELY_LANDED
entry:safely_landed=1;
entry:to_GCS.Safely_Landed=1;
VISION
OPTICAL_FLOW
entry: to_GCS.navigation_mode=2;
entry: to_UAV.Controls.mode=0;
[from_UAV.Status.vision_mode==2 ]
[from_UAV.Target.acquired]
[!from_UAV.Target.acquired ]
TARGET
entry:to_GCS.navigation_mode=1;
entry:to_UAV.Controls.mode=1;
[from_UAV.Status.vision_mode==1 ]
[from_UAV.Status.vision_mode==0]
[from_UAV.Status.vision_mode==0]
FAULT
entry:to_GCS.navigation_mode=0;
send(UAV.fault)
[from_UAV.Status.vision_mode==0]
UAV_CONTROL
ON
entry:to_UAV.Pos_cmd.yaw=0;
BUSY
entry : busy=1;
entry : to_UAV.Pos_cmd.position.x=from_GCS.Go_to.x-pos_relative.x;
entry : to_UAV.Pos_cmd.position.y=from_GCS.Go_to.y-pos_relative.y;
entry : to_UAV.Pos_cmd.position.z=from_GCS.Go_to.z;
entry:to_UAV.new_cmd=1;
exit:to_UAV.new_cmd=0;
exit:busy=from_GCS.cancel;
Near
[!near_waypoint]
Far
exit:to_UAV.new_cmd=0;
[near_waypoint]
[from_GCS.new_cmd && from_UAV.Status.motors]
after(parameters.wp_ok_time,sec)
[from_GCS.cancel]
READY
entry: ready=1;
entry:to_UAV.new_cmd=0;
exit: ready=0;
[from_GCS.cancel && !from_GCS.stop]
CANCELLED
entry:to_UAV.new_cmd=0;
exit: busy=from_GCS.stop;
[!from_GCS.stop]
[from_GCS.stop]
[safely_landed]
fault
STOP
entry:to_UAV.Pos_cmd.position.x=from_UAV.Current_Pose.position.x;
entry:to_UAV.Pos_cmd.position.y=from_UAV.Current_Pose.position.y;
entry:to_UAV.Pos_cmd.position.z=from_UAV.Current_Pose.position.z;
entry:to_UAV.new_cmd=1;
exit:busy=0;
[from_GCS.Land_now && (!from_GCS.cancel) && (!from_GCS.stop) && from_UAV.Status.motors ]
LANDING
entry: landing=1;
exit: landing=from_GCS.cancel;
[from_UAV.Status.vision_mode==1]
ON_TARGET
entry:to_UAV.landnow=1;
exit:to_UAV.landnow=0;
[from_GCS.cancel]
[from_UAV.Status.vision_mode==2]
OFF_TARGET
to_UAV.Pos_cmd.position.z=0;
truthtable
enabled=checkctrl
fault[safely_landed]
fault [!checkctrl()]
[!checkctrl()]
OFF
entry: paused=1;
exit: paused=0;
after(parameters.startup_time,sec)[checkctrl()]
BATTERY
during:battery=from_UAV.Status.battery_voltage;
[battery>=fullBatt ]
[battery< lowBatt && battery>critBatt]
[battery>lowBatt && battery<fullBatt]
[battery<critBattery]
[battery>=fullBatt]
[battery>lowBatt]
[battery>critBatt]
FULL
entry :
to_GCS.Battery=3;
GOOD
entry :
to_GCS.Battery=2;
LOW
entry:to_GCS.Battery=1;
CRITICAL
entry: to_GCS.Battery=0;
entry:send(fault);
[battery <fullBatt]
[battery <lowBatt]
[battery<critBatt]
TARGET
[from_UAV.Target.acquired]
[from_UAV.Target.acquired==0]
LOCKED
entry:to_GCS.Target_acquired=1;
[from_UAV.Target.acquired==0]
LOST
entry:to_GCS.Target_acquired=0;
[from_UAV.Target.acquired]

Fig. 13. Matlab Stateflow diagram implementation of the UAV Autopilot and High level Interface control logic