CAPÍTULO 14

FIELD TESTS WITH AN AERIAL-GROUND CONVOY SYSTEM FOR COLLABORATIVE TASKS

J. VALENTE, A. BARRIENTOS, A. MARTINEZ, C. FIEDERLING

Robotics and Cybernetics Group, Centro de Automática y Robótica UPM-CSIC; joao.valente@etsii.upm.es

This chapter presents the design, implementation and field experiments of a convoy between an aerial and a terrestrial robot. The convoy strategy proposed is indeed very simple and based in a PD control law. We introduce the robots *Pinky* and *Gaia*, robots which have been part of the FRACTAL fleet, the general system set up is also addressed, such as the ground station workloads and the middleware architecture. Finally, comprehensive experimental results shown herein, demonstrate the good performance and usability of the system in multi-robot behavioral research.

1 Introduction

A Multi-robot system (MRS) has many advantages over a Single-robot system (SRS), robot systems with robots endowed with different abilities and different degrees of autonomy play an important role in coordination and cooperation tasks. The usability of a MRS can range from military to civil applications, such as, De-mining (MacArthur, 2005), Agriculture (Elfes, 1999), Forestry, Urban Search and Rescue (Blumenthal, 2008), Inspection and monitoring (Chaimowicz, 2004).

In order to carry out complex tasks in a feasible way is important that the robots collaborate, coordinate and share their own resources between them. In the context of the FRACTAL project we have developed a general framework to handle this issues. Moreover, we set up a simple convoy strategy between an aerial and a ground robot to proof the feasibility and stability of the architecture, likewise to apply in scenarios where an aerial and ground robots can help each other to achieve a common goal.

The set up of a physical robot fleet is not an easy task, when working with those platforms, dozens hours in outdoor environments are needed until you achieve the minimum desired results, moreover this often means handling difficult climate conditions, like strong winds, cold, rain and even snow, preventing the rapid research development. This chapter present our effort to set up a convoy of terrestrial and ground robots, and is organized as follow: Section 2, reviews some of the researches comprising the usage of ground and aerial robots, Section 3, gives an overall description the system, namely the platforms employed, the software architecture and the base station workloads, Section 4, described the methods implemented and the experimental set up, Section 5, report the results achieved, and Section 6 present the conclusions and research directions of our system.

2 Related work

A ground and aerial robot arrangement have been used oftentimes in research, due to an augmented range of applications. Many fruitful results have been presented so far, however we will just briefly review some of them.

In (Vidal, 2009), the authors present the implementation of a distributed hierarchical architecture to handle the coordination and control of a multi-robot team, which main aim was the development of a pursuit-evasion game. In order to accomplish this task two ground robots and one aerial robot were used. Elfes et al. reported the AURORA's project status in (Elfes, 1999), a project which aim is the development of an air-ground robotic system to be applied to biodiversity and agricultural applications. The overall system is described, being emphasized the software architecture, airship control and cooperative visual control. In (Phan, 2008), is presented a three layer centralized and hierarchical framework for the cooperative control of a team of heterogeneous robots. The top level of this architecture handles mission planning, task allocation, and have been designed in the context of wild fire detection an fighting. The system involves three UAVs and two UGVs. In the GRASP laboratory from the University of Pennsylvania experimental results with a blimp and ground vehicles were achieved. This work addresses multi-robot localization in highly cluttered urban environments (Chaimowicz, 2004). The authors in (MacArthur, 2005) reported an UGV/UAV collaboration platform for the simulation of mines disposal. In this work, the UAV mapped the field with the purpose to identify targets (i.e. mines) locations and then dispatch the UGV to those positions.

3 System overview

In this section we will give an overview of the platforms used in those experiments, the system software architecture and the main components that made up the ground station.

3.1 Robot platforms

The platforms depicted in Fig. 1 belong to the FRACTAL fleet and have been acquired commercially with the purpose to be applied in cooperative and collaborative tasks. With them we have developed architectures, algorithms and hardware components, specifically designed to address the problems of cooperation, coordination, command and control, in different scenarios. In Fig. 1 are depicted the aerial and ground robots.

Aerial robot

The aerial robot called *Gaia* is a mini vertical takeoff and landing helicopter denoted by quad-rotor.

Gaia provides a 3-axis gyroscopes, 3-axis accelerometers, 3-axis magnetometer, GPS, and in addition a barometer. Moreover, has also a servo mounted in the main frame, which enables a commercial zoom digital camera to be tilted up to 100 degrees. This instrumentation assures the platform stabilization, manual control, and way-point navigation, as well to dispatch the commands from a software mission planner set in the ground station, which enables the operation of the robot to a higher level.

Ground robot

Pinky is an UGV based in the Pioneer3-AT with high re-configurability and the possibility to embedded a on-board computational unit. The layout of the robot includes the development of a low-level interface for angular rate sensors, ultrasound, Joystick, gyroscope and external manipulators, as well as sensory fusion algorithms based on a Kalman filter to provide an accurate state estimation of the vehicle, and without the need to resort to differential corrections.

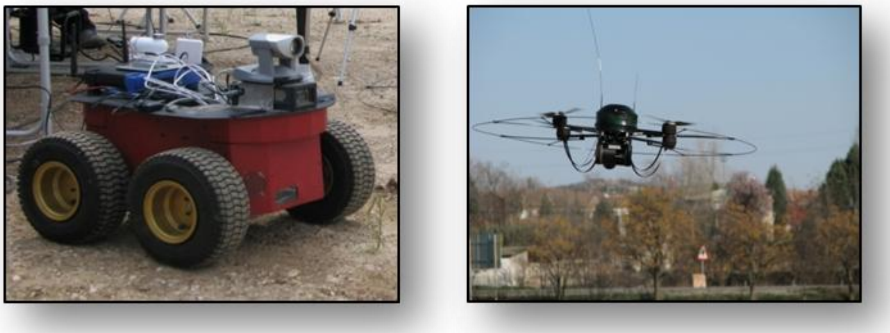


Fig. 1. From the left to the right, the ground robot *Pinky* and the aerial robot *Gaia* respectively

3.2 Ground station

The main components of the ground station are the communications workload and the Graphical User Interface (GUI). The ground station computer receives and sends data to *Gaia* through a RS-232 down-link and up-link, which communicates with the quad-rotor through a Radio Control (RC) system that has available 9 channels and operates at 35 MHz. Additionally, a video transmission downlink link at 688 MHz, is also set to receive video images acquired through the digital camera.

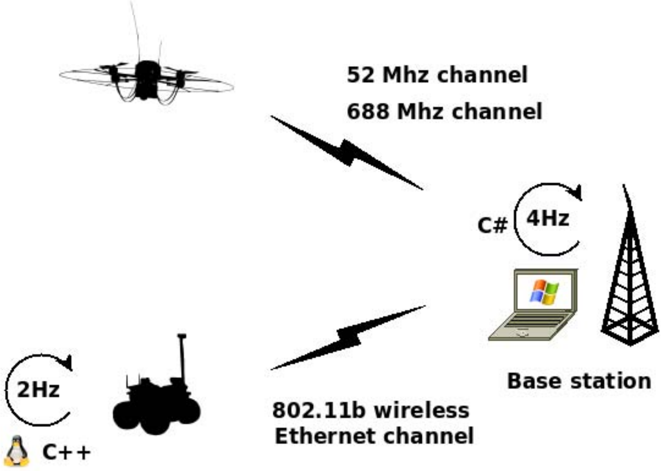


Fig. 2. The communication workflow

As part of the conditioning of an aerial robot system, is to develop a Graphical User Interface (GUI) that provides the operator with basic information about the flight and mission status. Since it is more operator than user oriented, we denote it as Graphical Operator Interface (GOI).

The interface display all navigation data, vehicle status, video image from the on-board camera, providing also, in-flight image and video recording and storing, and general data logging, likewise a TCP/IP client connection enables the real time in-flight data transmission to a server.

The main feature of this interface is the embedded real time web mapping API therein. Some examples of web mapping software are Open Layers, Google earth, Open Street Map, Marble, etc.

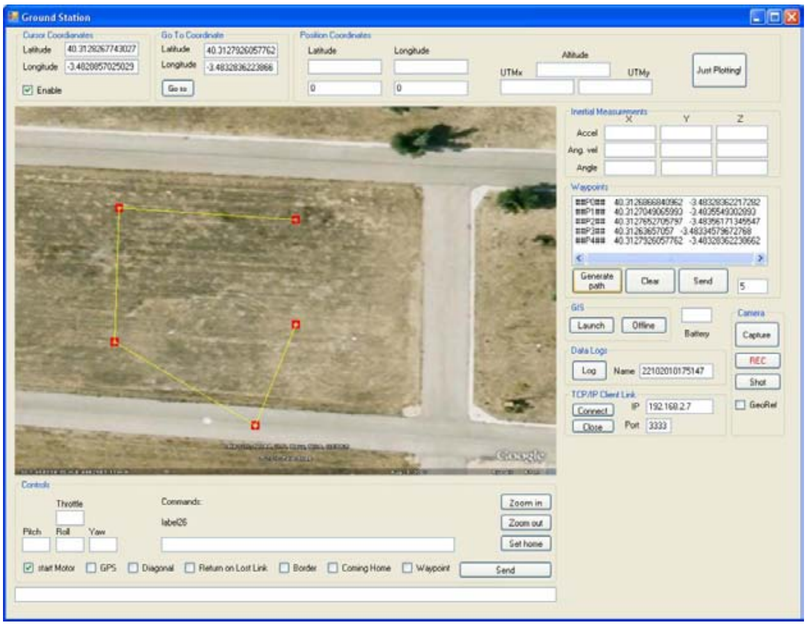
The goal of this design is to improve the operator vigilance level by stimulating his/her situational awareness, the main idea is not only to feedback the operator with the robot sensory information but also provide an approximate real map of the robot environment. Although we have not made Human-Robot interaction (HRI) experiments with this interface design we believe that this approach improves significantly the operator responsiveness.

This feature also plays an important role in the tele-operation of mini aerial robots over large environments, where the operator field-of-view is affected by the robot distance.

In tele-operation mode an operator can steer the quad-rotor through a RC command or through a joystick. With the RC command, odd from steering the platform, the operator can tilt the digital camera servo, take image shots and zoom the camera.

Moreover, is also possible to define a way-point route by directly click in the embedded web mapping panel, as shown in Fig. 3. When steering the quad-rotor with the joystick, if the operator needs to have a larger image from the digital camera, it can switch view between the web mapping panel and the video image panel (i.e. by pressing a joystick button), likewise the opposite (see Fig. 4).

This design criterion lack in some points, namely the absence of an Internet connection, or an Internet connection with a low bandwidth, the minimum network speed requirement for our API is 128 Kbits/sec and the recommended is 768 Kbits/sec. We have handled this issue by making available an off-line modality, where the operator can download and set a static image from the robot workspace with geographic information (see Fig. 5).



Cursor Coordinates: Latitude 40.3128267743027; Longitude -3.4828857025029; ☑ Enable

Go To Coordinate: Latitude 40.3127926057762; Longitude -3.4832836223866

Position Coordinates: Latitude 0; Longitude 0; Altitude; UTMx; UTMy

Inertial Measurements: Accel; Ang. vel; Angle

```
##P0##   40.3126866840962   -3.48328362217282
##P1##   40.3127049065993   -3.4835549302093
##P2##   40.3127652705797   -3.48356171349547
##P3##   40.31263657057   -3.48334579672768
##P4##   40.3127926057762   -3.48328362230662
```

5

Name 22102010175147

IP 192.168.2.7

Port 3333

☐ GeoRef

Throttle; Pitch; Roll; Yaw

Commands:

label26

☑ start Motor ☐ GPS ☐ Diagonal ☐ Return on Lost Link ☐ Border ☐ Coming Home ☐ Waypoint

Fig. 3. The GOI in the stand-by modality where is depicted a prior planned way-point trajectory

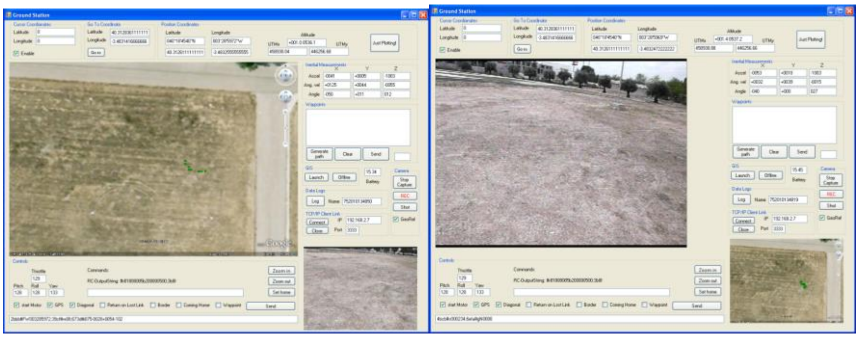
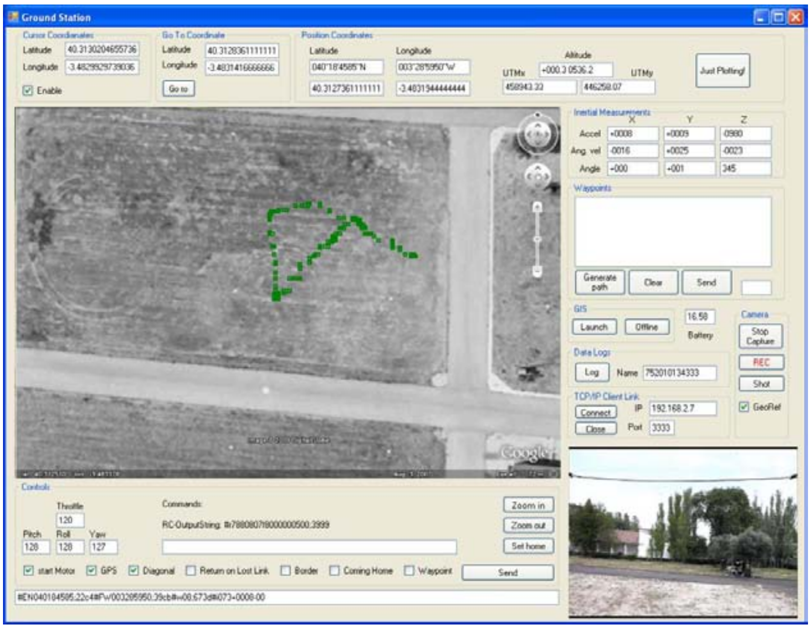


Fig. 4. From the left to the right, snapshots from the GOI in the web mapping mode and the camera mode



Cursor Coordinates: Latitude 40.3130204655736, Longitude -3.4829929739036, ☑ Enable

Go To Coordinate: Latitude 40.3128361111111, Longitude -3.4831416666666

Position Coordinates: Latitude 040°18'4585"N, 40.3127361111111; Longitude 003°28'5950"W, -3.4831944444444; Altitude +000.3 0536.2; UTMx 458943.22; UTMy 446258.07

Inertial Measurements:

	X	Y	Z
Accel	+0008	+0009	0980
Ang. vel	0016	+0025	0023
Angle	+000	+001	345

Battery 16.58

Data Logs: Name 752010134333

TCP/IP Client Link: IP 192.168.2.7, Port 3333

☑ GeoRef

Controls: Throttle 120, Pitch 128, Roll 128, Yaw 127

Commands:

RC-OutputString: #r78808071800000000500.3999

☑ start Motor ☑ GPS ☑ Diagonal ☐ Return on Lost Link ☐ Border ☐ Coming Home ☐ Waypoint

#EN040184585.22c4#FW003285950.39cb#v00.673d#073+0008-00

Fig. 5. The GOI in the off-line modality, this is without internet connection

Finally, this design is focused in outdoor applications, being difficult, to extended it applicability to indoor environments since the robot position is mapped through the GPS coordinates.

3.3 Fractal architecture

The first step to the development of a MRS is the design of a reliable architecture to handle the perception, localization, guidance, navigation and control modules, namely is from vital importance the low-level communication that ensures an efficient data exchange between each individual robot and/or the team and the ground station.

A fleet of robots has many different software requirements such as the control of each robot at the lowest level, the communication to the base station, reactive control, autonomous behaviors, such as detection and obstacle avoidance and other measures for self-protection, mission planner, etc. Moreover, the architecture should make available a data flow control to detect faults in the transmission and have flexibility for incorporating new functions without changing the entire system software.

The overall software architecture developed can be seen in Fig. 6. In this particular case, the software for the FRACTAL fleet was developed to meet the following main requirements:

- **Multi-platform**
  - Must be able to handle the work-flow of several robots in real time
- **Robots Control**
  - This task is the lowest level and provides the link between sensors and actuators of each robot.
- **Communication link**
  - It is responsible for receiving orders of the base station and communicate the status of the robot and the entire fleet.
- **Reactive Control**
  - This instance enables the reaction to unpredictable events, such as, unexpected obstacles or failures in communication. These autonomous functions are the main security of fleet.

The communication server is the central module of the fleet, its main task is to send commands and read the robots status, as also to promote the collection, and sharing of those variables through the connected peers. For security reasons, this server is designed to work inside the robot leader because it increases the service availability to the subordinate robots in case of connection failure with the base station. Furthermore, the software can also be executed from any computer within the network, enabling the use of the fleet without a leader.

The communication system uses the "Network Address Translation" (NAT) function, which is a IP packets address modification process. In the FRACTAL network, the NAT function is used to generate a subnet **FRACTAL BASE**. The leader is equipped with two wireless networks devices to generate a subnet to connect the slave robots. The last NAT type transformation occurs in the slave robots, where it can connect its on-board system, like a camera or other Ethernet device to a wireless network.

The system described above only allows to send data from the bottom-up, to send orders from the base station to the robots, is necessary a second addressing system, which is also part of NAT and operates based on port numbers, and where a port represents a channel used by a single instance of the system. The communication philosophy in the FRACTAL system is that, each device that exchanges information with the base station has its own unique port number assigned by an algorithm that allows the identification of the robot and the service requested, in order to aid the addition of new robots slaves to the system leader.

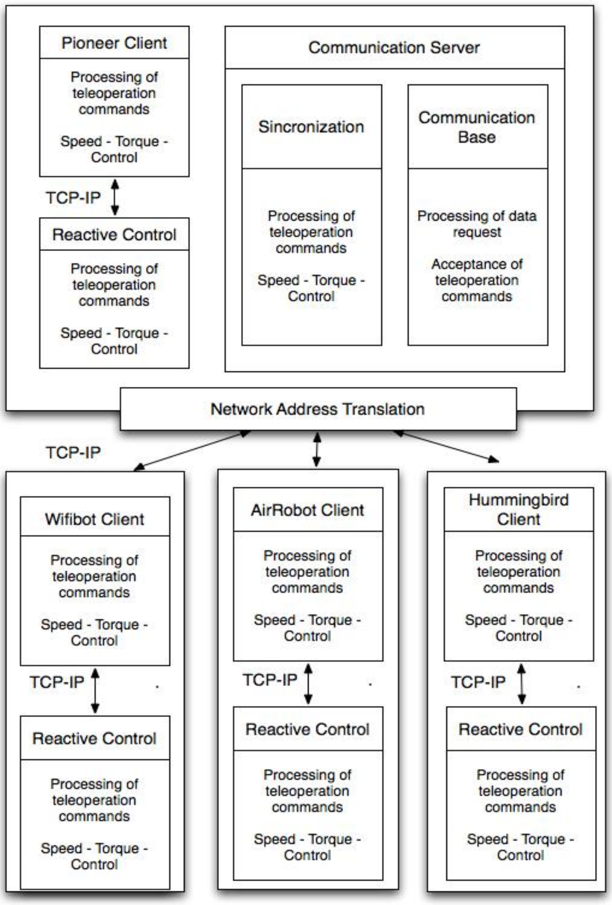


Fig. 6. The FRACTAL software architecture

Besides the above, the software architecture promote a distributed system design, including the following configurations:

1. Server, client, reactive control and command station on the same processor.
2. Server and base station running in a processor and client and reactive control in another.

3. Server, leader client, leader reactive control and distributed reactive control running in a processor, slave client running in another, and the command station in a third one.

4 The convoy strategy

The proposed approach works by using a simple PD control law which enable the ground robot to follow the trajectory of the aerial robot based in their global position raw data.
The discrete control law applied can be written as,

$$u(k) = \mathrm{B} + \alpha e(k) + \beta(\frac{e(k) - e(k-1)}{\mathrm{T}}), \tag{1}$$

where B, α, β, T stand for bias reference value, proportional gain, differential gain and time. Additionally, e is an error term, which is the difference between the set point and the process variable. An abstract schematic of the control employed is shown in Fig. 7.

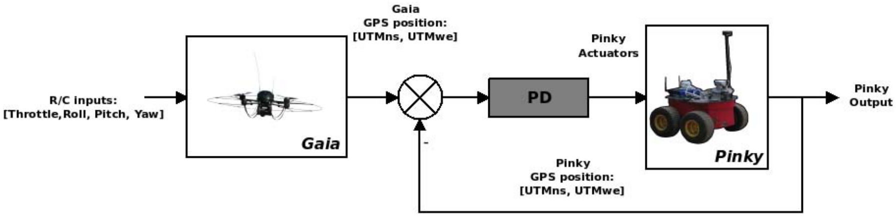


Fig. 7. An abstract control scheme from the strategy employed

The experimental set up was arranged as follow, first the robots were put together in a start position, nearby each other, in order to be easy to verify the path made by them and validate the aim of the experiment. Then, tele-operated by a skilled pilot the aerial robot took off and was settled in hovering until it position got established. Thereafter the pilot took again the manual control and performed a random trajectory. The UTM positions of the aerial robot were sent to the ground robot in real time through a TCP/IP channel.

5 Experiments and results

The experiments were conducted at the CSIC installations, on the north-west of Madrid, Spain, with geographic coordinates 40º18'47.43" N and 3º29'01.02" W.

Although the aerial robot has the capability to perform autonomous way-point navigation, for a matter of safety, it was not possible to perform an autonomous trajectory based in way-points, mainly due to strong winds during the day. For that motive the quad-rotor was tele-operated during this experiment. In Fig. 8 is shown the 2D trajectory and in Fig. 9 the 3D trajectory (i.e. the height of the *Gaia* is also depicted), were is clear the path traversed simultaneously by the robots.

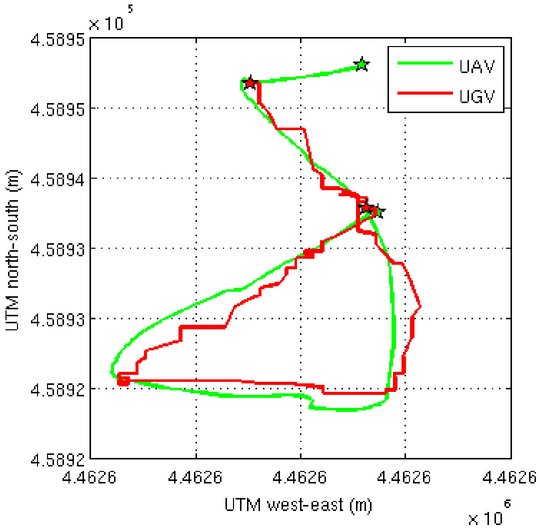


Fig. 8. Convoy 2D trajectory

Fig. 10 shows the robots positions versus time, where we can see that both positions plots, from the UGV and UAV have similar patterns, the following lasted five minutes, and the data samples were sent to the *Pinky* at 20 Hz. In the development of outdoor multi-robot systems, position errors and delays in the transmission can affect considerably the system performance, depending of course on the application. In this way is important to maintain greater data transmission frequencies to promote accurate updates of the world coordinates.

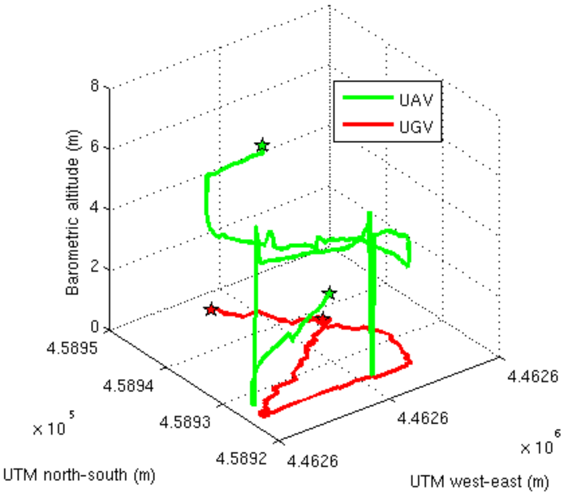


Fig. 9. Results from the Aerial and Ground robots positions in 3D during the experiments

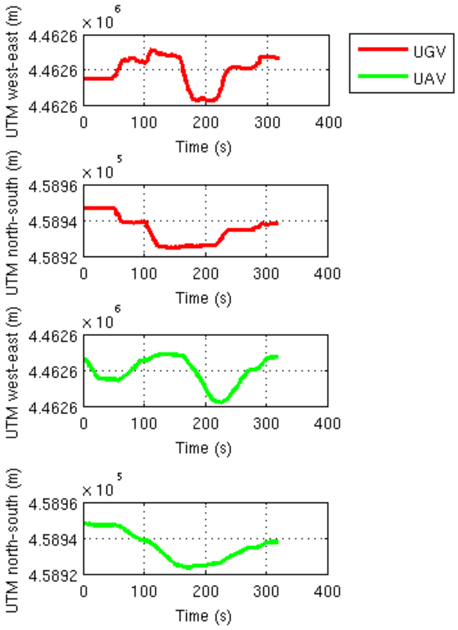


Fig. 10. Robots positions versus time

The graph shown in Fig. 11 shows an average error of 2.13 meters in the UTM North-South coordinates, and an average error of 2.52 meters in the UTM West-East coordinates. This error is perfectly normal and is due the own nature of the global position system. Moreover, the data packages haven't been processed, instead we are using directly the raw coordinates through the UGV following control.

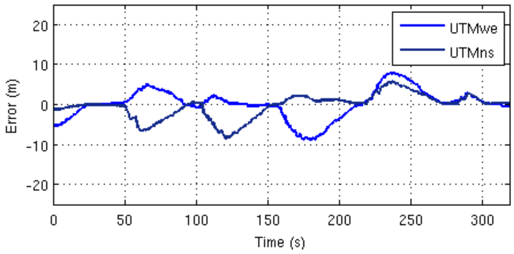
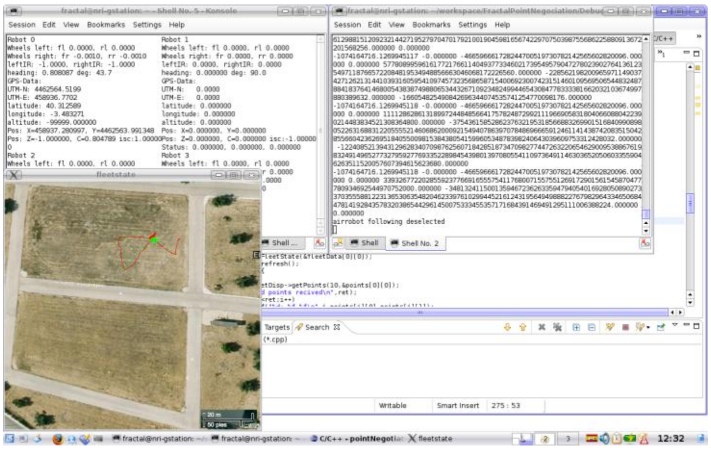


Fig. 11. Position errors

In Fig. 13 are shown a sequence of video samples from the experiment and a wind speed measurement of the same day, although the screenshot have little resolution the video from the experiment will be available at the research group home page[1]. Another screenshot was taken in the same day from the *Pinky* operator computer where is also depicted both robot paths (see Fig. 12).


fractal@nri-gstation: ~ - Shell No. 5 - Konsole
Session Edit View Bookmarks Settings Help
fractal@nri-gstation: ~/workspace/FractalPointNegociation/Debu
fleetstate
Shell ...
Shell
Shell No. 2
C/C++
Targets
Search
Writable
Smart Insert
275 : 53
fractal@nri-gstation: ~
C/C++ - pointNegoti
fleetstate
12:32

```
Robot 0                                          Robot 1
Wheels left: fl 0.0000, rl 0.0000                Wheels left: fl 0.0000, rl 0.0000
Wheels right: fr -0.0010, rr -0.0010             Wheels right: fr 0.0000, rr 0.0000
leftIR: -1.0000, rightIR: -1.0000                leftIR: 0.0000, rightIR: 0.0000
heading: 0.808087 deg: 43.7                      heading: 0.000000 deg: 90.0
GPS-Data:                                        GPS-Data:
UTM-N: 4462564.5199                              UTM-N:    0.0000
UTM-E: 458936.7702                               UTM-E:    0.0000
latitude: 40.312589                              latitude: 0.000000
longitude: -3.483271                             longitude: 0.000000
altitude: -99999.000000                          altitude: 0.000000
Pos: X=458937.280997, Y=4462563.991348           Pos: X=0.000000, Y=0.000000
Pos: Z=-1.000000, C=0.804789 isc:1.00000Pos: Z=0.000000, C=0.000000 isc:-1.00000
0                                                Status: 0.000000, 0.000000, 0.000000
Robot 2                                          Robot 3
Wheels left: fl 0.0000, rl 0.0000                Wheels left: fl 0.0000, rl 0.0000
```

```
612988151209232144271952797047017921001904598165674229707503987556862258809136 72
201568256.000000 0.000000
-1074164716.1269945117 -0.000000 -4665966617282447005197307821425656028200 96.000
000 0.000000 5778089959616177217661140493773346021739549579047278023902764136123
549711876657220848195349488566630460681722265 60.000000 -2285621982009659711490 37
427126213144103931605954109745732356865871540069230074231514601095695065448324 87
884183764146800543838749880653443267109234824994465430847783338166203210367499 7
880389632.000000 -1660548254908426963440745357412547700981 76.000000
-1074164716.1269945118 -0.000000 -4665966617282447005197307821425656028200 96.000
000 0.000000 1111286286131899724484856641757824872992111966905831804066088042239
0214483834521308364800.000000 -375436158528623763219531856688326990151684099089 8
052263168831220555521460686200092154940786397078486966659124611414387420835150 42
855660423626951840550098153843805415996053487839824064303960975331242803 2.000000
 -1224085213943129628340709876256071842851873470982774472632206546290095388676 19
832491496527732795927769335228984543980139708055411097364911463036520506033559 04
626351152005760739461562368 0.000000
-1074164716.1269945118 -0.000000 -4665966617282447005197307821425656028200 96.000
000 0.000000 3393267722028559237766916555754117680071557551269172901561545870477
7809346925449707520 00.000000 -34813241150013594672362633594794054016928050890273
370355588122313653063548204623397610299445216124319564949888227679829643346506 84
478141928435783203865442961450075333455357171684391469491295111006388224.000000
0.000000
airrobot following deselected
```

```
FleetState(&fleetData[0][0]);
refresh();
{

etDisp->getPoints(10,&points[0][0]);
d points recived\n",ret);
<ret;i++)
```

Fig. 12. Snapshot from the ground robot operator computer after the completion of the experiment

[1] http://robcib.etsii.upm.es/

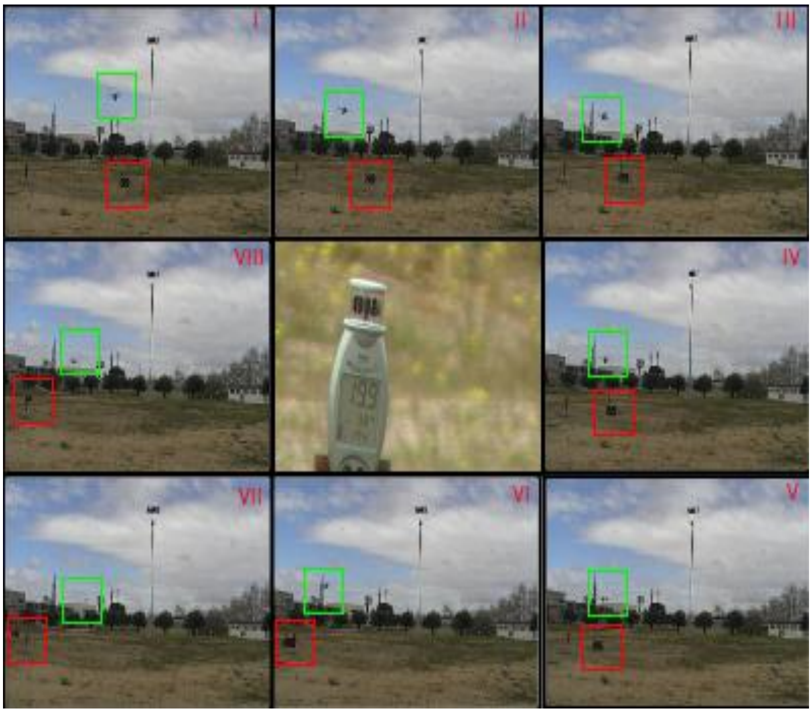


Fig. 13. *Pinky* (red square) following the *Gaia* (green square)

6 Conclusions and Future work

In this chapter we have shown some experiments with a convoy made up by an heterogeneous team of robots, one aerial and another terrestrial, robots which have contribute to the development of the FRACTAL project. We have also made a brief introduction to the FRACTAL software architecture, focusing in their structure and usability. A simple convoy scheme between *Gaia* and *Pinky* was arranged with the purpose to prove the reliability of the framework proposed and to study the development of a behavior where an aerial robot and ground robot could collaborate/coordinate among them. The results achieved in the experiments demonstrate that the architecture has a good performance in multi-robot coordination tasks, and that using just raw data from a GPS can be inefficient for certain tasks demands.

Further research with our system will consist in the improvement of the position errors of each robot, likewise the development of a common global reference frame. Our strategy will consist in the implementation of a differential GPS, and in the development of an embedding visual system.

Acknowledgements

This work was supported by the Robotics and Cybernetics Group at Technique University of Madrid (Spain), and funded under the project FRACTAL: Fleet of cooperative terrestrial and aerial robots, sponsored by the Spain Ministry of Education and Science (DPI 2006-03444), by the ROBOCITY 2030 project, sponsored by the Community of Madrid (S-0505/DPI/ 000235). The authors would like also to thanks to Angela Ribeiro for the availability during the experiments.

References

Blumenthal, S., Holz, D., Linder, T., Molitor, P., Surmann, H., and Tretyakov, V. 2008. Teleoperated visual inspection and surveillance with unmanned ground and aerial vehicles. In Remote Engineering and Virtual Instrumentation.

Chaimowicz, L., Grocholsky, B., Keller, J. F., Kumar, V., and Taylor, C. J. 2004. Experiments in multirobot air-ground coordination. In in Proc. of the 2004 International Conference on Robotics and Automation, pp. 4053-4058.

Elfes, A., Bergerman, M., Carvalho, J., de Paiva, E., Ramos, J., and Bueno, S. 1999. Air-ground robotic ensembles for cooperative applications: Concepts and preliminary results. In 2nd International Conference on Field and service Robotics, Pittsburgh, Pa (USA), pp. 75-80.

MacArthur, E. Z., MacArthur, D. and Crane, C. 2005. Use of cooperative unmanned air and ground vehicles for detection and disposal of mines. SPIE.

Phan, C. and Liu, H. 2008. A cooperative uav/ugv platform for wild fire detection and fighting. In System Simulation and Scientific Computing, pp. 494-498.

Vidal, R., Rashid, S., Sharp, C., Shakernia, O., Kim, J., and Sastry, S. 2001. Pursuit-evasion games with unmanned ground and aerial vehicles. In Proc. IEEE International Conference Robotics and Automation, Vol. 3, pp. 2948-2955.