A Cloud Based Service for Management and Planning of Autonomous UAV Missions in Smart City Scenarios

A Cloud Based Service for Management and Planning of Autonomous UAV Missions in Smart City Scenarios

Gabriele Ermacora[1], Antonio Toma[1], Stefano Rosa[2], Basilio Bona[2], Marcello Chiaberge[3], Mario Silvagni[1], Marco Gaspardone[4] and Roberto Antonini[4]

[1]Politecnico di Torino, DIMEAS, Turin, Italy
[2]Politecnico di Torino, DAUIN, Turin, Italy
[3]Politecnico di Torino, DET, Turin, Italy
{gabriele.ermacora,antonio.toma,stefano.rosa,basilio.bona, marcello.chiaberge,mario.silvagni}@polito.it
[4]Telecom Italia S.p.a., Turin, Italy
{marco.gaspardone,roberto1.antonini}@polito.it

Abstract. Cloud Robotics is an emerging paradigm in which robots, seen as abstract agents, have the possibility to connect to a common network and share on a complex infrastructure the information and knowledge they gather about the physical world; or conversely consume the data collected by other agents or made available on accessible database and repositories. In this paper we propose an implementation of an emergency-management service exploiting the possibilities offered by cloud robotics in a smart city scenario. A high-level cloud-platform manages a number of unmanned aerial vehicles (quadrotor UAVs) with the goal of providing aerial support to citizens that require it via a dedicated mobile app. The UAV reaches the citizen while forwarding a real-time video streaming to a privileged user (police officer),connected to the same cloud platform, that is allowed to teleoperate it by remote.

Keywords: UAV, smart city, cloud robotics, open data, shared knowledge, navigation, path planning.

1 Introduction

At the dawn of cloud robotics [8] [9][10] many and interesting applications are possible exploiting the power of this technology [11][12]. Moreover the number of applications that see UAV as interesting devices for environment monitoring is growing. Indeed users can access to services built by fetching data from UAVs, such as telemetry or video streaming. As this paper illustrates, these services, which are part of the IT architecture, can be accessed via web or other devices, such as smartphone applications. A real security problem in a urban context is proposed as a test case in this paper; urban spaces monitored by cameras are not an efficient way to decrease crime rates since criminal events e.g., theft, robbery, rape tend to occur in unmonitored zones. Thus the aim of this test case is to apply this cloud architecture, based on ROS [1] [2], to crime prevention. In the case of aggression the user requests the emergency service from the IT architecture, by providing GPS coordinates and an identification number. The IT architecture organizes a UAV to reach him/her for offering monitoring and support. In the meantime a police officer will use the service to see the current position of the UAV, its telemetry and video streaming from its camera. When UAVs are required for search and rescue or emergency interventions:

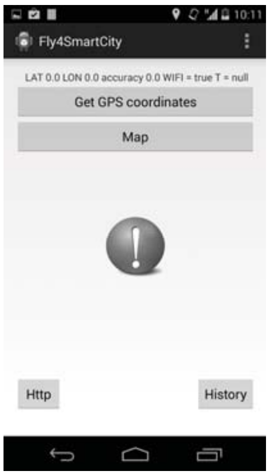
1. User sends a request to the service.
2. The request is automatically forwarded to the platform.
3. The platform transforms the request into a mission in a lower level language message.
4. Depending on the mission the message will be deployed to either a single or a swarm of UAVs

2 Interfaces

In this paper two interfaces to service are presented:

- A user-side interface which requests help and assistance from the UAV;
- A police-side interface for monitoring and management

The first interface is a mobile application running on an Android smartphone. It acquires GPS coordinates and phone ID when the user request for help. Then it sends these data (GPS coordinates and phone ID), over HTTP protocol, through a GET request to the server of the cloud platform. An example of the GET request could be: *http:/ffsc/request/?gps=45.674524,7.398732&cell=32308937642&command=takeoff*



LAT 0.0 LON 0.0 accuracy 0.0 WIFI = true T = null

Fig. 1. Android application

The second interface is addressed to the police force. The officer accesses all the information about the UAV collected by telemetry and video streaming via a web browser. In this way he/she can know the actual position of the UAV, displayed on a map embedded in the web page. Information about remaining estimated time and distance for mission accomplishment are also made available. In addition, the video streaming from the UAV camera can offer assistance to the person in emergency.

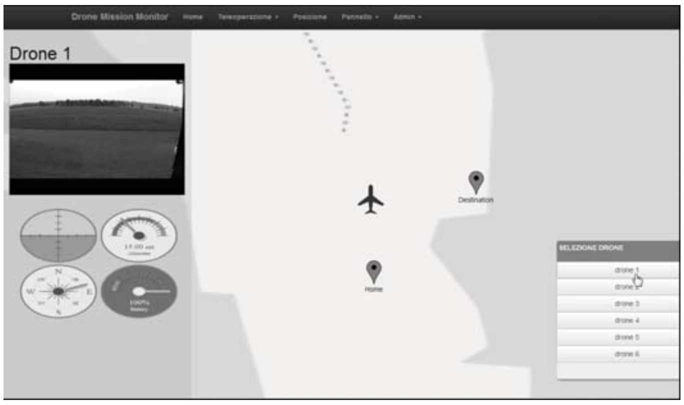


Fig. 2. Web browser GUI

3 The Cloud Platform

The cloud platform mainly consists of three layers :

- Front End which contains APIs to build new services;
- Application which contains all specific applications (the so called "remote brain"), that support APIs above;
- Adaption which contains adapters and drivers to connect different robots, and abstracts their basics functionalities to the above applications and APIs.

The platform is based on ROS framework [2], as showed in Figure 3, where gray boxes represent ROS nodes. The platform context is composed by two additional layers:

- Robots which contains all robots. They are connected to the platform through specific ROS nodes, named drivers and adapters (Adaption Layer);
- Services which contains all services exploiting APIs exposed by the platform (Front End Layer).

In order to have the cloud platform more robust and resilient we add the following managing elements:

- WatchDog System (WDS): to manage ROS nodes and represent the node status;
- Message Discovery Function (MDF): to enable or disable APIs according to ROS messages.

4 The Agents

The first validation-tests of the overall system have been conducted using a quadrotor as agent. Three different products are used in the validation of the proposed architecture:

Parrot AR.Drone: The AR.Drone [3] is a commercial low-cost quadrotor solution, fully equipped for remote control via smartphone. It features a front HD camera and the flight stability is ensured by a mother board (running a real-time linux-based operating system) and a navigation board interfaced with the on-board sensors (two cameras, ultrasonic range finders, gyroscopes and accelerometers).

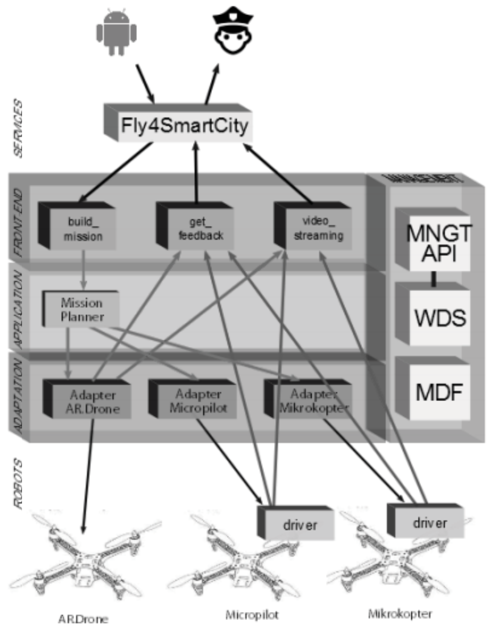


Fig. 3. The cloud platform architecture

Mikrokopter: Mikrokopter [4] is a complete auto-pilot designed for the control of generic multi-rotor platforms. It features two different boards: the Flight Control board guarantees vehicle inherent stabilization and altitude-hold function, the Navi Control board adds a set of GPS/Compass based autonomous navigation functions (waypoint navigation, come-home function, position hold mode). The Flight Controller relies on Atmel ATMEGA644 board running at 20MHz, and interfaces with the main inertial sensors (3-axis accelerometer, three gyros, one barometric sensor). Mikrokopter allows the user to take external control of the UAV (i.e., bypassing the radio controller) by means of a dedicated serial protocol.

Micropilot 2128: uPilot 2128 [5] is an auto-pilot board embedding all the peripherals needed for a stable and autonomous quad-rotor flight. This auto-pilot is specifically addressed to professional use and applications, this is reflected by its higher price and its market segment. Though Micropilot uses a completely closed-source software, it offers some tools allowing the user to write his own code. These functions come with an add-on product called "Xtender" [6]; Xtender provides a dedicated dynamic linking library that acts as a intermediate layer between the user code and the autopilot software. Using the functions encoded in the library the developer is able to get access to several low-level parameters of the auto-pilot and can modify their values. Due to Micropilot's high price and to its relatively young support to multi-copters when compared to other solutions on the market, it is not so common to find academic works that use this hardware.

Notice also that AR.Drone is a commercial ready-to-fly quadrotor while Micropilot and Mikrokopter are just two different models of autopilot electronic boards; in order to fly these require a mechanical frame, four or more motors, the same number of ESCs (Electronic Speed Controllers) and a battery. However since the ROS interface has to communicate directly with the autopilot, from a functional point of view the ROS architecture is not impacted by the general architecture of the agent, therefore this has not been described in detail.

Table 1. Quadrotors features and integration status in ROS environment

	Market	**Command**	**Telemetry link**	**Autonomous navigation**	**SDK**	**ROS support**
AR.Drone	Videogames /Hobby	Smartphone (via wifi)	Wifi (TCP/UDP packages)	☹	☺	☺
Mikrokopter	Hobby/ Photographer	Radio controller	UART (Custom Serial Protocol)	😐	☹	😐
Micropilot 2128	Professional applications	Radio controller	UART (Custom Serial Protocol)	☺	☺	☹

The three architectures offer growing functionalities, but also growing difficulties in implementation. Table 1 summarizes their main features and their integration status in ROS environment.

5 Conclusion and Future Work

In this paper we propose a test case for cloud robotics for emergency management and monitoring service. We apply emerging technologies such as web services and mobile applications to use robotics in the proposed cloud architecture. In the future work we intend to improve the cloud robotics platform enhancing the concept of robustness and resilience. In particular we intend to apply the multi-master robot concert technology [7] enabling ROS container multiplicity. We are testing the architecture in a real smart city environment with LTE connectivity. Then we want to exploit Open Data from the Internet in order to plan a flight according to obstacles present in the environment and information regarding LTE signal strength and availability. A preliminary simulation of the path planning exploiting Open data is in Figure 4.

In figure 4 there is the path computed by the platform according to LTE signal strength and obstacles present in the environment using a simple A*. The platform finds a path (in purple) .Waypoints are calculated with radiuses. Starting from the first point of the path it calculates a radius from the point to the nearest no fly zone (both physical obstacle and LTE low signal). With the calculated radius it draws a circle (in Figure 4 is in blue) that intercepts the path finding the next waypoint. This procedure continues for any waypoint.

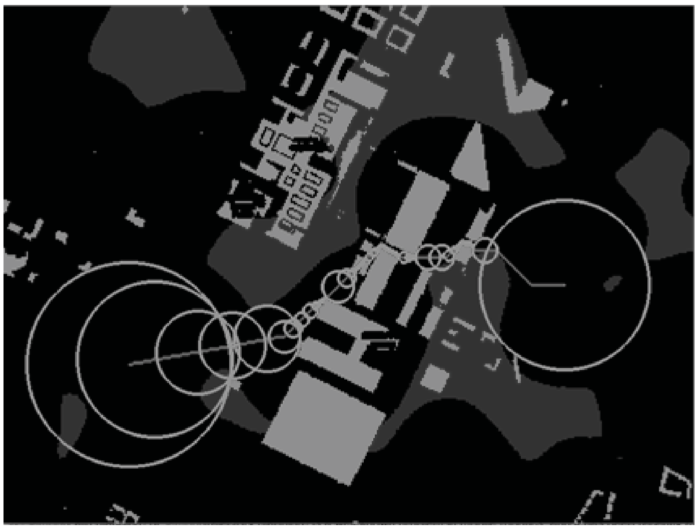


Fig. 4. Path planning simulation. LTE areas with low RSSI are shown in grey. Obstacles shown in white, free space in black. Waypoint are shown with associated radiuses in blue.

Acknowledgment. This work is in collaboration with Telecom Italia S.p.A. in the project Fly For Smart City.

References

1. Quigley, M., et al.: ROS: an open-source Robot Operating System. In: ICRA Workshop on Open Source Software 3(3(2)) (2009)
2. `http://www.ros.org`
3. `http://ardrone2.parrot.com/`
4. `http://mikrokopter.de/en/home`
5. `http://www.micropilot.com/`
6. `http://www.micropilot.com/products-xtendermp.htm`
7. `http://www.robotconcert.org/wiki/Main_Page`
8. Mell, P., Grance, T.: The NIST definition of cloud computing (draft). NIST special publication 800.145 (2011): 7
9. Goldberg, K., Kehoe, B.: Cloud Robotics and Automation: A Survey of Related Work. Electrical Engineering and Computer Sciences University of California at Berkeley
10. Sanfeliu, A., Hagita, N., Saffiotti, A.: Network robot systems. Robotics and Autonomous Systems 56(10), 793–797 (2008)
11. Chibani, A., et al.: Ubiquitous robotics: Recent challenges and future trends. Robotics and Autonomous Systems (2013)
12. Kamei, K., et al.: Cloud networked robotics. IEEE Network 26(3), 28–34 (2012)