From Simulated to Real Scenarios: A Framework for Multi-UAVs

Andrea Cesetti, Adriano Mancini, Emanuele Frontoni, Primo Zingaretti, and Sauro Longhi

Universitá Politecnica delle Marche, DIIGA, Ancona, Italy
{cesetti,mancini,frontoni,zinga}@diiga.univpm.it,
sauro.longhi@univpm.it

Abstract. In this paper a framework for simulation of Unmanned Aerial Vehicles (UAVs), oriented to rotary wings aerial vehicles, is presented. It allows UAV simulations for stand-alone agents or multi-agents exchanging data in cooperative scenarios. The framework, based on modularity and stratification in different specialized layers, allows an easy switching from simulated to real environments, thus reducing testing and debugging times. CAD modelling supports the framework mainly with respect to extraction of geometrical parameters and virtualization. Useful applications of the framework include pilot training, testing and validation of UAVs control strategies, especially in an educational context, and simulation of complex missions.

Keywords: modelling framework for robots and environments, testing and validation of robot control software, simulated sensors and actuators, UAV.

1 Introduction

Nowadays mobile robotics is going through a period of constant growth, producing tangible results in both scientific and commercial areas. However there is a significant difference between the results achieved with ground vehicles and aircrafts. *Unmanned Aerial Vehicles* (UAVs) represent a challenging research field due, on one hand, to the complexity of systems and operating environment and on the other hand to the variety of tasks they can perform. The range of aerial vehicles is ample (blimps, gliders, kites, planes, helicopters, etc.) and each one has a particularity that makes the difference in a mathematical description of physical phenomena.

Mathematical models are really complex because an aerodynamic description has to be taken into account and dynamics is also influenced by turbulence from rotors and wind. Small-scale helicopters probably represent the most difficult systems to model because of the complex nature of their dynamics. At the same time their unique manoeuvrability capabilities (including hovering, vertical take-off and landing) and multiple flight modes make them able to perform various tasks, such as surveillance, search and rescue, photogrammetry and mapping.

In many cases, complex missions can be carried out by fleets of cooperating autonomous and heterogeneous vehicles, hence interaction, cooperation and

S. Carpin et al. (Eds.): SIMPAR 2008, LNAI 5325, pp. 17–28, 2008.

supervision become the main problems. UAV application development is closely linked to the possibility of exploiting all benefits of simulation: modularity, repeatability and low cost. The risks produced from a direct use of real aircrafts are obvious. The only alternative to a powerful simulation framework could be the supervision of an expert pilot, but this solution is often quite difficult to practise. The complexity correlated to today challenges in terms of missions and tasks sets up the necessity of simulating, debugging and testing. Simulation activities are essential for testing and validation of control strategies because different methodological approaches can be easily implemented and evaluated to reduce developing times [1].

To allow an easy transfer of results from simulated to real applications is important to design a modular structure in which dedicated modules can be substituted with real devices.

In the case of ground robots a lot of simulation and test frameworks have been developed [2,3]. Player/Stage/Gazebo is actually one of the most complete framework owing to advanced features like the emulation of 2D-3D environments, the simulation of sensors (LRF, sonar,...) and the integration with commercial robotic platforms (i.e., MobileRobots, Segway...) [3]. Other simulation environments are Carmen [4], Microsoft Robotics Studio [5] and USARsim (for RoboCup) [6]; these are taking the attention of scientific community for the full integration with a lot of commercial platforms. For the UAV branch of robotics the state of the art is a bit different.

In this paper a framework for simulation and testing, oriented to rotary wings aerial vehicles, is presented. The framework allows the simulation of UAVs (as stand-alone agent or exchanging data for cooperation) owing to a *Ground Control Station* (GCS) that supervises the tasks of each agent involved in the mission. The control of a single agent can be switched between the GCS and a human pilot using a joystick. The framework, based on modularity and stratification in different specialized layers, allows an easy switching from simulated to real environments, thus reducing testing and debugging times. CAD modelling supports the framework mainly with respect to extraction of geometrical parameters and virtualization. Useful applications of the framework include pilot training, testing and validation of UAVs control strategies, especially in an educational context, and simulation of complex missions.

The paper is organized as follows: next session introduces our framework. The use of a UAV CAD modelling for parameter extraction and simulation aids is proposed in Section III; the modelling activity is contextualized to the Bergen Twin Observer Helicopter. In Section IV, a test case involving two helicopters in a leader-follower mission is presented. Section V presents a methodology to validate the proposed framework. In Section VI conclusions and future works are outlined.

2 Framework

Robotic systems are inherently multi-disciplinary and for such applications software aspects are of prime importance. Even a single robot application generally

implies the use of external hardware and sensors having each their own control system and has de facto a distributed architecture. Several research has been devoted to build simulation frameworks of distributed systems. Two different approaches have been considered when identifying requirements for a framework. The first approach takes into account the functionality of typical applications that would be performed with the framework itself, whereas the second one considers the needs of potential users. From this analysis we derived the following requirement list for our framework: integration of different robotic systems, concurrent control of several robots, platform independent GUI, shared control between several users, easy integration of user's algorithms, flexibility (Distribution, Modularity, Configurability, Portability, Scalability, Maintainability), performance and efficiency. It is obvious that some requirements conflict with each other: performance and efficiency for instance have to be traded with flexibility.

Looking at the simulator panorama, game engines and flight simulators are the only available frameworks to simulate UAVs. Also most of them are developed for planes and not for helicopter. Game engines (like FlightSimulator [7] or FMS [8]) are optimal for visualization, while flight simulators (like JSBSim, YASim and UUIU [9]) are characterized by a high-fidelity mathematical model, but they are lacking in visualization. A good but expensive exception is the RotorLib developed and commercialized by RTDynamics [10]; in the helicopter context, frameworks with good performances are almost absents [11]. The framework here proposed aims at overtaking this lack. In Fig.1 a graphical abstraction with the main layers of the developed simulator is shown. The stratification of the framework permits to identify five different modules as Supervision, Communication, Dynamics, Agent, User Interaction.

An interface for sockets allows the data exchange between GCS and agents in the case of simulated agents, while the communication makes use of a dedicated long-range radio modem if a real vehicle (e.g., helicopter) is used [12].

All the modules are implemented in Matlab/Simulink; the main motivation of this choice is the reduced complexity for code development and costs of commercial products. In particular, the end-user of the framework can easily integrate his

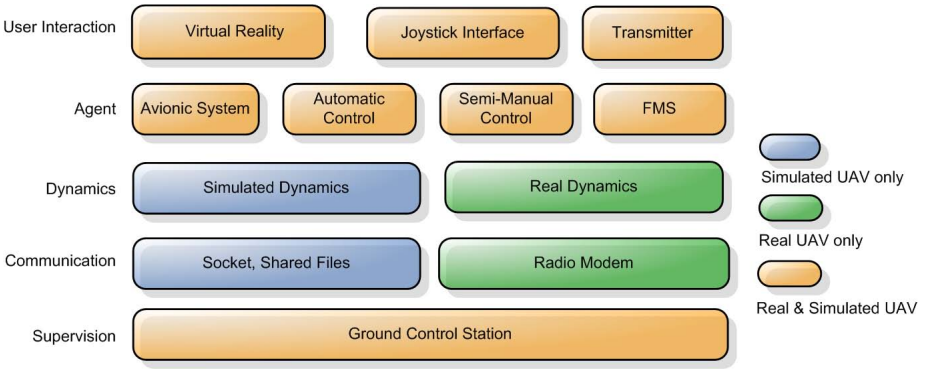

Fig. 1. Structure of the framework for UAV simulation

code for developing and testing an algorithm, e.g., for obstacle avoidance, without the necessity of re-compiling other activities. An additional motivation for the adoption of Matlab is the capability to interface the AeroSim toolbox released by Unmanned Dynamics [13]. The AeroSim Blockset is a Matlab/Simulink block library that provides components for rapid development of nonlinear 6-DOF aircraft dynamic models. In addition to aircraft dynamics the blockset also includes environment models such as standard atmosphere, background wind, turbulence, and earth models (geoid reference, gravity and magnetic field). These blocks can be added to the basic framework to increase the realism of simulation.

2.1 Agent Structure

In a simulated or real case, the structure of an agent in the context of Unmanned Aerial Vehicles is based on a complex interaction of different specialized modules. In the real case, the *Flight Management System* (FMS) is implemented as real-time code running on high performance architectures; in the simulation environment, FMS is a complex set of S-functions to reduce the simulation complexity.

However, in both cases FMS has a series of basic packages as: Communication Module, Queue of Tasks, Guidance Module, Fast Path Re-planner, Attitude and Pose Estimator, Auto and/or Semi-Manual Pilot, Obstacle Avoidance, Fault Diagnosis Identification and Isolation (see Fig.2).

Tasks like take-off, landing, point to point or waypoint navigation are currently available in the developed framework.

FMS exchanges data continuously with GCS for telemetry and task assignments/supervision. Its *Communication Module* makes use of sockets or functions to interface the radio modem.

References about position are generated by the *Guidance Module*, which decides step by step what references should be passed to the controllers (auto-pilot). This module takes into account the actual position of the agent with respect to the local inertial frame and the goal to reach.

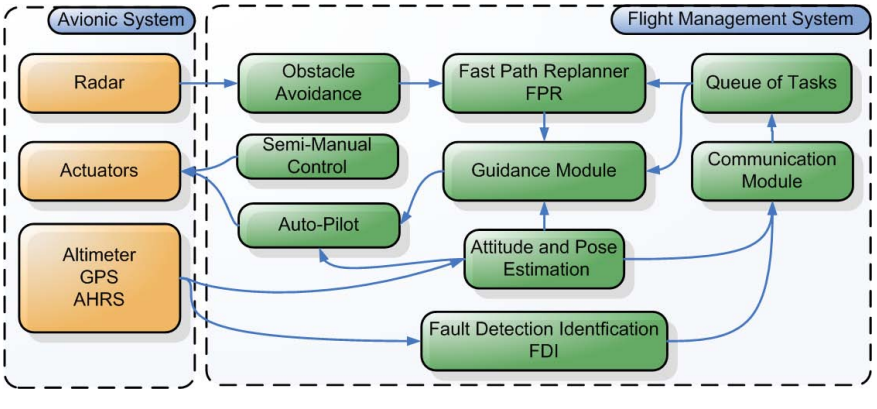

Fig. 2. The Avionic and Flight Management Systems

The *Fast Path Replanner* (FPR) provides a real-time re-calculation of path according to information provided by the *Obstacle Avoidance* package. FPR provides also for correcting the path if external disturbances (e.g., wind) generate a high error in position.

The *Attitude and Position Estimator*, using the inertial data obtained by the *Attitude Heading Reference System* (AHRS) and the *Inertial Navigation System* (INS) calculates the position and attitude of the vehicle; inertial strapdown equations are currently implemented and solved [14]. The *Auto and/or Semi-Manual Pilot* is the core of vehicle's control. The user can control a set of axes by a joystick/transmitter interface. This feature is especially suitable in the field of photogrammetry, where the user concentrates only on forward and/or lateral movements, while the control of altitude and heading (heading lock) is performed by inline controllers. The adopted philosophy tries to emulate the training process of a novel-pilot, who usually controls directly only a limited set of vehicle's axes, while the teacher supervises the activities.

Controllers can be easily updated or modified by changing their code; no additional activities are required. Controllers can be simple PID or PID with gain scheduling and fuzzy logic. Feedback linearization is available in the framework, with some tricks to increase its robustness: computational cost is a major drawback of this technique. Other control techniques, e.g., based on H_∞ can be included.

The *Obstacle Avoidance* module tries to avoid obstacles owing to information obtained by the avionic system, e.g., by radar and Laser Range Finder (LRF) sensors; actually, a set of modules (based on fuzzy logic) are available in the framework to improve the safety of vehicles during navigation.

The *Avionic System*, in the case of simulated or real vehicles, is formed by actuators and sensors. Actuators are usually analog or digital servos in reduced-scale helicopters; a second order model to represent the servos dynamics is adopted in simulated environments. Sensors provide information for a large set of aspects as navigation, obstacle avoidance, mapping and other. Using a radar sensor new tasks become feasible, as flight/operate at a given altitude or avoid an unexpected obstacle. In fact, the *Radar Altimeter* provides the altitude above the ground level, calculated as the difference between height above the sea level and ground elevation (Digital Elevation Model maps); geometric corrections, due to pitch and roll angles, are then applied. Noise is added to make the simulation more realistic; failure occurrences are also simulated. Simulated sensors as IMU and AHRS are also available in the framework; in this case an error model of each sensor is implemented (misalignment, temperature drift, non-linearity and bias).

In a similar way, to emulate the Global Position System (GPS), the geographic coordinates of the aircraft are computed from the knowledge of data about a starting point; noise is then added to match the performance of a common GPS receiver able to apply EGNOS corrections.

An analysis of main differences between the real and simulated case is presented in Table 1; this table summarizes an analysis of main differences between real and simulated case.

Table 1. Many elements (features) are shared between real and simulated scenario. The main difference concerns the Communication and Avionic System.

Aspect	Simulated Scenario	Real Scenario
Supervision	Similar	Similar
Communication	Socket	Radio Modem
Dynamics	Blade element, Actuator Disk	Real Phenomena
FMS	Similar (different control laws)	Similar
Avionic System	Simulated Sensors & Actuators	Real HW
User Interaction	Similar	Real streaming video

The switch from virtual to real world and vice versa is relatively easy; mainly FMS is the module that requires different set-up especially for the automatic control (control laws of each servo installed on the real helicopter).

2.2 Helicopter Dynamics Simulation

The framework is actually provided with a helicopter mathematical model. Employing the principles of modularity and standardization, the complete model is broken down into smaller parts that share information and interact among themselves, as shown in Fig.3. In particular, we identified four subsystems describing actuator dynamics, rotary wing dynamics, force and moment generation processes and rigid body dynamics [15].

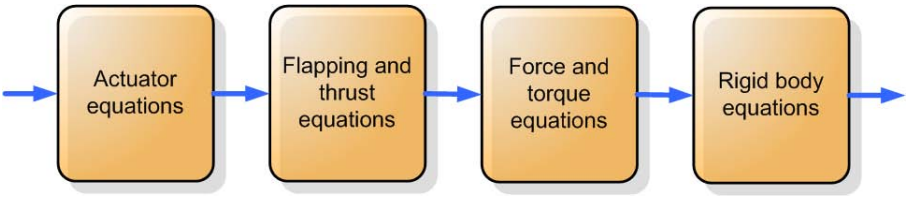

Fig. 3. Helicopter dynamics

2.3 Ground Control Station

The Ground Control Station has a lot of capabilities among which telemetry data acquisition and data logger for post flight analysis; in the cooperative context GCS is responsible for mission and task allocation/supervision. Data are collected and sent using the communication layer. A GUI was developed to obtain a visual feedback of a single agent, all agents, mission status, telemetry. A screenshot of the developed GCS is shown in Fig.4. User can control the mission of each agent choosing the vehicles; the main panels allow to monitor in real-time the agent status owing to the Attitude Direction Indicator (ADI); information as global position (GPS coordinate), status of embedded electronics-batteries, fuel consumption are currently shown in the GUI. An interesting feature of GUI is

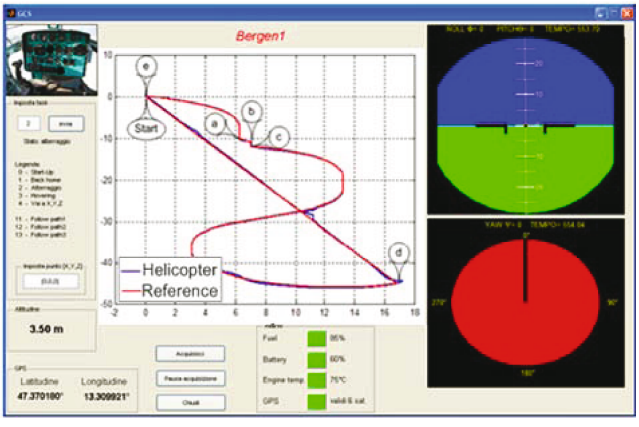

Fig. 4. A screenshot of developed GCS' GUI

the capability to control directly a set of vehicle's axes using a joystick interface; in this case the interaction between human and machines (remote controlled vehicles) allows to control the vehicles taking into account the information provided by ADI indicators. In a simulation context, joystick is interfaced using the Virtual Reality Toolbox (described below).

2.4 Virtual Reality and World Representation

A synthetic rendering of world and agents is one of the basic module; as mentioned in the introduction section, market offers a series of different complex systems to virtualize world and agents. The choice adopted in the proposed framework is to integrate the Matlab Virtual Reality Toolbox. A VRML model of world (environment) and agents (aerial vehicles) can be easily controlled in a Simulink diagram. Students are often familiar with the Matworks software. The mission area is represented as a digital grid map or Digital Elevation Model (DEM). A set of different world scenarios is available in the developed framework. Scenarios are generated considering the DEM of mission area. DEM maps represent the mission area (real or simulated) as a grid map; a critical parameter is the cell resolution. The resolution of available maps is usually 10m in the case of real scenarios. This value is too high in critical mission where an accurate localization is required. The GUI allows to edit/create a new DEM to overtake this limitation; data are obtained by exploration of mission area.

Virtual Reality Toolbox is used to present in soft real-time the state of each vehicle involved in the mission. A VRML world can be customized in terms of textures, position of camera(s) (attached to vehicle or fixed), light(s). The above mentioned toolbox is also used to interface a joystick; this kind of device allows a manual control of the helicopter (user can select the set of axes that wants to control). This features is really useful for novel pilot(s) during the training phases. A 3D model of Bergen Twin Observer Helicopter was developed; a more

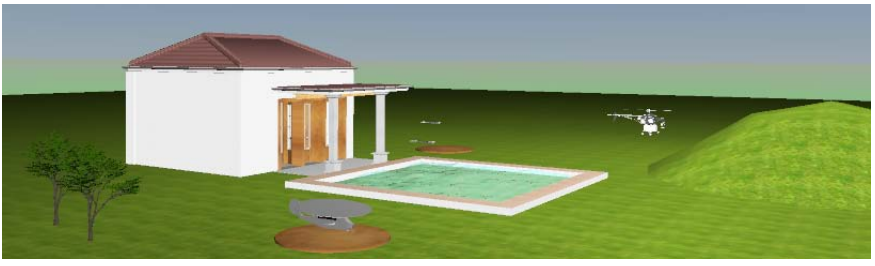

Fig. 5. An example of scenario where two UAVs are performing a mission

detailed introduction to 3D CAD modelling is presented in Section 3. In Fig.5 a basic virtual scenario is presented.

Currently, work is focused on the adoption of other virtual reality environments inspired to flight simulators games as FlightGear [16] and Microsoft Flight Simulator [7].

3 CAD Modelling

CAD modelling plays an essential role, supporting the framework mainly with respect to mathematical model parameterization and virtual reality rendering.

Blocks describing the helicopter simulated dynamics need a set of geometrical and inertial parameters such as inertia matrix, mass, distances between Centre Of Gravity (COG) and force attacking points, rotors geometry and leverage gains.

Fig. 6. A view of the CAD model of Bergen Twin Observer; the transparencies allow to see hidden parts, e.g., avionic box and fuel

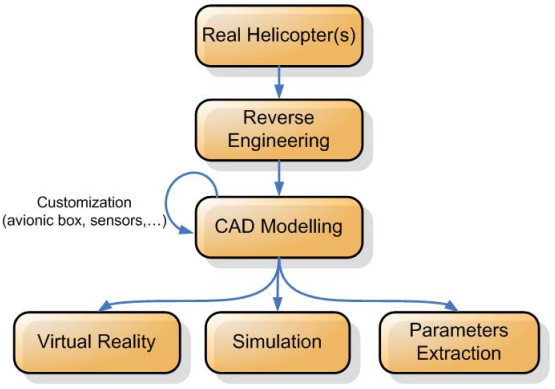

Fig. 7. Diagram of Reverse engineering process

Providing the model with the real parameters of a specific helicopter makes really useful the simulations, allowing an effective shift of results to real applications.

Because some data are time-variant, due to fuel consumption, not trivial to be determined, and should be re-calculated at every change of mass disposal like a new device installation, a detailed 3D CAD model helps to solve the problem, allowing to simply extract all information needed.

Performing an accurate reverse engineering process on a RC mini helicopter a model implementation was carried out. The Bergen Twin Observer available at our laboratory, designed in Solid Edge environment, is presented in Fig.6.

Solid Edge represents a standard in 3D mechanical modelling. It is a powerful feature-based CAD software, quite simple to use and available in a cheap academic license. It allows an accurate and rapid design of the system and its geometric and inertial characterization. The model can be exported to specific software, saved in VRML format or merely used for a rendering process. In Fig.7 the whole procedure is shown.

The obtained digital model can be mostly used to evaluate the effect of customization (e.g. addition of payloads, sensors) by simply extracting geometrical and inertial parameters after any structural or set up variation. It is also functional to visualize the agent in a Virtual Reality environment, allowing a pleasant and more significant representation of the simulation results.

4 Test Case: A Leader-Follower Mission

In this section, a simulation using the presented framework is reported. This simulation presents two Bergen Twin Observer helicopters involved in a "leader-follower mission" [17,18]. Leader-follower mission belongs to the problem of *coalition formation* inspired by the motion of bevies; the main objective of coalition formation is the clustering of a series of agents to reach a target or to cooperate, extending the capabilities of each agent ("union is strength") [19,20].

Simulated sensors adopted are AHRS, GPS and Radar. The helicopters are linked to GCS using sockets for data exchange. Each helicopter has five servos (digital and analog) and one main engine (piston engine). The simulation time of reported simulations is strongly close to the real time (simulation is a bit slowly in particular conditions and the lag depends on the complexity of controllers). The GCS and the two instances of helicopters run on three PCs connected by an Ethernet link.

Leader starts the mission (a simple circular path) and follower tends to maintain a fixed distance minimizing the error in terms of the following expression.

$$P(s) = \left[x(s)\ y(s)\ z(s) \right]^T \ \| P_f(s_f) - P_l(s_l - k) \|^2 < \varepsilon k, \varepsilon \in \mathbb{R}$$

where subscript l and f stand for leader and follower, respectively (see Fig. 8); P is the helicopter position and k the distance between helicopters evaluated along the trajectory.

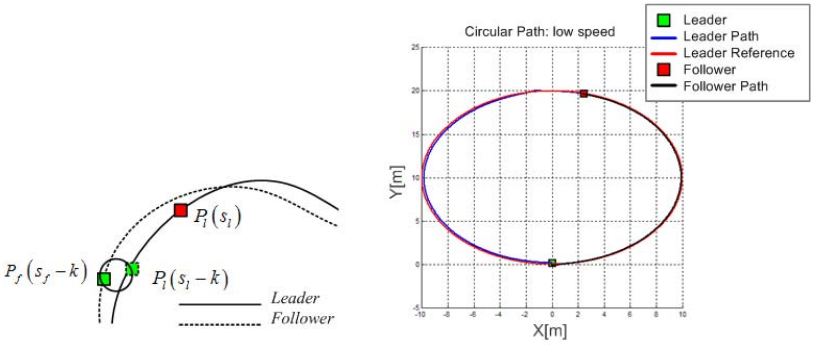

Fig. 8. During the flight, the helicopter-follower maintains a fixed distances to leader

Follower estimates leader trajectory using leader position that is obtained by radar and GCS telemetry. Then, on the base of estimated trajectory, follower tends to track leader trajectory minimizing the error [21]. A graphical representation of simulation is shown in Fig. 8.

5 Validation

According with a principle of modern behaviour-based robotics, an efficient framework should omit internal representations, centering rather on the direct relation between stimulus and action. Hence a quality simulator has to implement accurate models of: robots' geometry and kinematics, sensors, environment and, finally, robot-environment interactions. Since all these components work properly, simulation will provide an adequate model of the process and the results may be shifted to real applications.

This approach to robotics research, however, depends crucially on validation of the models used so that researchers have reasonable assurance that the problems

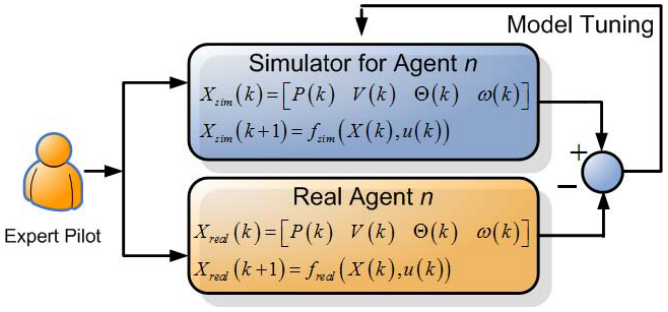

Fig. 9. An approach for fine-tuning of simulation model

they encounter and solutions they devise are representative of actual problems and solutions in robotics rather than simply artifacts of the simulation.

The level of effort devoted to validation has been a distinguishing feature of this work, even if in preliminary stage. Each of its major constituents (robot kinematics, interaction with the environment, sensors and camera video) have been subjected to ongoing validation testing.

A schematic idea for *fine model tuning* is shown in Fig. 9. An expert pilot controls the helicopter performing manoeuvres with a high dynamic content. The actions of pilot are real-time recorded; real and simulated measures of agent state are compared evaluating the goodness of the simulation model; this methodology is inspired by adaptive control systems (e.g, Model Reference Adaptive System).

6 Conclusions and Future Works

In this paper a framework for UAV simulation in cooperative scenarios and testing was presented. The modularity of its architecture permits to update or rewrite a block in a short time; new controllers can be easily tested. This activity does not require re-compiling or deep rearrangement of the code.

Adding or changing the mathematical model, different aerial vehicles can be simulated; actually the research unit is working on simulating a quad-rotor helicopter. This kind of vehicle is versatile and useful for short range missions; due to these characteristics, the quad-rotor is widely used by researchers in the UAV context. Moreover, the proposed approach allows an easy switching from simulated to real environments; this is possible owing to stratification of functions in specialized layers. User interaction, e.g., training of novel-pilots, is supported by GCS, joystick or RC-transmitter interfaces.

Future works will be steered to improve the quality of the VR module for an easy interaction with vehicles without a video streaming feedback. Integration of new kind of aerial vehicles will be the main activity. The adoption/integration of FlightGear or Microsoft Flight Simulator graphical engines will be then investigated. New robust non-linear control techniques to enhance the performance (in terms of energy consumption and Time To Task) of agents will be tested.

At the end of validation phase (introduced in Section V), the simulator will be relesed to scientific community under the GNU license.

References

1. Sanders, C.P., DeBitetto, P.A., Feron, E., Vuong, H.F., Leveson, N.: Hierarchical control of small autonomous helicopters. In: Proceedings of the 37th IEEE Conference on Decision and Control (December 1998)
2. Frontoni, E., Mancini, A., Caponetti, F., Zingaretti, P.: A framework for simulations and tests of mobile robotics tasks. In: Proceedings of 14th Mediterranean Conference on Control and Automation, MED 2006 (2006)
3. Collett, T.H.J., MacDonald, B.A., Gerkey, B.P.: Player 2.0: Toward a practical robot programming framework. In: Australasian Conf. on Robotics and Automation, ACRA 2005 (2005)
4. Carmen robot navigation tool kit (2008), `http://carmen.sourceforge.net/`
5. Microsoft robotics studio (2008), `http://msdn.microsoft.com/robotics/`
6. Usarsim (2008), `http://sourceforge.net/projects/usarsim/`
7. Microsoft fs (2008), `http://www.microsoft.com/games/flightsimulatorX/`
8. Fms project (2008), `http://www.flying-model-simulator.com/`
9. Jsbsim home page (2008), `http://jsbsim.sourceforge.net/`
10. Rotorlib home page (2008), `http://www.rtdynamics.com/v1.0/`
11. Taamallah, S., de Reus, A.J.C., Boer, J.F.: Development of a rotorcraft mini-uav system demonstrator. In: The 24th Digital Avionics Systems Conference, DASC 2005 (2005)
12. Frontoni, E., Mancini, A., Caponetti, F., Zingaretti, P., Longhi, S.: Prototype uav helicopter working in cooperative environments. In: Proceedings of IEEE/ASME international conference on Advanced intelligent mechatronics, AIM 2007 (2007)
13. Aerosim toolbox (2008), `http://www.u-dynamics.com/aerosim/`
14. Jetto, L., Longhi, S., Venturini, G.: Development and experimental validation of an adaptive extended kalman filter for the localization of mobile robots. IEEE Trans. on Robotics and Automation 15(2), 219–229 (1999)
15. Koo, T.J., Sastry, S.: Output tracking control design of a helicopter model based on approximate linearization. In: Proceedings of the 37th IEEE Conference on Decision and Control (1998)
16. Flightgear project (2008), `http://www.flightgear.org`
17. Wang, X., Yadav, V., Balakrishnan, S.N.: Cooperative uav formation flying with obstacle/collision avoidance. IEEE Transactions on Control Systems Technology 15, 672–679 (2007)
18. Lechevin, N., Rabbath, C.A., Sicard, P.: Trajectory tracking of leader-follower formations characterized by constant line-of-sight angles. Automatica 42(12), 2131–2141 (2006)
19. Merino, L., Caballero, F., Martinez de Dios, J.R., Ollero, A.: Cooperative fire detection using unmanned aerial vehicles. In: Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2005 (2005)
20. Beard, R.W., et al.: Decentralized cooperative aerial surveillance using fixed-wing miniature uav. Proceedings of the IEEE 94(7), 1306–1324 (2006)
21. Mahony, R., Hamel, T.: Robust trajectory tracking for a scale model autonomous helicopter. Int. Journal of Robust and Nonlinear Control 14(12), 1035–1059 (2004)