A test bed for rapid flight testing of UAV control algorithms

Tiago Oliveira [*], Gonçalo Cruz [†], Eduardo R. B. Marques [‡] and Pedro Encarnação [§]

[*] *Academia da Força Aérea Portuguesa, Portugal, tmoliveira@academiafa.edu.pt*

[†] *Academia da Força Aérea Portuguesa, Portugal, gccruz@academiafa.edu.pt*

[‡] *Faculdade de Engenharia da Universidade do Porto, Portugal, edrdo@dcc.fc.up.pt*

[§] *Faculdade de Engenharia da Universidade Católica Portuguesa, Portugal, pme@fe.lisboa.ucp.pt*

Abstract

This paper presents a test bed for rapid flight testing of control algorithms for unmanned aerial vehicles (UAVs), framed in the Portuguese Research and Development Program on Unmanned Aerial Vehicles (PITVANT). The test bed can be used to validate UAV control algorithms at the kinematic level, being the aircraft dynamics controlled by commercially available autopilots. It supports all stages from numerical simulations to flight testing (with control on the ground) using the same framework. It allows testing multi-vehicle operations and the integration of on board systems with data from and to those systems transmitted through a single communications data link. A common console integrates all information and controls every aspect of multi-vehicle missions. The paper describes the architecture of the test bed and its deployment, in particular its operational framework, software and hardware components and user interaction.

keywords: test bed, UAV, control algorithms, hardware-in-the-loop simulation, software architecture

1 Introduction

The Portuguese Research and Development Program on Unmanned Air Vehicles (PITVANT), supported by the Portuguese Ministry of National Defense, has been carried out since 2009 by the Portuguese Air Force Academy (AFA) and the Faculty of Engineering of the University of Porto (FEUP). The main objectives of this seven-year project are the exploration of small platforms and the development of new technologies and new concepts of operation, with an emphasis on the exploration of cooperative control systems for teams of autonomous aerial, marine and land vehicles. The envisioned applications of the systems to be developed are forest and coastal patrolling, military aerial surveillance, search and rescue and support/tracking of land or marine vehicles.

Currently, the AFA owns a variety of platforms with different wingspans (from 2 m to 7 m) and maximum takeoff weights (from 2Kg to 100Kg) built from scratch, allowing for the testing of technological systems on board and different control algorithms.

Given the early stage of the project, testing of the control algorithms has been carried out on small platforms. Namely, the ANTEX-M X02 platforms (Figure 1) have been widely used due to their convenient size/payload ratio Table 1).

Figure 1: ANTEX-M X02.

Maximum takeoff weight	10 Kg
Wingspan	2.415 m
Payload	4 Kg
Maximum Speed	100 Km/h
Autonomy	1.5 h

Table 1: ANTEX-M X02 - Main features.

The main goal of the work reported here was to develop a software and hardware architecture that allows rapid hardware-in-the-loop simulations and subsequent flight tests of different high level multi-vehicle control systems.

The standard approach with UAV control is to assume that the vehicle has an off-the-shelf inner loop controller that accepts references at kinematic level (angular rates and linear velocities) and generates the UAV control signals necessary to follow those references in the presence of model uncertainty and external disturbances, like wind. See for example [1, 2, 3, 4, 5, 6, 7]. Outer loop control laws are thus derived using a kinematic model of the vehicle and provide the references to the inner control loop. This approach permits the development of control algorithms based on simpler models. The use of an off-the-shelf autopilot also adds to the operational safety of the aircraft since it is always possible to switch to autopilot control mode or even to remote control mode, if a potentially dangerous situation is detected. The downside is that it usually complicates formal proofs of convergence and performance analysis of the designed control algorithms, due to the inner-outer loop structure. Following the same strategy, all the PITVANT platforms are equipped with a Piccolo II controller [8]. Within this R&D project, the UAV control algorithms are usually developed and tested in simulation using Matlab [9]. Thus, one of the main features of the here reported test bed is that it provides a mean to rapid testing of kinematic control laws running under Matlab, allowing for the validation and implementation of the high level controllers in different controlled conditions before final implementation. Firstly, it provides hardware-in-the-loop simulations, where the vehicle dynamics is simulated by software provided by Piccolo. Secondly, it allows for control algorithm tests in real flight situations with the control loop closed by a Matlab routine running on a PC on the ground. Thirdly, it is possible to close the control loop with an embedded computational system (*e.g.* PC-104). The latter flight tests only differ from the final application by the localization of the computational system (on ground instead of on on board the aircraft). In fact, these flight tests are more demanding since with the on board software it is possible to achieve faster sensor data and control updates. Finally, the presented architecture allows the fast migration of the embedded system to the aircraft.

Test beds for UAV testing have been reported in the literature. They differ from each other essentially

in the system architecture utilized. In [7], both high level planning strategies and low-level servo commands are computed on the same hardware. Other research groups use a separate off-the-shelf hardware to compute low-level control commands, such as aileron and elevator servo commands [1, 5]. For collaborative control applications, the high-level planning may occur on a ground station computer [2], or onboard the aircraft [6]. The hardware and the software used for the flight tests may also be commercial [5] or proprietary [3]. In these architectures, the telemetry from the on board systems is usually transmitted to the ground using a dedicated wireless data link. For command and control purposes, a second telemetry link operates between the autopilot and the ground control station.

In the here proposed test bed, the telemetry from all the on board systems and their control commands can be performed through a single communications data link and a common command and control console, shared by all the vehicles of the network. This feature enables testing missions with multiple platforms saving hardware. The proposed test bed was successfully utilized to test several different control algorithms designed for the PITVANT project, namely algorithms for target tracking [10], obstacle avoidance [11] and thermals navigation [12].

The paper describes the hardware and software architecture of the PITVANT test bed and is organized as follows. Section 2 presents the basic control architecture used for the autonomous flights using autopilot tools and the proposed test bed software and hardware architectures. Section 3 describes the entire process of control algorithms validation and implementation, from numerical simulations to flight tests, including hardware-in-the-loop validation. Finally, section 4 presents the main conclusions and future work.

2 Test bed software and hardware architecture

2.1 Operation with a Piccolo autopilot

All PITVANT platforms are equipped with a Piccolo II controller that deals with the inner control loop of the UAVs. It relies on a mathematical model parameterized by the aircraft geometric data and has a built-in wind estimator. Several model and controller parameters can be set by the user [8]. The ANTEX-M X02 model parameters were fine tuned in more than 30 hours of autonomous flights.

A Piccolo Ground Station and a Piccolo Command Center (PCC) can be used for basic UAV operation (see Figure 2). The Piccolo Command Center (PCC) is a graphical user-interface for operators developed by Cloudcap that shows flight information and allows the operator to send waypoints, velocity, bank or altitude references to the aircraft (Figure 3).

Telemetry data is transmitted to the ground station via a 2.4 GHz Piccolo link. If an on board system is added (e.g. a video camera), another data link must be used to relay the video signal to a different monitor on the ground. Although simple and easy to operate, this architecture does not allow the control of the on board systems that are not compatible with the Cloudcap's software. Additionally, the PCC does not support the inclusion of command and control windows devoted to the operation

with the control algorithms implemented. Next section describes the here proposed architecture that overcomes the limitations of this basic setup.

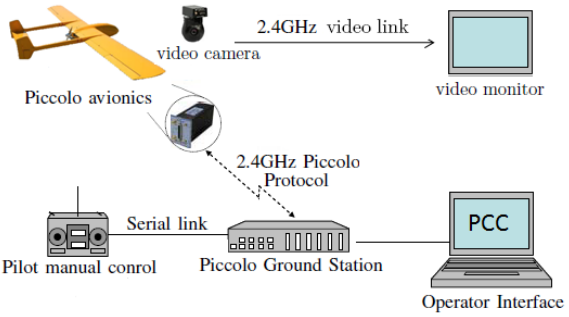

Figure 2: Operation with a Piccolo autopilot.

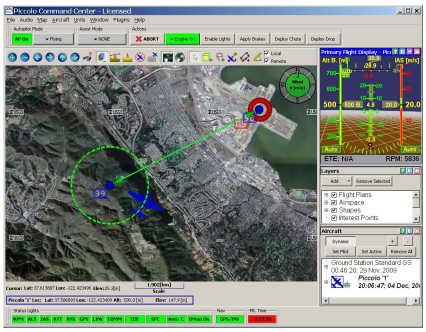

Figure 3: Piccolo Command Center.

2.2 Proposed test bed

All development and test stages within the PITVANT project share the software architecture depicted in Figure 4. It has a layered structure, comprising network control modules, DUNE software tasks, and physical or simulated hardware.

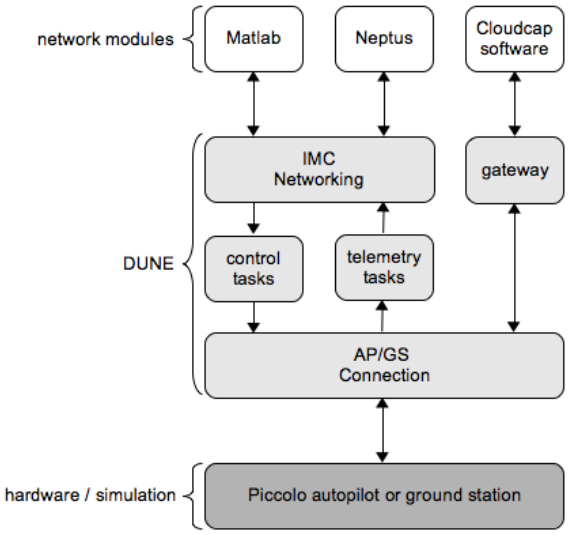

Figure 4: Software architecture.

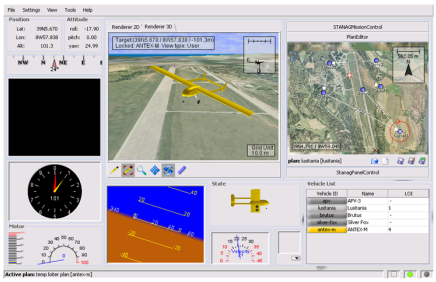

Figure 5: Neptus command and control console.

The physical hardware comprises Piccolo autopilots and ground stations. These may also be simulated using software tools provided by Cloudcap.

DUNE is an embedded software platform used in several autonomous vehicles developed by the Underwater Systems and Technology Laboratory (LSTS) at FEUP (please refer to [13] for an application to autonomous underwater vehicles). DUNE provides programmers with a C++ abstraction of real-time tasks, and runs on several heterogeneous operating systems (*e.g.*, Linux or Windows) and hardware platforms (*e.g.*, Intel x86 or ARM). In this context DUNE is used to implement software tasks that:

- handle the physical/simulation connection to an autopilot or ground station;

- provide a network gateway for that connection to Cloudcap software;
- decode and encode, respectively, telemetry and control packets [14], from Piccolo and other on board systems (*e.g.* gimbal camera), from and to IMC packets [15];
- locally control on board systems and aircraft systems that can run independently from the ground station.

IMC is a message-oriented interoperability protocol [15] that is used in all LSTS software deployments of autonomous vehicles. When a new system is included on board (*e.g.* gimbal) the DUNE software is used to convert its communication protocol to IMC, thus allowing its control from the ground, without relying on the commercial software that comes with the equipment. Therefore, the telemetry from all on board sensors and their control commands can be performed through a single communications link, instead of using a particular control software and a dedicated communication link for each system. Moreover, in the proposed architecture, IMC is shared by all vehicles, thus allowing the exchange of information between heterogeneous systems (e.g., aerial and marine vehicles) and the use of a single console for mission control.

Network modules may comprise instances of Matlab, Neptus [16], and GUI software from Cloudcap. Cloudcap GUI applications can be used to operate UAVs equipped with the Piccolo autopilot, and interface with DUNE's gateway functionality for that purpose. Neptus is a command and control infrastructure for autonomous vehicle missions, covering diverse aspects such as mission planning, online vehicle control and monitoring, or data analysis (Figure 5) [16]. The Neptus console gives the operation team the freedom to create a terminal suited to the needs of the mission and allows for the inclusion of new command and control windows associated with the control algorithms implemented. Thus, the proposed architecture enables the simultaneous use of Matlab, Cloudcap and Neptus consoles to control the missions. This is useful since different stages of algorithm validation require different consoles. While Matlab is in the loop, dedicated Matlab consoles can be used to monitor the algorithm performance, and PCC provides the safety of an alternative way to monitor and control the aircraft. When the algorithms migrate from Matlab to C++, everything can be monitored and controlled from Neptus, using again the PCC as a redundant system.

Figure 6 presents the hardware architecture of the test bed. PCC, Matlab and/or Neptus can receive telemetry data from the aircraft, and send commands to the aircraft using the DUNE gateway. When the PCC is used to control the mission, commands to the aircraft are sent through the Piccolo Ground Station. That is also the case when MATLAB is in the loop. An IMC data link is used when the mission control is done through Neptus.

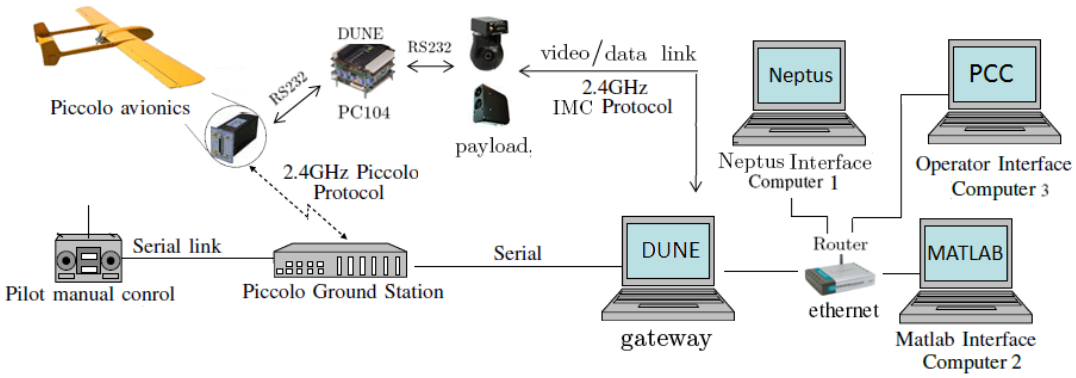

Figure 6: Hardware architecture.

3 Implementation and validation process of control algorithms

Validation and implementation of control algorithms usually goes through the following steps: numerical simulations, hardware-in-the-loop simulations, flight testing with the control algorithm implemented on a computer on ground (standard PC and then a dedicated computational system), and finally implementation of the control algorithm on an on board embedded computational system. The here proposed test bed supports all stages from numerical simulations to flight testing (with control on the ground) using the same Matlab code.

3.1 Numerical simulations

The first phase of implementation and testing of the control algorithms is their numerical validation, considering only the kinematics of the system (Figure 7). At this stage, the development team has the freedom to choose the architecture and the syntax of the code being written, including a Matlab console to monitor simulation data. In fact, different applications require different parameters and data, thus different consoles to match each application needs are usually designed.

These consoles are often created utilizing GUIDE [9] to rapidly create a suitable Graphical User Interface (GUI).

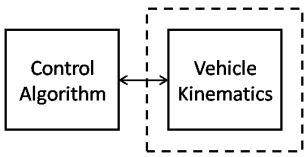

Figure 7: Numerical simulations structure.

Despite the freedom involved in this first stage, the code implementation should be done keeping in mind its future hardware-in-the-loop and real flight tests. Thus, the control algorithm and its console should be implemented independently of the vehicle kinematics.

3.2 Hardware-in-the-loop simulations

Hardware-in-the-loop (HIL) simulations refer to simulations were the Piccolo autopilot and the Piccolo Ground Station are included in the loop, replacing the model simulation used for numerical simulations (Figure 8). A communication block to allow the exchange of data between the controller and the Piccolo autopilot must be included. In the here proposed test bed, the UAV dynamics is simulated by the Simulator software provided by Cloud Cap (Figure 9). It receives the commands from the Piccolo autopilot and provides the UAV state information via a CAN bus interface. The communication between the Piccolo and the algorithm implemented on Matlab is mediated by the Ground Station and DUNE. The Ground Station is connected to the Piccolo autopilot via an UHF link, receiving the UAV state and transmitting the control references, and to the PC that runs the algorithm through a serial cable (Figure 9).

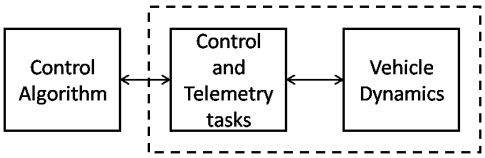

Figure 8: Hardware simulations and flight tests basic structure.

Hardware-in-the-loop simulations are instrumental in identifying interface problems between different subsystems and in validating the control algorithm now including the dynamics inner loop closed by the autopilot. The proposed setup still relies on a Matlab implementation of the control, thus facilitating control algorithm modifications. Additionally, Matlab consoles can still be utilized to monitor the control system's performance.

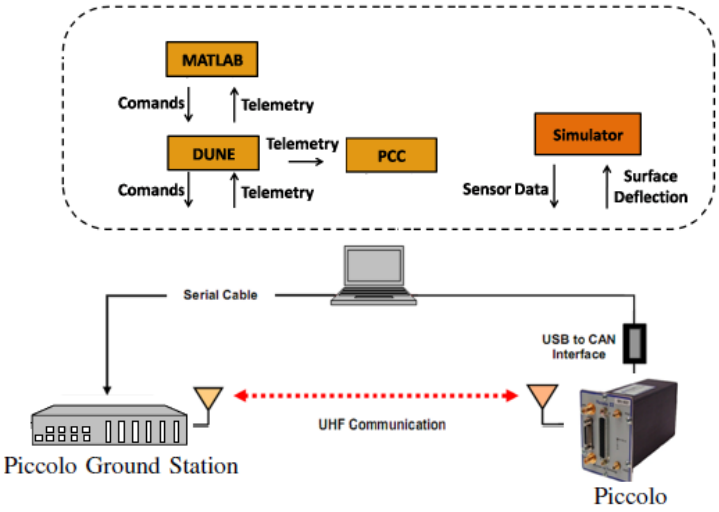

Figure 9: Control system architecture used in the HIL simulations. Adapted from Piccolo Setup Guide [8].

Figure 10 presents a hardware-in-the-loop simulation of an obstacle avoidance algorithm [11]. The usage of MATLAB at this stage enables an easy design of intuitive interfaces (left-hand side of Figure 10) to tune the algorithm before testing it on a real flight. Theses interfaces provide a better understanding of the system's behaviour and can be used during the flight tests. From the Matlab routine point of view, there is no difference between the hardware-in-the-loop and the real flights operations. After tuning the algorithm, the user can evaluate its performance by using Matlab's set of tools, as shown in right-hand side of Figure 10.

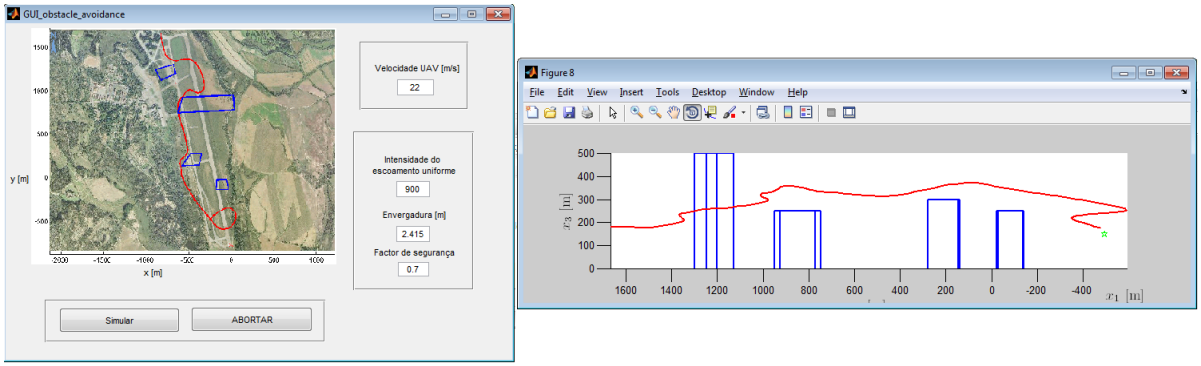

Figure 10: Trajectory obtained in a obstacle avoidance test, using hardware-in-the-loop simulations.

This test bed was also used to perform hardware-in-the-loop simulations of target tracking [10] multi-UAV missions, and thermal estimation algorithms [12].

3.3 Flight tests

For the flight tests, the Piccolo autopilot is on board the aircraft, along with a PC-104 and several other systems, including video cameras and other sensors. The autopilot sends the telemetry data to the ground station through a 2.4 GHz Piccolo protocol link. The data received can be used to close the UAV control loop using a standard PC. This is useful for the first flight tests where the control algorithms are still implemented in Matlab and run on a computer on ground.

Using DUNE software [16] and an ethernet connection, the computer running Matlab is enabled to receive telemetry data, compute the new control references and send them to the aircraft via Piccolo Ground Station (please refer to Figure 6). Using the Neptus console [16] is possible to monitor the telemetry and video data.

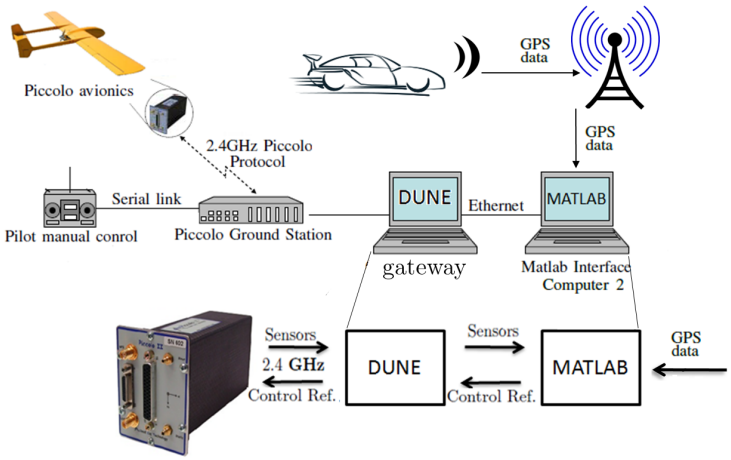

Figure 11: Fligth test operation frame for target tracking.

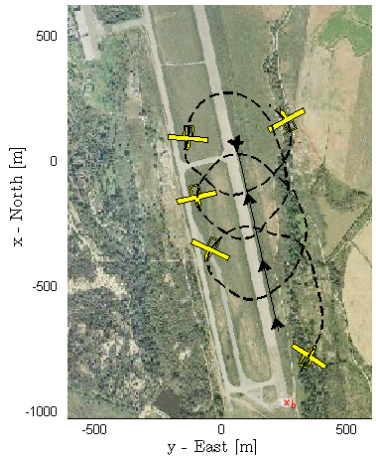

Figure 12: Aircraft's trajectory following a target on the ground.

The above described hardware architecture was successfully utilized to test a wide range of high-level controllers within the PITVANT project, namely a ground target tracking algorithm, using a GPS in the loop to provide the target state (Figures 11 and 12) [10], and a thermal estimation algorithm[12].

The following validation and implementation stages require the implementation of the control algorithms in an embedded computational system. This can be done within the DUNE framework [13], following specific rules and design methods still under development within the PITVANT project.

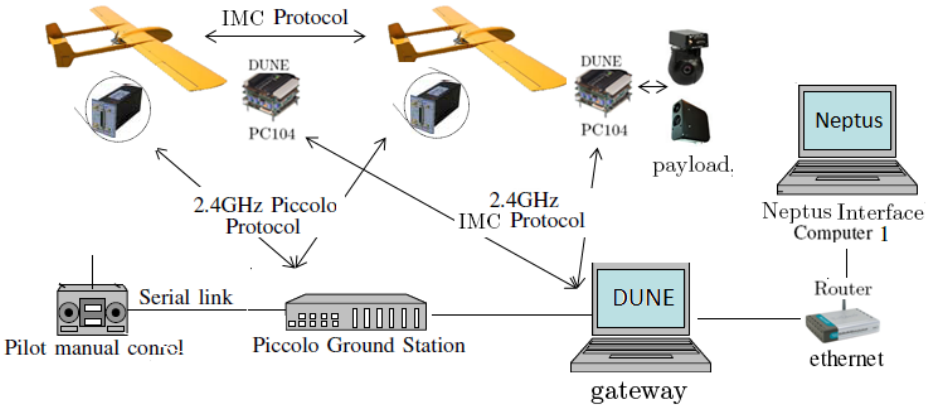

Figure 13: System's final hardware architecture.

Figure 13 presents the envisioned system's final architecture after the migration of the control algorithms from Matlab to DUNE running at the on board computational system. Neptus console will be used to send commands to the UAV and to on board systems. It will allow for the design of different windows to control and monitor each system. Through the IMC data link, different vehicles will be able to exchange information and thus high-level planning for cooperative missions may occur on board the vehicles or on a ground station.

4 Conclusions

The test bed for rapid flight testing of UAV control algorithms developed under the PITVANT research project allows for validation and implementation of high level control algorithms, from hardware-in-the-loop simulations to real flight tests with multiple vehicles. It provides a hardware and software architecture to progressively test new control algorithms in controlled situations before the final implementation on board the aircraft. Future work will include the definition of detailed operational procedures for flight testing, the establishment of software rules and design methods to implement the algorithms within the DUNE software to be run on an on board dedicated computational system, and the development of software tools to include its consoles within NEPTUS.

REFERENCES

[1] I. Kaminer, O. Yakimenko, V. Dobrokhodov, and K. Jones, "Rapid flight test prototyping system and the fleet of UAVs and MAVs at the Naval Postgraduate School," in *Proceedings of the 3rd AIAA "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, 2004, pp. 20–23.

[2] J. How, E. King, and Y. Kuwata, "Flight demonstrations of cooperative control for UAV teams," in *Proceedings of 3rd AIAA Unmanned Unlimited Technical Conference, Workshop and Exhibit*, 2004.

[3] M. Quigley, M. Goodrich, S. Griffiths, A. Eldredge, and R. Beard, "Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA*, 2005.

[4] T. Mclain and R. Beard, "Unmanned air vehicle testbed for cooperative control experiments," in *Proceedings of the 2004 American Control Conference*, 2004, pp. 5327–5331.

[5] L. Ma, V. Stepanyan, C. Cao, I. Faruque, C. Woolsey, and N. Hovakimyan, "Flight test bed for visual tracking of small UAVs," in *Proceedings of the 2006 AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.

[6] J. Tisdale, A. Ryan, M. Zennaro, X. Xiao, D. Caveney, S. Rathinam, J. Hedrick, and R. Sengupta, "The software architecture of the Berkeley UAV Platform," in *Proceedings of the 2006 IEEE International Conference on Control Applications*, 2006, pp. 1420 –1425.

[7] J. Jang and C. Tomlin, "Design and implementation of a low cost, hierarchical and modular avionics architecture for the DragonFly UAVs," in *Proceedings of the 2002 AIAA Guidance, Navigation, and Control Conference*, 2002.

[8] B. Vaglienti, R. Hoag, M. Niculescu, J. Becker, and D. Miley, *Piccolo System User's Guide*, Cloud Cap Technology, 2009.

[9] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[10] T. Oliveira and P. Encarnação, "Ground target tracking for unmanned aerial vehicles," in *Proceedings of the 2010 AIAA Guidance, Navigation, and Control Conference*, 2010.

[11] G. Cruz and P. Encarnação, "Obstacle avoidance for unmanned aerial vehicles," in *Journal of Intelligent and Robotic Systems, in press*, 2011.

[12] R. Bencatel, "Thermal localization," in *Proceedings of the 2010 AIS Conference*, 2010.

[13] A. Tinka, S. Diemer, L. Madureira, E. Marques, J. Sousa, R. Martins, J. Pinto, J. Silva, P. Saint-Pierre, and A. M. Bayen, "Viability-based computation of spatially constrained minimum time trajectories for an autonomous underwater vehicle: implementation and experiments," in *Proceedings of the 2009 American Control Conference*. IEEE Computer Society, 2009, pp. 3603–3610.

[14] B. Vaglienti, "Communications for the Piccolo Avionics, version 2.1.2," Cloud Cap Technology, 2011.

[15] R. Martins, P. Dias, E. Marques, J. Pinto, J. Sousa, and F. Pereira, "IMC: A communication protocol for networked vehicles and sensors," in *Proceedings of the 2009 IEEE Oceans Europe'09*, 2009.

[16] P. Dias, G. Goncalves, R. Gomes, J. Sousa, J. Pinto, and F. Pereira, "Mission planning and specification in the Neptus framework," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, (ICRA)*, 2006, pp. 3220–3225.