Tools for UAV operations

P. Dias E. Marques R. Martins R. Bencatel Jose Pinto and J.B. Sousa

Department of Electrical and Computer Engineering, University of Porto, Porto, Portugal

Email: pdias, bencatel, zepinto, jtasso@fe.up.pt,edrdo@dcc.fc.up.pt,rasmartins@gmail.com

Abstract

In this paper, we present various tools for safe UAV research missions. Most of the tools available from off-the-shelf autopilots are not adequate to carryout research activities and have restricted modification capability. We have developed a new command and control system (Neptus) that is flexible in terms of planning and situational awareness, messaging system that has inter-interoperability capability (IMC) and an on-board software system (Dune) that interacts with various sensors and can bridge an off-the-shelf autopilot with Neptus for operator in loop interactions. These tools have been tested for various mission lasting from minutes to hours and also under multi-UAV operations.

keywords: UAV, command and control, communication

1 Introduction

UAVs are used in many military and civilian operations ranging from surveillance [1, 2] to agriculture [4, **?**]. Operating these UAVs requires an autopilot to control the aircraft, communication capability for modification of plan and situational awareness, and command and control system for managing the operation with operator in the loop at the ground station. Most of the commerial autopilots like Piccolo [5], Micropilot [6], Kestrel [7], have propertierty softwares to handle the UAV operations. Currently, they allow modification of waypoints, command velocity and heading using the groud station. However, these softwares have restricted capablity to modify the operations which is a necessary component in research. Also, they do not have inter-operability capability that is required for multi-vehicle operations.

In order to have inter-interoperability across different systems, we have developed several tools to govern the UAV operations. First, we developed the on-board software Dune that takes care of the on-board operations that include autopilot, system health monitoring, logging and has delay tolerant networking capability. We developed an Inter-Messaging Protocol (IMC) that uses XML to send and receive messages [8]. The messaging system is compatible with STANAG 4586 [9]. Additionally, IMC allows low level control commands like JAUS (Joint Architecture for Unmanned Systems)/SAE AS-4 [10] and CCL (Compact Control Language) [11] that provide a set of commands (messages) to control

the vehicle. For command and control, we developed a software called Neptus [12] that has advanced capabilities compared to off-the-shelf ground station softwares in terms of planning the mission, review and analysis of mission, reporting and heterogeneous vehicle operations. In the following section, we provide brief description of these three systems.

2 IMC Overview

The IMC protocol comprises different logical message groups for UAV operations. It defines an infrastructure that is modular and provides different layers for control and sensing. The use of IMC in the context of message flow between subsystems is shown in Figure 1. The message flow corresponds to the several control and sensing layers within IMC:

1. *Mission control messages* define the specification of a mission and it's life-cycle, for the interface between a CCU (Command and Control Unit) such as a Neptus [12] console and a mission supervisor module. A mission is a sequence or graph of maneuvers.

2. *Vehicle control messages* are used to interface the vehicle from an external source, typically a CCU or a mission supervisor module, for example to issue maneuver commands or other external requests, and to monitor the vehicles state.

3. *Maneuver messages* are used to define maneuvers, associated commands and execution state. The simplest maneuver types are related to waypoint tracking - encoded through a *Goto* message - or loitering patterns - *Loiter*.

4. *Guidance messages* are related to the guidance used for autonomous maneuvering. A guidance step generates new reference measures for the vehicle heading, height, and velocity, in the form of a *Desired Guidance* message.

5. *Navigation messages* define the interface for reporting a vehicles navigation state. The *Estimated State* message defines a vehicles navigational state.

6. *Sensor messages* are used to report sensor readings by the respective hardware controllers. some of the sensor messages are related to IMU, GPS, servos, motors, batteries, etc.

7. *Actuator messages* specify the interface with hardware actuator controllers. The actuators are servos and propeller motor.

This layered control and sensing approach enables modular development of applications. Software components can run in logical isolation, interfacing with other modules merely through the exchange of IMC messages. The control infrastructure for autonomous vehicles implemented on-board, using the DUNE framework, that enables message exchange using a message bus abstraction, and provides transport mechanisms for external communications. Vehicles can be monitored and controlled externally using Neptus consoles. Networking of vehicles and consoles, is enabled through traditional IP-based

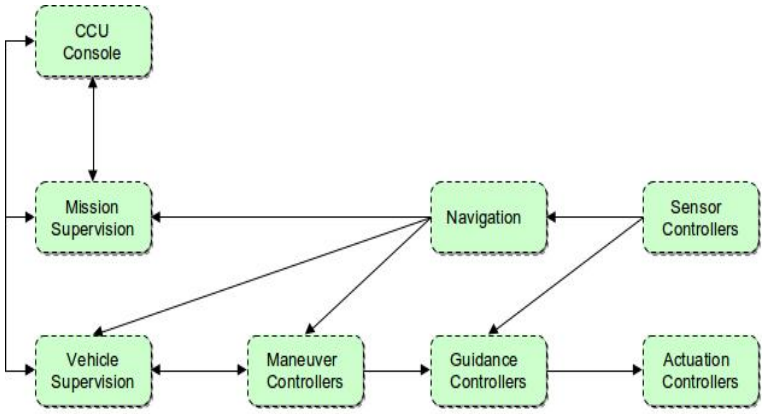

Figure 1: Message flow in IMC for different systems and their interaction

communication-mechanisms, like raw UDP or TCP sockets, or using ubiquitous GSM/GPRS communications.

3 Dune

To abstract vehicle hardware and specificities, we use the DUNE onboard software. DUNE is composed of a set of asynchronous tasks that communicate between each other using the IMC protocol. Some of these tasks may be coupled to a sensor (publishing messages) or actuator (consuming messages) or may be also implementing a high-level navigation or control algorithm. All vehicles run the same software only with an adapted configuration which states the tasks that should execute (possibly multiple instances) and also configuration for individual tasks.

4 Command and control system

We built the Neptus command and control infrastructure in a modular way with a typical mission life cycle in mind. We divide the life cycle into planning, executing, review and analysis, and dissemination. Despite the reuse of several modules, each phase has a dedicated graphical interface focusing on what tasks are more important in that particular phase.

4.1 Mission Planning

The planning of a mission encompasses the definition of a virtual representation of the mission site, choosing the area for operation, and creating mission plans that the vehicle(s) will execute autonomously. Neptus includes the Mission Planner application, which is exclusively targeted for these tasks. In the Neptus Mission Planner, the map editor interface allows the definition of world maps in both 2D and 3D visualizations. Maps are composed of several geo-referenced elements: marks to denote points of

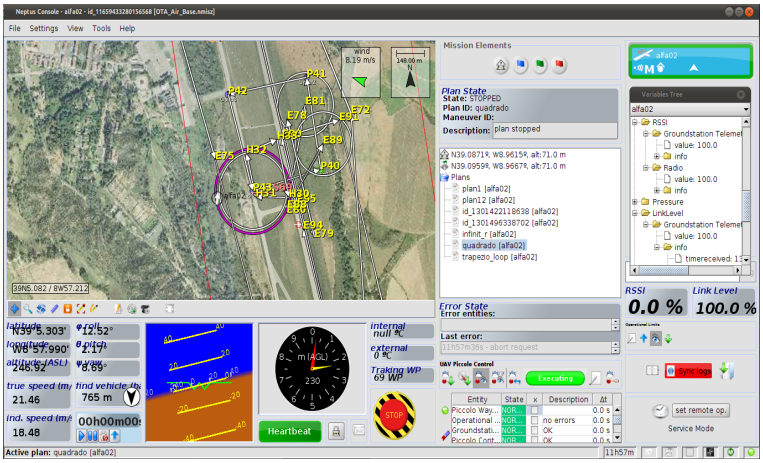

Figure 2: Neptus Operational Console

interest, geometric figures, surface images (with optional elevation map), generic 3D models, and paths (composed of lines). Neptus mission plans are defined as a set of maneuvers that are connected through conditional transitions, forming a finite state machine or (if one considers the continuous dynamics of the maneuver controllers) a hybrid automaton. Mission plans can be edited directly as its graph representation or, for usability purposes, they can be drawn directly on top of the map previously created. As can be seen in Figure 2, to insert a new maneuver at a plan, the user may click on the map where he wants to position the maneuver and a popup menu will appear displaying the possible choices of maneuvers available for that particular vehicle. After a maneuver is added to the mission, it can be dragged to other locations in the map or its parameters can also be set by double clicking the maneuver. For increased usability, the user may also set parameters for all maneuvers at a time (when there are shared parameter types) and translate/rotate the plan in its entirety.

Due to the numerous field-tests made with Neptus, we noticed that some of the plans carried out by autonomous vehicles are very similar (sanity tests, regression tests, area scanning, etc). As a result, we added the possibility of defining mission templates through a scripting language based on JavaScript. This scripting language allows the definition of template parameters and the generation of plans. XML (eXtensible Markup Language, VI) is the format of choice to store data in Neptus. For convenience (file size, portability) all maps and mission plans are stored as a single Zip file.

4.2 Mission Execution

The Neptus Console application is the main interface for vehicle interaction and real-time monitoring, as depicted in Figure 2. This application allows the establishment of connections to one or more vehicles, display incoming data through interface plugins and control vehicles behavior by sending messages (also through visual plugins).

In order to mitigate different communication interfaces and protocols we constructed the data feed to Neptus in a flexible way. The Communications Manager is responsible for the I/O to and from

the systems. In terms of communication protocols, we set UDP to be the default data I/O, but NDDS (Network Data Delivery System), GSM/GPRS, acoustics, TCP, and HTTP are also available with DTN (Delay-Tolerant Networks) underway. The IMC message protocol is the main data-entry protocol used in Neptus, but to be more flexible we define a XML Schema (XSD, VI) in which IMC can be defined. The resulting XML is then dynamically loaded in Neptus making IMC messages known. The same can be done for other message protocols like STANAG 4586 [9]. The complete data received is then available in a shared data environment VI and then the various console components can subscribe to this data and update themselves accordingly. Console components are visual (or hidden) elements that display system data and can be added to Neptus under the form of software plugins. To further improve flexibility, the console layout can be edited allowing adding, removing, and dragging of any console components. Components have parameters that, for instance, allow the user to select which variable to display or which vehicle to monitor. Moreover, the consoles provide an alarm logic to which visual components can deliver events. The alarm logic logs all events and warns the user in case any fault has been reported changing the error state of the console. Warnings can be reported as popup messages or spoken phrases. The console layout (plugin locations and respective parameters) can be stored in an XML file and the resulting console can then be associated to a vehicle class. Neptus design supports concurrent operations. Vehicles, operators, and operator consoles come and go. Operators are able to plan and supervise missions concurrently. Additional consoles can be built and installed on the fly to display mission related data over a network.

4.3 Mission Review and Analysis

After a mission is completed, operators are interested in accessing and revising the vehicles data. For this, there exists the Neptus Mission Review & Analysis (MRA) application (Figure 3). Again, recurring to a plugin architecture, this software application provides various visualizations over a set of log files produced in a single mission. By default, there are a number of predefined visualizations that are automatically generated upon opening a mission log. However, the user may also select a number of fields to inspect and generate new plots with a couple of mouse clicks.

Another useful visualization is the mission replay of the logged data. This comes in two flavors: one simpler mode displays the evolution of the estimated state of the vehicle in a 2D or 3D renderer; and the other consists on taking all the logged messages and feeding them to the Neptus communications interface. As a result, by opening the correct mission in an operational console, it is possible to follow the evolution of the mission as the messages were being logged in the vehicle. The replay visualizations give access to the mission timeline and it is possible to pause, advance, rewind, and speed up or slow down the replay. MRA also allows the generation of a mission execution report with the most common plots and statistics in the form of a PDF document. Another output of this tool is a partial automated elevation map generation for map integration.

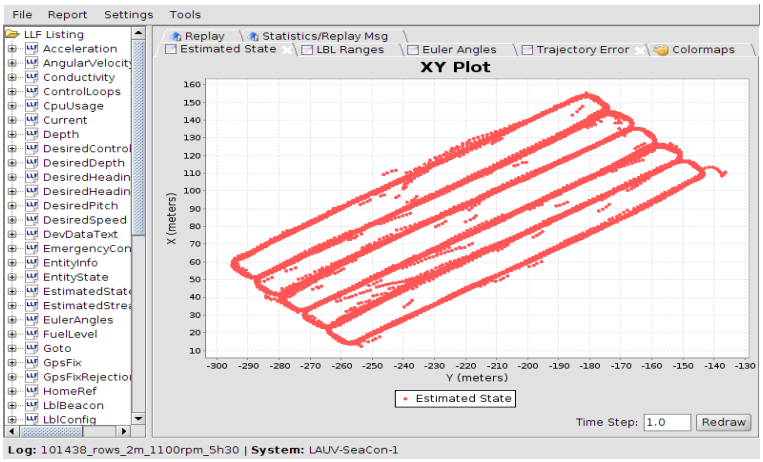

Figure 3: Neptus MRA

4.4 Disseminating Results

Dissemination plays an important role in our infrastructure because it allows external applications to access real-time data feeds of our sensors and also to integrate external data in our system. We use mostly web technologies (HTTP, REST, ATOM, XML/XSLT) for dissemination. The implementation involved embedding Mortbays Jetty application server (compatible with the Servlet 2.0 API) in Neptus. We developed various entry points for this server, both for data publication and querying. Different data access modes (file types, queries) correspond to different URLs. For instance, there exists URLs that display data in KML format (allowing visualization in Google Earth (see Figure 2)), and other URLs that generate plots, HTML tables, etc.

5 Integrating Piccolo with Neptus/DUNE

With the presence of commerial autopilot and ground station softwares and with the builin user confidence in operating these systems, it is difficult to convince the user not to use the system and replace it with a more advanced system. In order to have compatibility with the exisiting system before replacing it, we have developed an integration methodology and the required software. As an initial step we integrate the DUNE, IMC and Neptus sub-systems with the exisiting Piccolo aupilot system with its ground station.

A first implementation of the Piccolo controller in Neptus and DUNE was made on top of Piccolo waypoints interface. To allow a smooth transition for both the operating team and the newly implemented interface, a protocol was drafted in order to allow handover of the UAVs between PCC and Neptus operators. This protocol is a contract between Neptus/DUNE and the PCC (Piccolo Command Center) operator so none of the parts touch each other designated waypoints.

The 100 available waypoints are divided in 3 groups: the handover waypoints; the plan waypoints; and all the remaining (designated as external). The handover waypoints serve, as the name indicates, to

handover the vehicle between PCC and Neptus operators. The PCC operator creates here a safe piccolo plan and he is responsible for maintaining these waypoints. The Neptus operator don't change any of these waypoints for a safe transition of UAV command. He only sends the UAV to one of the waypoints and takes it out of them to take full control of the vehicle. The plan waypoints are to be used by the Neptus/DUNE for the execution of plans and must not be touch by the PCC operator. As seen earlier, these plans are defined in a different way than in Piccolo. So DUNE translates all executing plan to a set of waypoints and send them to Piccolo. This also makes the PCC operator aware of what to expect from the vehicle as it executes Neptus/DUNE plans. From this period, the interaction similar to that of operating the vehicle without piccolo and its ground station. At any time the PCC can trigger any other waypoint. At this point the execution of the plan is aborted and the PCC is responsible for the vehicle. Any other waypoint is considered external and of full PCC control.

6 Conclusions and future work

We presented tools that can assit UAV operations to be safe and efficient. The IMC allows easier integration of various standards and different communication modes. The Neptus allows 2D and 3D views of the world and has all the components to conduct a full mission cycle. Dune allows monitoring and execution of various tasks and is open source software. Further, we plan to integrate some of the AI techniquies into the mission planning stage for better degining of mission and also for situational awareness.

REFERENCES

[1] E. Semsch, M. Jakob, D. Pavlicek, and Michal Pechoucek: Autonomous UAV surveillance in complex urban environments, *In Proc. of the IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*,Washington, DC, 2009, pp. 82-85.

[2] B. Bethke, J.P. How, and J. Vian: Group health management of UAV teams with applications to persistent surveillance,*Proc. of the American Control Conference*, Jun 2008, Seattle, WA, pp.3145-3150.

[3] S.R. Herwitz, L.F. Johnson, S.E. Dunagan, R.G. Higgins, D.V. Sullivan, J. Zheng, B.M. Lobitz, J.G. Leung, B.A. Gallmeyer, M. Aoyagi, R.E. Slye and J.A. Brass: Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support, *Computers and Electronics in Agriculture*, Vol. 44, No. 1, July 2004, pp. 49-61.

[4] A. Simpson, T. Stombaugh, L. Wells and J. Jacob : Imaging techniques and applications for UAVs in agriculture, *American Society of Agricultural and Biological Engineers Annual Meeting*, St. Joseph, Michigan, 2003, #031105.

[5] www.cloudcaptech.com

[6] www.micropilot.com

[7] procerusuav.com

[8] Martins, R.; Dias, P.S.; Marques, E.R.B.; Pinto, J.; Sousa, J.B.; Pereira, F.L.; , "IMC: A communication protocol for networked vehicles and sensors," OCEANS 2009 - EUROPE , vol., no., pp.1-6, 11-14 May 2009.

[9] STANAG 4586 Second Edition, Standard Interfaces of UAV Control System (UCS) for NATO UAV Interoperability, *NATO Standardization*, November 2007.

[10] Open JAUS, www.openjaus.com.

[11] R. P. Stokey, L.E. Freitag, and M.D. Grund: A compact control language for AUV acoustic communication,*In proc. of the OCEANS Conference*, Brest, FRANCE, JUN 2005, pp. 11331137.

[12] J. Pinto, P.S. Dias, G.M. Goncalves, R. Goncalves, E. Marques, J.B. Sousa, and F.L. Pereira: Neptus a framework to support a mission life cycle, *IFAC Conference on Manoeuvring and Control of Marine Craft*, Lisboa, Portugal, Sept 2006.