**Proceedings of the 8th IFAC International Conference on Manoeuvring and Control of Marine Craft
September 16-18, 2009, Guarujá (SP), Brazil**

Integrated Tele-Operation & Mission Control: preliminary experiments with a small USV

M. Bibuli [*] **G. Bruzzone** [*] **M. Caccia** [*]

[*] *Consiglio Nazionale delle Ricerche
Istituto di Studi sui Sistemi Intelligenti per l'Automazione
Via De Marini 6, 16149 Genova, Italy
(e-mail: {marco,gabry,max}@ge.issia.cnr.it)*

Abstract: Preliminary experiments, carried out with the Charlie USV, of an automatic mission controller able to alternate the execution of autonomous and human tele-operated activities are presented. Simulative and field trials validates the control architecture enabling the management of standard USV missions. In particular, the interconnections and interactions between the basic modules of the control architecture, i.e. mission controller, execution controller, execution level, path planner, and execution monitor, are verified. The definition of a mission control language as well as the configuration of different missions in suitable files is left to further developments.

Keywords: Mission control, unmanned surface vehicles, Petri net, autonomous robots, mobile robots.

1. INTRODUCTION

Recent progresses in the functionalities, performance and reliability of Unmanned Marine Vehicles (UMVs) increase the demand of mission control systems able to online replan the vehicle activities, guaranteeing system's consistency, on the basis of the perceived operating environment and robot state. Since the development of fully automatic situation awareness modules for high uncertain environment is beyond the state-of-the-art, the human operator can play the role of intelligent sub-systems able to interrupt or modify the mission evolution. Thus, this research addresses the issue of designing and implementing mission control systems for remotely supervised and tele-operated UMVs with high degree of autonomy. Many approaches to the Mission Control problem have been investigated in recent years; some works are founded on systems which explicitly track the state of evolution of the mission plan, like Finite State Machines, as in Poole (2000), and Petri Nets, as in Oliveira et al. (1998) and Chang et al. (2004). In Turner (1995), the schema-based Orca architecture is presented; Orca is a context-sensitive adaptive problem solver, that uses procedural schemas, which are like hierarchical plans, to control its actions and using contextual schemas to ensure that its behavior is tailored to its problem-solving situation. The MOOS architecture, Newman (2002), contains a set of libraries and executables which are designed to control a land or underwater robot; it works using independent objects and a blackboard where all the objects read and write messages. Application of custom programming languages and scripts for mission plan definition are pointed out in Pebody (2007) and in Palomeras et al. (2008) a system for the automatic translation of a mission control language into a Petri net, describing the mission thread of execution, is developed. The main difficulty, when facing mission control problem, and rarely treated in practical applications, is the handling of forced interruptions and restores of the mission execution; mission control systems are often considered as mere plan executors, which minimize the human interventions during UMV operations. Instead, the main idea, addressed in this paper, is to emphasize the interaction between human interventions and automatic operations execution. In particular, in many operational cases, missions of unmanned marine vehicles (UMVs) consist of a mix of repetitive autonomous tasks and human interventions to handle unexpected situations and/or strict interactions with the operating environment and other agents therein. For instance, in the case of the Charlie USV, used as a testbed in this research, anti-collision, docking and online detection of sites of interest require remote human intervention. Thus, the presence of the human operator in the perception-control loop increases the situation awareness capabilities of the whole system: when human controlled, the robot can handle high uncertainty conditions at the cost of sometimes unpredictable actions, being the attitude to online mission replanning typical of human beings.
In practice, conventional missions of the Charlie USV for harbour/coastal monitoring and robotic research experiments present the following structure:

- **human controlled deployment and compass calibration**: the USV is piloted away from the peer to a free area by the human operator and then execute a couple of slow rotations for auto-calibrating its flux gate compass; the auto-calibration manoeuvre can be stopped and restarted by the human operator;
- **self-identification of dynamics**: the USV, under the supervision of the human operator, executes a set of suitable manoeuvres for identifying its dynamics; the completion of this procedure is detected automatically;

- **main tasks**: the USV autonomously executes its main tasks which, from the point of view of navigation, guidance and control, is seen as a sequence of basic manoeuvres such as *move at constant heading*, *move at constant course*, *follow a straight line*, *follow a generic path*, *move through a sequence of way-points*; the human operator, on the basis of his/her perception of the operating environment, can take and release the control of the vehicle and modify the planned manoeuvres at any time;
- **human controlled recovery**: the USV is piloted to the peer by the human operator.

In this context, the human operator plays the role of intelligent sub-systems able to interrupt or modify the mission evolution and the mission controller has to be able to support these requests, guaranteeing system's consistency. In Section 2 a brief description of the control system architecture, developed by CNR-ISSIA, is reported, presenting the already existing modules and the on-going upgrades towards the mission control. Sections 3 and 4 describe respectively events signaling and management, and main mission controller components. A basic application of the proposed approach to mission control is described in Section 5, reporting experimental results in Section 6.

2. CONTROL SYSTEM ARCHITECTURE

To the aim of obtaining a control system architecture characterized by a high application flexibility, for the most different operative conditions, the proposed architecture has been developed following a modular approach, where all the various components of the system are completely independent one to each other. Communications among such components are provided through the dispatch and reception of proper messages, over a set of communication queues. In such a way the independency, also from the point of view of execution scheduling, of the modules composing the architecture is provided; in fact being not required any intermodule synchronization, the execution sample time of all modules are completely uncoupled. The

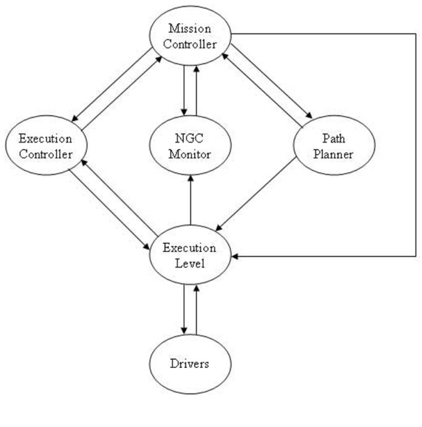

Fig. 1. Control System Architecture.

proposed control architecture, reported in Fig. 1, presents at the lowest layer the *Driver* modules; these modules, related one to each sensing or actuation device mounted on the vehicle, execute Input/Output operations on such devices, defining in this way an interface between control software and hardware architecture. The driver module set directly communicates with the *Execution Level* module; such module is in charge of executing navigation, guidance and control (NGC) tasks, acquiring sensing measurements from drivers and generating actuation commands. The *Execution Level* module contains two sets of variables, defined as *Control Variables* and *Estimation Variables*, which different tasks in execution share, reading or updating such values. In particular, the *Execution Level* is composed by two sub-systems, one defined as *NGC Executor* in charge of the execution of actual NGC tasks, and the other one as *Plant Manager* which acts as a logic interface between drivers modules and NGC tasks and variables layer. Different NGC and estimation tasks activations are managed by the so called *Execution Controller* module. From remote control interface, the user can select the desired tasks to be executed; the *Execution Controller* provides the automatic activation of required tasks. Such automatic selection guarantees consistency with a set of constraints on the task activation/deactivation sequences which are related to the topological structure of the *Execution Level*, i.e. the graph representing the I/O relationship between the tasks, rather than the semantics of each task. On this basis, the *Execution Controller* module can automatically detect the conflicts between tasks, reconfiguring the *Execution Level* using only the information embedded in the task I/O relationship, contained in a set of configuration files loaded during start-up phase, without any additional knowledge of their semantics. The *Execution Controller* capability of automatically activate user-chosen tasks is based on the definition and on-line reconfiguration of a Petri net describing the I/O relationships of execution level tasks (reader can refer to Caccia and Bruzzone (2007) for further details). On the other side, the *NGC Monitor* is responsible of the generation of events related to the semantics of the tasks execution; in such a way the *NGC Monitor*, with the *Execution Controller*, completes the interface layer between continuous-time driven software modules and event-driven components. In particular the *NGC Monitor* signals specifics conditions related to the evolution of continuous time variables. A generic mission description usually involves the definition of specific events the mission controller has to wait for, such as "variable value has been reached", "controller converges", "point has been reached". Thus, the *NGC monitor* implements a library of monitors of the execution variable values corresponding to conditions as the above-mentioned ones. The architecture is completed with the integration of a module able to online re-plan the vehicle motion. The *Path Planner* is in charge of computing and handling a collision free path to a desired location with additional constraints on way-points, local tangent and curvature, etc. For the sake of generality, a path is specified as a sequence of points with the respective tangent and curvature. The *Path Planner* supplies the guidance modules with the coordinates, orientation, curvature of the actual point of the path that has to be tracked. Currently, only functions computing basic paths such as straight lines, splines for N way-points and circumference arcs have been implemented. The latest upgrade to the *Path Planner* module regarded the implementation of an on-line path generation, on the basis of position data received from a moving master vehicle. This was made with the aim of providing a *master-slave* type motion coordination, i.e. a *master* vehicle (not necessary