FUNCTIONAL MODULES FOR INTERMIXED PLANNING AND EXECUTION OF AN OBSERVATION MISSION

Elodie CHANTHERY
Dr. Magali BARBIER
Department of Systems Control and Flight Dynamics (DCSD)
Office National d'Etudes et de Recherches Aérospatiales (ONERA)
BP 4025 - 2, av. Edouard Belin - 31055 Toulouse CEDEX 4 - FRANCE
Tel + 33 5 62252561
Fax + 33 5 62252564
[Elodie.Chanthery, Magali.Barbier]@cert.fr

ABSTRACT

This paper presents an implementation of functional modules for intermixed planning and execution of an observation mission for an UAV. The mission is a mean altitude navigation in a 3D partially known and dynamic environment. The mission plan includes operations such as area survey, outline following and object search, defined off-line by an operator. It must satisfy constraints due to the vehicle, the environment and the mission.
The ProCoSA tool is used to specify the UAV desired behaviour through hierarchical Petri nets, to check the behaviour of the UAV during the execution and to communicate with computational sub-systems of the on-board system architecture. One highest level Petri net describes the observation mission : first off-line planning and take-off, then navigation to mission point and operation achievement if it exists, finally return-to-base and landing. Secondary generic Petri nets specify the different phases of the mission execution. System replanning capabilities are activated on-line if current trajectory or itinerary is no more consistent with prediction. The main external sub-system optimises the 2D itinerary and 3D associated trajectories given the mission and its constraints. A second sub-system computes the guidance sent to the modelled UAV. A third one centralises dynamic information.
First simulation tests on a 200km×140km zone with all types of operations and a simplified model of a mean altitude UAV highlight the feasibility of the proposed structure and the reactive capabilities of intermixed planning and execution architecture in nominal and degraded situation.

BIOGRAPHIES

Elodie CHANTHERY
Electrical and Automatic Control Engineer, at National Polytechnic Institute of Engineering in Electrotechnology, Electronics, Computer Science, Hydraulics and Telecommunications (ENSEEIHT). Specialisation in Automatics and Industrial Computer Science.
PhD Student at ONERA/DCSD (System Control and Flight Dynamics Department of the French National Aerospace Research Establishment) in Toulouse, on "Mission Planning for an Unmanned Air Vehicle System".

Dr. Magali BARBIER
Research Engineer on Automatic Control at ONERA/DCSD.
Teaching activities (Closed Loop Control, Optimisation, Operational Research, Discrete Events Systems, Optimal Control) in French High Schools.
Research activities :
1986-1990 PhD on "Modelling and Control of Assembly Systems"
1990-1998 Dynamic Route Guidance Systems and Simulation work in Transportation field - European DRIVE and PROMETHEUS projects
1999-2002 Intermixed Planning and Execution for an Unmanned Underwater Vehicle, Advanced - Surface Movement Guidance and Control System for Airport, ProCoSA specialist

1. Introduction

Mission management of an autonomous vehicle in a partially known, dynamic and unsafe environment is a complex real-time problem. Two ways allow the reduction of system complexity. In their distributed hierarchical system architecture, H. Jin Kim *and al.*[1] divide the control of the vehicle into different layers of abstraction ; in compositional methods proposed by B. Sinopoli *and al.*[2], the problem is decomposed into a sequence of smaller problems of manageable complexity. That functional hierarchical structure is widespread but precise implementation has not yet been described in the literature.
This paper presents such an implementation of functional modules (Figure 1) developed for intermixed planning and execution of an observation mission for an Unmanned Air Vehicle (UAV). The behaviour of the vehicle is modelled hierarchically and the execution control follows this more and more detailed structure. Computational part is distributed in independent sub-systems.

The architecture determines the UAV's capabilities to achieve its mission and to react to events. Therefore, it should possess some essential characteristics. The problem is comparable to design of mobile robot control systems given by Laffey *and al.*[3], then extended by Ingrand *and al.* [4], with expected criteria :

- Asynchronous event handling : the architecture has to deal with events that do not occur at designed times, or with data that are not sent to the system at regular intervals of time.
- Programmability : the system should not be designed for a specific environment or for achieving a single task. It should be able to achieve multiple tasks in different situations. Its functions should be easily combined according to the task to be executed.
- Reactivity : it is necessary that the system be able to provide guaranteed reaction and response times. Important events must be noticed in a timely manner and appropriate action have to be taken before it is too late. The system should handle multiple problems concurrently.
- Autonomy and adaptability : the system should be able to carry out its actions, to refine or modify the task and its own behaviour according to its goals and to the environment.
- Extensibility : integration of new functions and definition of new tasks should be easy.

The mission of the UAV, the assumptions on the environment and on observation areas and the data to be defined by the operator are detailed in section 2. In the same section, we also describe the possible operations, then the constraints and the criterion taken into account during the optimisation of the mission.
The choice of an on-board structure intermixing planning and execution control is justified and the implemented architecture is described in section 3.
The UAV desired behaviour during the mission is specified in hierarchical Petri nets (section 4) : the high level mission is modelled in one net, secondary Petri nets specifying the detailed UAV behaviour and planning and replanning capabilities of the proposed control architecture.
Embedded sub-systems executing the computational part of the on-board system architecture are defined in section 5 : a planning sub-system computes an optimal plan given the mission and its constraints ; a guidance sub-system achieves the UAV itinerary following ; dynamic data are centralised in a third sub-system named SOCRATE.
Section 6 presents numerical experiments by simulation with scenarios and results and highlights the advantages of the implemented architecture.
Section 7 concludes on this work and proposes new ways of research.

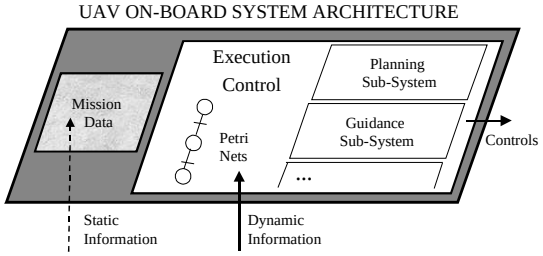

Figure 1 – Functional Modules

2. UAV Mission

2.1 Introduction

UAVs can execute several types of mission at different altitudes. In this work, the objective selected for one vehicle is to observe from a mean altitude several local areas in an unsafe and partially known environment ; take-off and landing phases are performed in safe ground stations. Amongst navigation constraints, forbidden areas limit the navigation of the vehicle and specific procedures are required for the crossing between safe and unsafe zones.

In this work, it is assumed that during the execution of the mission, there is no communication between the UAV and either a ground operator or another vehicle (unmanned or not). After the take-off phase, the UAV is then totally autonomous until the waited landing phase. This hypothesis is not restrictive from an architecture point of view and co-operative modules could later be added.

The mission is thus defined as a mean altitude autonomous navigation from a start mission point (initial position) to an end mission point (final position) in a 3D environment (Figure 2). The mission plan of the UAV is a sequence of local areas restricted to mission points from a navigation point of view.

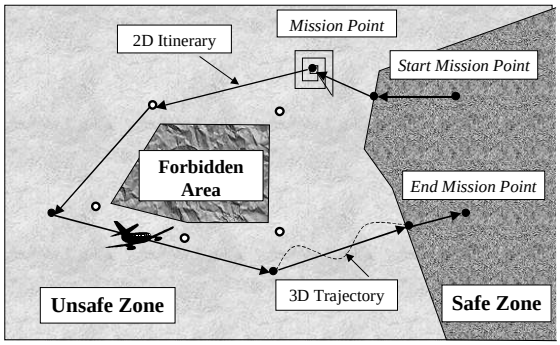

Figure 2 – Observation Mission

When arriving on a mission point, the UAV proceeds according to predefined types of operation :

- area survey,
- outline following, or
- static object search.

In order to bypass forbidden areas, particular points are defined. This type of points together with crossing points can be included in the plan.

To define the mission, the operator selects then off-line a set of mission points in the mission zone, bypass points to avoid forbidden areas and two crossing points to enter and exit from the unsafe zone. If a problem occurs during the execution of the mission, priority levels assigned to mission points allow the on-board system to update the list of areas to observe.

2.2 Operations

For each mission point, the operator chooses off-line the operations that the UAV has to proceed. Currently, all operations are realised at steady distance from the ground ; we assume there is no constraint due to payload. Nevertheless, the wind can limit the achievement of an operation.

The operator can choose amongst four types of area survey (Figure 3) with one or two scans and a sweeping or a centred survey. He specifies thus the wanted type and the geometric points of the convex area.
Outline following is defined by a set of successive angular points, for example to describe a coast or a railway.
The static object search operation is a particular area survey centred on the position of the object (type 3 or 4). The operator defines the object simplified geometry and its expected position. When the object has been found, the operation stops and the mission goes on.

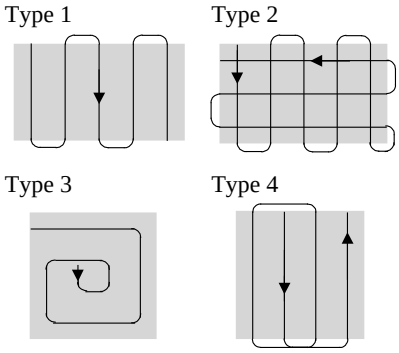

Figure 3 – Area Survey Operation Types

Given these operations, the choice was made to model them by the succession of trajectories and course changes.
For area survey and object search operations, the operative sequence, defined as the set of points of end of trajectory and end of course change, is computed at the beginning of the mission for all time intervals, to avoid possible long on-line computations. The main direction of the survey depends on the direction of the predicted wind if any. Those computations are valid on-line if the effective wind is consistent with the predicted one. Otherwise, a second computation taking into account the new wind constraints is performed on-line only for the time interval for which the UAV arrives in the operation area,.

The planning algorithm computes an itinerary composed of mission points, bypass points and crossing points, and from this itinerary point of view, the operation is only a duration. In order to reduce complexity, the vehicle comes back to the mission point location at the end of each operation.
The operator gives for each operation a maximum duration and the system controls the satisfaction of this constraint.

2.3 Constraints and Criterion

The above definition of the mission allows to formalise the problem in terms of constraints and criterion to optimise in the planning problem.

Constraints induced by the vehicle are modelled in the planning algorithm with either maximal/minimal values or physic equations :

- on-board fuel limits the autonomy,
- vehicle capabilities limit the speed between two extreme values,
- the existence of a relation between consumption and velocity has to be taken into account.

Constraints due to the environment of the mission are geometrically defined by the operator and verified during the itinerary computation :

- the mission zone, with unsafe and safe zones (with the definition of crossing points),
- forbidden areas in the unsafe zone (with the definition of bypass points).

The wind is considered in the itinerary and trajectory computations through the consumption versus velocity relation and it constraints the main direction of survey area operations.

The mission constraints are given by the operator for each mission point :

- 3D location (x, y, ground distance),
- earliest and latest start time,
- priority level,
- maximal duration,
- operation type and specific parameters (cf. 2.2).

Bypass points have the first constraint and crossing points have the two first ones.

The criterion considered is a weighted sum of the itinerary length and the fuel consumption. The planning problem is thus written as the minimisation of this criterion satisfying above constraints.
During the mission execution, constraints can evolve because of uncertainties related to itinerary following and operation performing. Punctual events can also degraded the optimal execution : internal vehicle failure, payload information, events from outside the vehicle… On-board replanner capabilities are so required to take into account these uncertainties.

3. Execution Control Architecture

A process is required to control on-line the execution of the mission of the UAV. During the autonomous navigation, the execution control is connected with the physical system and runs and stops the decision modules. These relations give to the execution control its reactive capabilities.

Recent studies on the links between the computation of plans and their executions make significant great strides, moved by the increasing number of practical applications. These studies bring into question the assumptions generally adopted in planning : a static environment, no failure. Indeed, new events can appear and invalidate the on-going plan ; the execution of planned actions can fail and this must be taken into account by the planner to adapt the current plan.
On the one side, probabilistic or conditional planning, like Markov Decision Processes [5] makes it possible to manage uncertainties concerning the environment state and the effects of the actions. Those approaches are mainly focused on the planning process.
On the other side, techniques and systems like Procedural Reasoning System [6], based on the notion of rational agent that can reason and plan under possibly stringent constraints on both time and information, were developed to supervise the execution of plans. These approaches do not make planning and fail when they meet new goals for which they do not have predefined plans.
Some works propose to gather planning and execution [7]. The result is mainly the concatenation of two already existing systems rather than a combination of planning and execution control.

Our approach proposes to use a structure based on the mix of a planning algorithm with an execution control tool named ProCoSA [8], developed for programming and execution monitoring of autonomous systems. Behaviour of the UAV during the mission is described by Petri nets [9], which are directed graphes with two kinds of nodes, called places and transitions. In ProCoSA, places represent the considered behaviour internal states and transitions indicate the phenomenons that change the behaviour execution state. The Petri nets can here be captured through a graphic user interface named EdiPet [10]. This interface is used for the off-line specification of the desired behaviour of the system in one or several Petri nets. The same interface can be used during the execution phase to check this behaviour as the current logical state of the mission is displayed during the process. When a communication link

exists with an operator, he can see in real time the updated display.
A complex automate, called the "Petri Player", runs the system in accordance with Petri nets in updating the marking. Computational tasks of the deliberative part of the vehicle such as the planning algorithm, but also guidance controls suited to the vehicle, are modelled as sub-systems of the on-board system architecture (Table 1). The Petri Player is implemented with a LISP interpreter, called tiny [11], useful for the implementation of the real-time interfaces between the Petri Player and the sub-systems.

Type of module	Implementation
Deliberative Reactive, numerical	Computational sub-systems
Reactive, logical	Petri Nets

Table 1 – Implementation of an Architecture with Deliberative and Reactive Part

An original functionality of ProCoSA regarding to Petri nets is to assign events and requests to transitions for the link between Petri nets and sub-systems (Figure 4). The Petri Player sends requests to the sub-systems and when receiving events in return, it updates the Petri nets marking. The different Petri nets are also internally connected using this protocol.

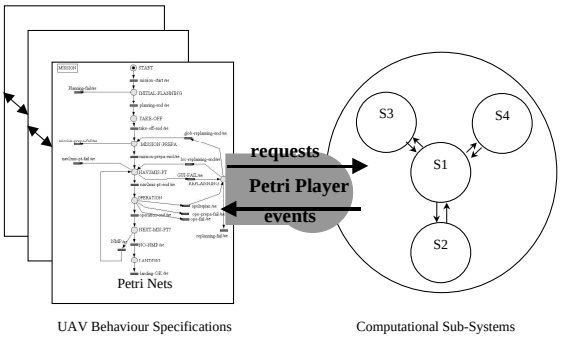

Figure 4 – Execution Control Architecture

ProCoSA is successfully used for the control of the mission of an Autonomous Underwater Vehicle in the French GESMA's project "Non Traditional Navigation Guidance and Control" [12][13].

4. Petri Nets Modules

4.1 Introduction

The UAV desired behaviour during the observation mission in nominal and degraded situations is thus detailed in Petri nets.
The behaviour of the UAV which objective is to observe specific areas can be viewed from a hierarchical point of view. Specification of Petri nets is so made by hierarchical tasks. An advantage of this architecture is to allow an easy integration of new tasks. Moreover, the understanding is easier for a possible operator in a ground station. The different phases of the mission, for example an operation execution at a mission point, are seen as macro-actions described with secondary Petri nets.

4.2 The Observation Mission

One Petri net (Figure 5) specifies the high level behaviour of the UAV from the start mission point to the end mission point. One marked place in this MISSION Petri net embodies the phase or macro-action in which the UAV can be.

Four places in this MISSION Petri net activate four navigation phases of the UAV : take-off, navigation to the next point (mission point, bypass point or crossing point), operation if it exists, return to the ground station and landing.
The other places detail the global planning and replanning requirements and the itinerary following with the loop {navigation to the next mission point and operation execution} until exhaustion of mission point.
Specific transitions and associated Petri nets manage the possible failures.

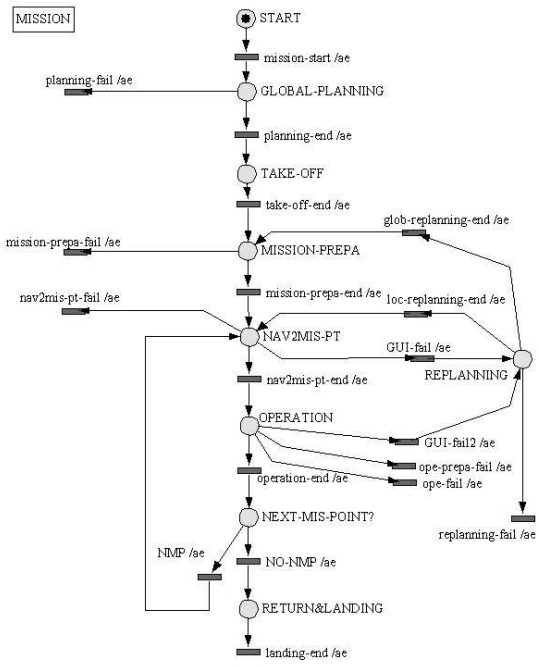

Figure 5 – High Level Mission Petri Net

The UAV behaviour as represented on Figure 6 is detailed in the above sections.

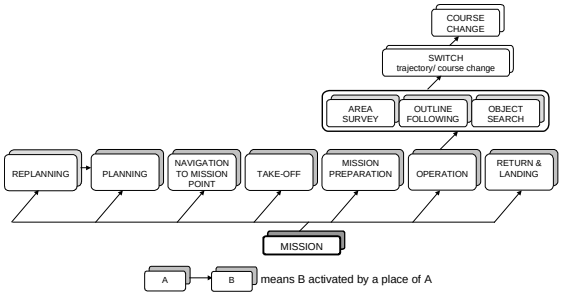

Figure 6 – Hierarchical Petri Nets

4.3 Detailed UAV Behaviour

The four navigation phases are detailed in secondary Petri nets ran from the MISSION Petri net :

- TAKE-OFF from the start mission point at the beginning of the mission,
- NAVIGATION TO MISSION POINT, to navigate to the next mission point according to the result of itinerary planning,
- OPERATION execution,
- RETURN to the end mission point and LANDING.

The OPERATION Petri net calls itself one Petri net according to the operation type (AREA SURVEY, OUTLINE FOLLOWING, OBJECT SEARCH). The on-line computation of the operative sequence by the planning sub-system is done in these Petri nets. Then each of them calls the specific Petri net SWITCH, which switches linear trajectory mode and course change mode. For the realisation of a course change, SWITCH calls upon the COURSE CHANGE network.
The separation of all operations in several Petri nets allows to add or remove some at will.

The MISSION-PREPARATION Petri net allows to position the vehicle towards the first point of the last computed itinerary with a course change.

4.4 Planning and Replanning Capabilities

Figure 5 and Figure 6 highlight the planning and replanning capabilities of the proposed intermixed planning and execution control architecture.
The two levels planning algorithm is exploited at this mission level (cf. section 5.2).

The GLOBAL-PLANNING place at the beginning of the mission calls the secondary Petri net PLANNING in which a request for computation of 2D itinerary and associated 3D trajectories is generated.
In case of failure during the navigation to the next mission point or during the execution of an operation, the REPLANNING place calls the secondary Petri net REPLANNING, which issues a request of local planning towards the next mission point (if the failure occurs during an operation, the specification is to give it up). If the local planning successes, the vehicle follows the navigation with the new trajectory. Otherwise, the PLANNING Petri net is called to perform the global itinerary computation.

5. Sub-Systems Modules

5.1 Introduction

The on-board system architecture includes several computational sub-systems, which contain algorithmic functions, either for deliberative work or specific tasks :

- The “PLANNING” sub-system aims at optimising itinerary and trajectories taking into account constraints due to the vehicle, the environment and the mission.
- The “GUIDANCE” sub-system computes the guidance controls sent to the UAV (commanded speed, next point location, commanded heading) taking into account the resulting trajectories given by the PLANNING sub-system and the flight dynamic characteristics of the vehicle.
- The “SOCRATE” sub-system centralises all information necessary to sub-systems in order to simplify communications within the on-board architecture.

These sub-systems implemented in C language are described in the following sections.

5.2 Planning

The planning sub-system provides an optimal plan, e.g. itinerary and associated trajectories, given the mission defined by the operator (cf. section 2). The itinerary contains the mission points, crossing points and eventually bypass points defined to avoid forbidden areas.
For real time implementation, the planning algorithm is separated in two levels : a first 2D computation, then a 3D one. Constraints defined in section 2.3 are taken into account either during the first 2D itinerary computation level, or during the 3D trajectory computation level.

First computation is made at the beginning of the mission during the off-line global planning. When beliefs about the UAV state or its environment change, on-line computation gives either a new trajectory to the next point of the current itinerary or a new itinerary and associated trajectories.

Constraints considered in the 2D planning are the supplied on-board fuel, knowing that energy consumption of the vehicle depends on its speed, the respect of the time constraints for each mission point (earliest and latest start times) and forbidden areas defined in the unsafe zone. Itinerary is computed in four steps :

1. A cost matrix is built between each pair of points of the {mission points, bypass points, crossing points} set. The calculated cost is the weighted sum of the itinerary length and the fuel consumption.
2. A Dijkstra's algorithm determines the optimal itinerary between each pair in the {mission points, crossing points} set considering cost matrix.
3. A Little's algorithm is performed on the {mission points, crossing points} set in order to have a list of hamiltonian itineraries with increasing costs.
4. An evaluation determines the first itinerary, which satisfies time and supplied on-board fuel constraints.

The planning sub-system also computes 3D trajectories between each pair of the optimal 2D itinerary points.
As constraints considered in the 3D planning are the respect of the vehicle physical limits, the obtained trajectory is linear. The trajectory is thus a sequence of relatively nearby 3D points, with a speed profile, nominal dates of passage and expected autonomy.
This 3D algorithm also computes the trajectory part of operative sequences.

5.3 Guidance

The guidance sub-system achieves the UAV itinerary following. It computes the orders to be sent to the vehicle.
Calculations are made according to the 3D points specifications provided by the planning sub-system. Orders contain either speed and heading or speed and 3D point coordinates the UAV must reach.

5.4 Data Centralisation

Communication between Petri nets and sub-systems through events and requests is dealt by the Petri Player of ProCoSA (cf. Section 3). Sub-systems are independent tasks but they punctually use dynamic information like the current state, the point the UAV must reach or the operation type to execute. One single communication sub-system named SOCRATE centralises all these data and sends them to the Petri nets on requests (Figure 7). Thus, communication between SOCRATE and another sub-system is always controlled by the Petri Player.
For real time requirements, the UAV is directly connected to the guidance sub-system. This sub-system periodically receives the current state (location, speed, remaining fuel) from the UAV and sends it to SOCRATE.
In case of planning request, the planning sub-system requests to SOCRATE the current state, the actual mission point and the next one. At the end of the computation, it sends to SOCRATE the itinerary and the trajectories.
For some specific transitions, SOCRATE emits towards Petri nets information, for example on the point type (mission, crossing, bypass point), the operation type associated to a mission point…

Data centralisation gives a better visibility to the on-board system architecture. Moreover, the independence of the different sub-systems is very useful for re-use or architectural re-organisation.

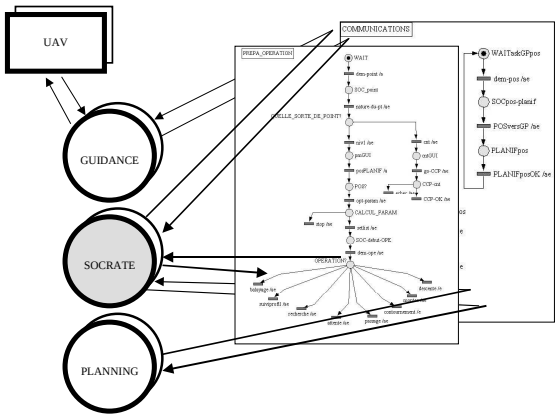

Figure 7 – Data Centralisation

6. Numerical Experiments

6.1 Mission Test

The objectives of simulations are :
- to verify the feasibility of the proposed architecture,
- to highlight the on-line replanning capabilities of the implemented architecture and thereby the efficiency of intermixed planning and execution architecture for an UAV.
- to show the reactive capabilities of the UAV regarding to awaited criteria for this type of architecture (cf. section 1).

An observation mission has been simulated on a 200km×140km zone (Figure 8). Three mission points with three different operations are defined. The mission zone includes two forbidden areas in the unsafe zone. Four bypass points are defined in order to avoid them and two crossing points are

chosen to cross the safe/unsafe zones border. All mission points have the same priority level.
It is assumed that this mission is executed without wind. The trajectory and the itinerary of the UAV are recorded each minute in order to verify the guidance controls. The predicted mission lasts about 30 hours. In this work, guidance controls are exactly applied by the simplified UAV model.

6.2 Experiments & Results

In the first simulation, the architecture is tested in a nominal situation, that is with no failure. Figure 9 illustrates the UAV flight executing the mission. The plan is optimal, computed in less than 5s. Forbidden areas are avoided, all the operations associated to mission points are correctly executed. The object is not found during the static object search operation. The mission lasts 22h 30min.

The second simulation (Figure 10) aims at testing the replanning requirements. A failure in the UAV is simulated and makes it impossible for the planning sub-system to correctly replan with all the remaining mission points. The planning sub-system updates the areas to observe and remove the outline following operation (the furthest one) from the mission points list. The mission is redefined and new optimal itinerary and trajectories are computed. The autonomous vehicle achieves its mission.

The feasibility of an on-board system architecture with replanning capabilities based on intermixed planning and execution is thus validated in a nominal situation and in a degraded one.

The two simulations presented in this paper exhibit the main characteristics of this type of architecture :

- Asynchronous event handling : events linked to various sensor data, requests, alarms can be sent asynchronously to all the modules. However, failure events have to be known and the system reaction must be specified and modelled through Petri nets.
- Programmability : the system could deal with other missions. Data files describing the environment, the vehicle and the mission can be modified with a wide set of parameters. The system is able to achieve multiple tasks like operations (area survey, outline following…) in nominal situation or in degraded one.
- Reactivity : the second simulation shows the system capability to deal with degraded situations. The system responds to events such alarms. The intermixed planning and execution of the mission allow a fast on-line replanning.
- Autonomy and adaptability : on-line replanning allows the system to refine or modify its tasks and its own behaviour according to the operator's off-line desires (priority level on each mission point and specified behaviour).
- Extensibility : the use of hierarchical Petri nets and data centralisation allows the easy integration of new functions and new sub-systems.

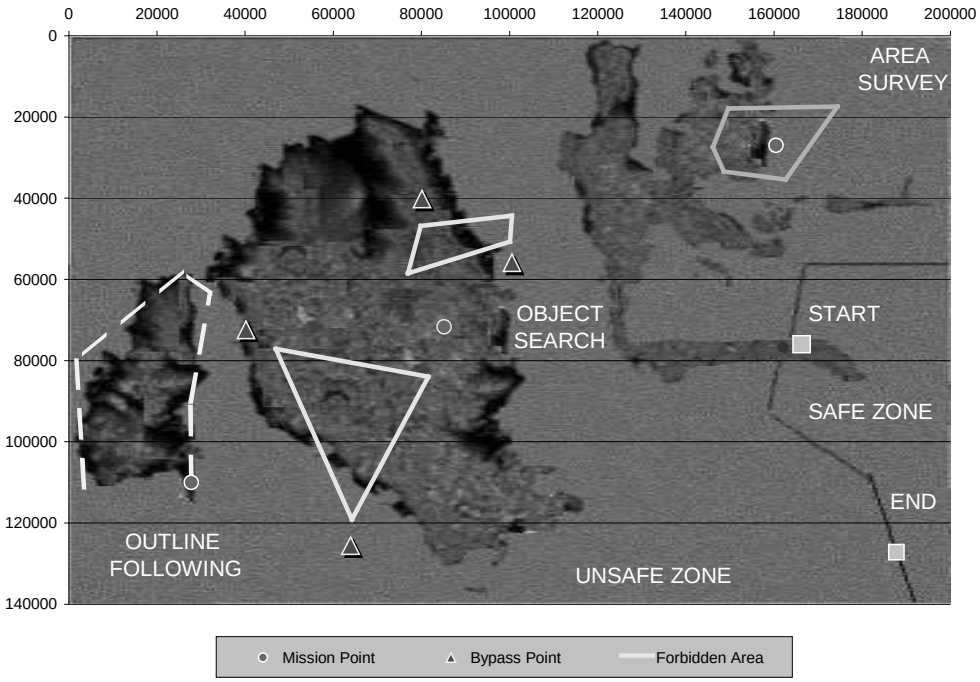

Figure 8 – Mission Test Map

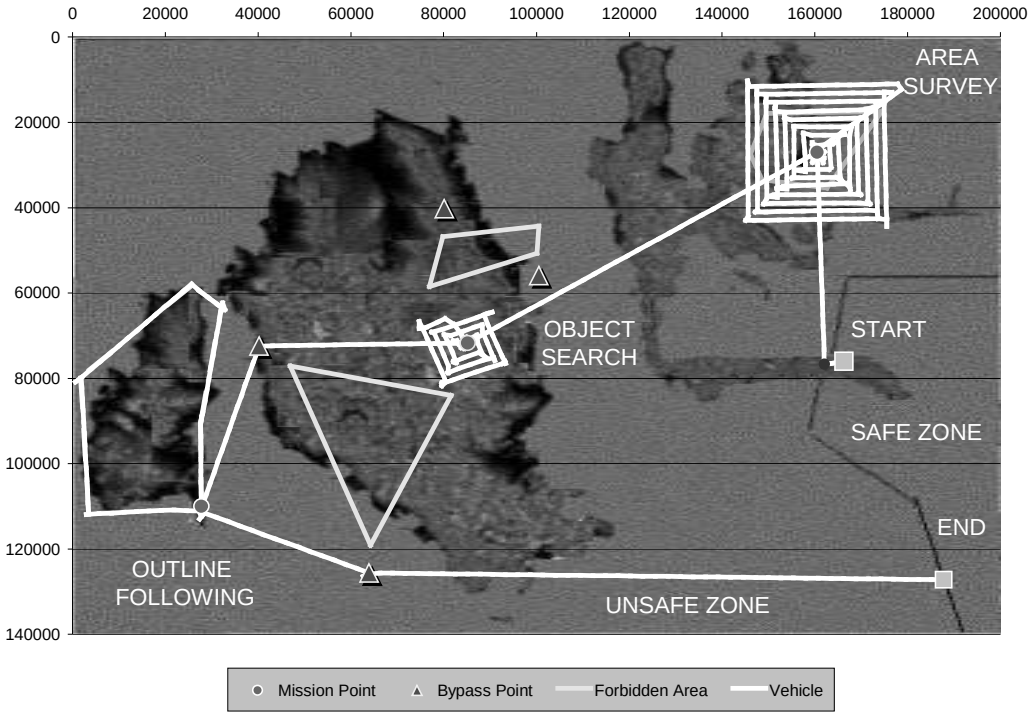

Figure 9 – Mission Execution in Nominal Situation

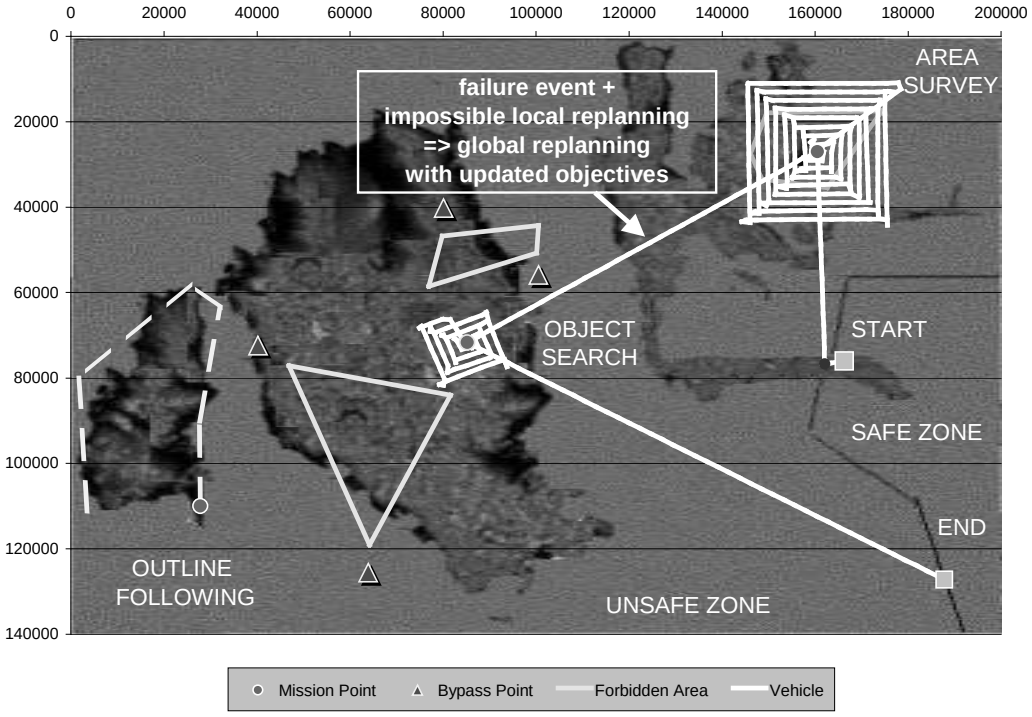

Figure 10 – Mission Execution in Degraded Situation

7. Conclusion & Future Work

This paper has proposed a hierarchical on-board architecture for the execution of an autonomous mission by an aerial vehicle. UAV mission and operations are specified with ProCoSA by using hierarchical Petri nets. The Petri net at the highest level specifies the execution of the different phases of the mission. Sub-systems compute the planning and the guidance. Communications are centralised. Simulations highlight the efficiency of intermixed planning and execution for the on-line control of an observation mission of an UAV.
The implemented architecture deals with asynchronous events and communication. It guarantees reactions consistent with specified behaviour. The system seems able to run in a totally autonomous way, modifying its behaviour according to a dynamic environment. ProCoSA tool allows the easy integration of new functions and sub-systems.

The next step in our work is the generalisation of the mission definition. We intend to adopt the notion of mission areas with in and out points rather than restricting the area to a single point. Consequently, planning algorithm will be improved so that the UAV does not come back to the mission point location at the end of an operation.
Moreover, the integration of an operation of waiting on hippodrome would permit to meet the constraints of the return crossing point without slowing down. It should also be useful in case of other mission type, for example target observation mission.
Concerning the constraints, the on-board fuel consumption is for the moment basically modelled. We intend to consider a more complex model. In the same way, uncertainties (meteorology and danger) are simplified in our present work. In the future, we could model these uncertainties by 3D probabilistic and dynamic maps.
The two levels planning algorithm could be brought into question. Other planning algorithms (multi-criteria algorithms for example), satisfying the time constraints and taking into account all above points, could be studied. They could optimise differently other criteria. Our architecture could be very useful as a test-bed for these new planning algorithms.

The model of the vehicle used in simulation is simplified. Guidance controls are exactly applied by the UAV model. In the future, a more precise model could be used, taking into account payload limitations, or uncertainties about the UAV location. The situation awareness of the overall mission, defined by Endsley [14] as *the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*, could be identified as a new sub-system of our architecture, dealing with the events received from the environment. Moreover, we assume that there is no communication between the UAV and either a ground operator or another vehicle. In the future, we intend to involve ground stations, in particular communication with one or several ground operators. Cooperation of other vehicles, on the ground or in the air, autonomous or not, is obviously conceivable.
Some work is on-going on ProCoSA tool to guaranty the on-board architecture time response and to allow the interruption of sub-system computation.
Finally, other simulations should be run in order to test the robustness of the functional modules proposed in this paper, specially in degraded modes.

References

1. H. JIN KIM, R. VIDAL, D. H. SHIM, O. SHAKERNIA and S. SASTRY, "A Hierarchical Approach to Probabilistic Pursuit-Evasion Games with Unmanned Ground and Aerial Vehicles", *Proceedings of IEEE Conference on Decision and Control*, Orlando, December 2001.
2. B. SINOPOLI, M. MICHELI, G. DONATO and T.J. KOO, "Vision Based Navigation for an Unmanned Aerial Vehicle", *Proceedings of IEEE International Conference on Robotics and Automation*, 2001.
3. T. J LAFFEY, P. A. COX, J.L SCHMIDT, S. M. KAO and J.Y READ, Real-time knowledge-based systems. *AI Magazine*, 9(1):27-45, 1988.
4. F. F INGRAND, R. CHATILA, R. ALAMI, An Architecture for Dependable Autonomous Robots, *IARP-IEEE RAS Workshop on Dependable Robotics*, 2001, Seoul, South Korea.
5. M.L PUTERMAN, Markov Decision Processes, *John Wiley & Sons, INC*, 1994.
6. F.F INGRAND, M.P. GEORGEFF, and A. S. RAO. An Architecture for Real-time Reasoning and System Control. *IEEE Expert* 7(6) 33-44, December, 1992.
7. O. DESPOUYS and F. INGRAND, An Integrated Architecture for Planning and Execution Control, *RFIA 2000*, Paris, France.
8. http://www.cert.fr/dcsd/cd/PROCOSA.
9. T. MURATA, Petri Nets: properties, analysis and applications, *Proceeding of the IEEE*, 77(4), 541-580, 1989.
10. http://www.prolexia.fr/francais/ingenierie/references/EDIPET_doc.htm.

11. Guy. ZANON, Tiny, *Technical Report*, September, 1998.
12. C. BARROUIL and J. LEMAIRE, An Integrated Navigation System for a long range AUV, *in OCEAN'S98 :"Engineering for Sustainable Use of the Oceans"*,1998.
13. M. BARBIER, J. LEMAIRE and N. TOUMELIN, Procedures Planner for an A.U.V, *12th International Symposium on Unmanned Untethered Submersible Technology*, UUST01, Durham, NH, August 27-29, 2001.
14. M. R ENDSLEY, Measurement of situation awareness in dynamic systems. *Human Factors*, 37(1), 65-84, 1995.