Task planning and control for a multi-UAV system: architecture and algorithms*

Jeremi Gancet, Gautier Hattenberger, Rachid Alami and Simon Lacroix
LAAS-CNRS, 7, Avenue du Colonel Roche,
31077 Toulouse CEDEX 04, France
{*firstname.name*}*@laas.fr*

***Abstract*— This paper presents a decisional architecture and the associated algorithms for multi-UAV (Unmanned Aerial Vehicle) systems. The architecture enables different schemes of decision distribution in the system, depending on the available decision making capabilities of the UAVs and on the operational constraints related to the tasks to achieve. The paper mainly focuses on the deliberative layer of the UAVs: we detail a planning scheme where a symbolic planner relies on refinement tools that exploit UAVs and environment models. Integration effort related to decisional features is highlighted, and preliminary simulation results are provided.**

***Index Terms*— Multi-UAV systems - Architecture - Decision**

I. Introduction

Several UAV projects are led in different research teams (for instance [1], [2]), but up to now, few multi-UAVs applications have already been demonstrated [3], [4], [5]. Most of them mainly rely on operational UAV autonomy, *i.e.* the UAVs receive a pre-planned sequence of tasks to achieve, and do not exhibit high level decisional skills, such as planning or task allocation. The work described here aims at endowing the UAVs with such skills, in the context of surveillance and monitoring applications. In particular, application examples and illustrations take place in the context of the COMETS project [6], [7], that deals with the development of heterogeneous multi-UAVs systems for forest fires detection and monitoring applications.

In multi-UAV systems, autonomous deliberative activities require to consider temporal constraints, high uncertainties on tasks execution, and stringent reactivity to contingencies. Moreover, depending on the operational context, the ground operators might want to handle the activities or increase their control on some UAVs when necessary.

In order to take into account these constraints, we propose an approach that enables both centralized (i.e. human-centered, in a ground station) and distributed (i.e. delegated to UAVs) configurations of the decision.

A. Problem statement

Designing multi-robot (MR) architectures requires to define a decision making scheme, and to specify the interaction framework among the different robots of the system. For instance ALLIANCE [8] provides a behavior-oriented solution, enabling the design of totally distributed, fault tolerant multi-robot systems, whereas Simmons & al. [9] extend the three layers architecture model [10] within a MR framework, where interactions between robots occur along the different layers.

Although the work introduced in this paper takes place in the area of MR architecture, our architecture is not intended to substitute to these ones. We focus instead on several concerns related to multi-UAV applications:

- How can heterogeneous UAVs (both in terms of physical and decisional capabilities) be gathered within the same multi-robot system ?
- How to organize the various software components of the UAVs, and especially the decisional features ?
- How to refine the missions allocated to the UAVs in a way that will provide an efficient basis for coordination ?

These concerns significantly influenced the design and implementation of the decisional architecture and algorithms introduced in this paper. Indeed, besides the ability to integrate physically heterogeneous robots, is the capacity to cope with heterogeneity in terms of *decisional capabilities*. This is typically the case in the COMETS project: some UAVs are directly controlled by an operator, some others are only endowed with *operational autonomy* (their tasks being planned and monitored by a central station and/or human operators), whereas others have *decisional autonomy* capacities, *i.e.* they are able to achieve high level missions by themselves. Moreover, depending on the situation, a human operator or the central station should be able to take control over any of these UAVs.

B. Outline

After a brief discussion on the possible autonomy levels an UAV can exhibit, section III then details the various deliberative components of the architecture and their interrelations.

Section IV introduces a simulated example of the planning mechanisms described previously. Finally, section V presents current results and future work.

II. An architecture for heterogeneous multi-UAVs systems

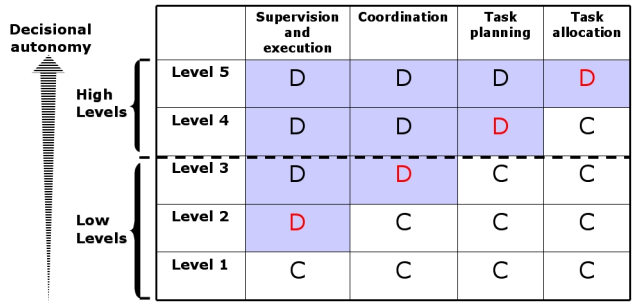
In a MR system, *decisional autonomy* encompasses the following features:

- **Supervision / execution**: the *executive* is a passive, reactive management of tasks execution, whereas *supervision* is a pro-active process managing the decisional activities of the robot.
- **Coordination**: It ensures the consistency of the activities within a group of robots and defines the mechanisms dedicated to prevent or solve possible resource conflicts.

*This work is partially supported by the Comets IST project # 34104 of the 5th European Framework program.

- **Mission refinement, planning and scheduling**: These decisional activities are dedicated to plan building, considering models of missions, tasks, and of the "world".
- **Task allocation**: How to distribute tasks among the robots. It requires to define both a task assignment protocol in the system, and some metrics to assess the relevance of assigning given tasks to such or such robot.

We define 5 *levels of decisional autonomy* for an UAV, according to whether these features are distributed (or delegated) to the UAV (in the DDN, for Distributed Decision Node), or centralized in a ground station (CDN, for Centralized Decision Node) (Figure 1).



	Supervision and execution	Coordination	Task planning	Task allocation
Level 5	D	D	D	D
Level 4	D	D	D	C
Level 3	D	D	C	C
Level 2	D	C	C	C
Level 1	C	C	C	C

Fig. 1. 5 levels of decisional autonomy. **C** stands for "Centralized", and **D** stands for "Distributed", *i.e.* autonomously performed by the robot.

This taxonomy is is to be understood in terms of *incremental delegation of decisional capabilities* by the multi-UAV system's user toward the UAVs. From the user's point of view, level 1 means a centralized full control of the system (*centralized* should be considered as available for operator). Level 2 enables an autonomous execution of partially ordered plan. Level 3 provides autonomous inter-UAV synchronization capabilities. Then a large gap appears between levels 3 and 4: up to the level 3, the CDN performs tasks planning and ensures the global consistency of the UAVs activities (**low levels** of decisional autonomy). Whereas level 4 is related to delegating mission refinement and planning activities to the UAV. Finally, level 5 enables autonomous tasks re-allocation : this is the highest delegation of decision making (i.e. the CDN only expresses high level goals to be achieved. Levels 4 and 5 are **high levels** of decisional autonomy.

Figure 2 depict the overall architecture of the UAVs, both for low levels and high levels of decisional autonomy. A CDN communicates with robots, exchanging messages whose abstraction is defined according to the robots levels of autonomy. Each robot has a number of functional components, and is endowed with a generic Distributed Decisional Node (DDN) that enables various configurations of decisional autonomy, ranging from the simplest up to the highest decisional capabilities. It encompasses an executive (this executive being actually common to all levels, we denote it as the *Multi-Level Executive* - MLE), and a Deliberative Layer (DL) which provides robots with higher levels of decisional capabilities.

- **The Multi Level Executive.** For the low levels, the DDN is restricted to an executive. For level 1, the MLE behaves as a transparent connecting point between the CDN and the robot's functional components. For levels 2 and 3, it manages tasks sequences execution, and at level 3 it enables simple coordination interactions with other robots of the same level (these mechanisms are detailed in [11]). It acts in the same way for levels 4 and 5, the only difference being that it is interfaced with the UAV's DL instead of the CDN.
- **The Deliberative Layer.** For the high autonomy levels, the DL deals with missions and tasks refinements, coordination activities, and task reallocation (for level 5). It encompasses the following components (figure 2):
  - The symbolic planner builds flexible plans skeletons: it transforms high level missions requests into partially ordered plans. For that purpose, it uses the algorithms of the *specialized refiners* (III-C).
  - The specialized refiners gather a set of features to support tasks decompositions and refinements during planning and coordination, relying on the UAVs and the environment's models.
  - The interaction manager provides the means to coordinate UAVs activities, relying on distributed negotiation mechanisms.
  - The supervisor has a central position in the DL: it transmits missions refinement requests to the symbolic planner, then triggers negotiation sessions with the interaction manager in order to coordinate the resulting plans. It finally sends plans to be executed toward the MLE, and monitors returned tasks / plans execution status.

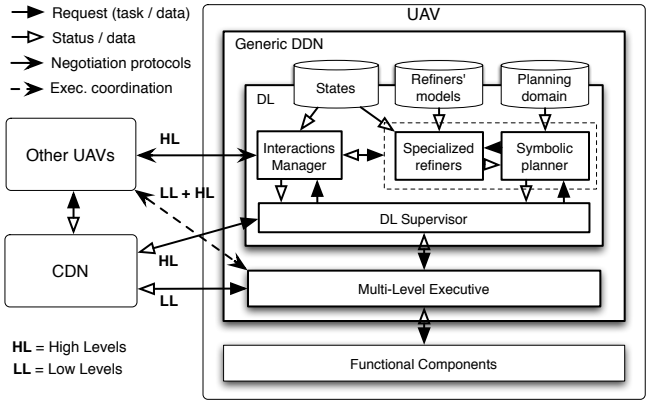


Fig. 2. DDN's components

III. Decisional Mechanisms

In this section, we first introduce the tasks model, before to focus on the deliberative activities and the specialized refiners.

A. Tasks model

Within the deliberative layer, we use an event-driven representation of tasks: a task is characterized by a starting event, triggered when the task is running, an ended event, triggered when the task is completed, and potentially other events triggered during this task's execution. Tasks have a temporal extent. The starting event is the only controllable event: all other kinds of events are contingent, *i.e.* the system

can not guarantee that such an event will occur, neither exactly when it will occur. A task can give rise to several, partially ordered contingent events during its execution.

Tasks may have pre-conditions and interruption conditions (exit-conditions), which satisfaction depends either on the occurrence of an event related to another task, on a time-related event, or on an external contingency event.

Plans consist in sequences of partially ordered tasks. We define a set of elementary tasks, assuming that any UAV included in the system is able to process such tasks in a consistent way. These tasks are:

- **Take-off**: the UAV should take-off and reach a given height
- **Land**: the UAV should land at a given location
- **Go-to**: the UAV should move up to the given location
- **Take-shot**: the UAV should perform a perception action (a single-shot, or a longer surveillance).
- **Wait**: the UAV should stay in a secured mode (e.g. hovering or loitering).

An additional **Synchronize** task is defined for low level coordination: such a task is processed by the MLE during the plan execution. It basically enables a synchronization of involved UAVs, through low level messages sending. Detail related to synchronizations processing is provided in [11].

B. Deliberative activities

1) General considerations related to the planning scheme: The symbolic planner we use is based on the Shop 2 HTN planner [12], exploiting a hierarchical definition of the planning domain. According to this paradigm, high level *methods* are decomposed into lower level tasks (either other methods or operators) when methods' preconditions are satisfied, until the planner reaches primitive tasks (*operators*).

We introduce time thanks to a particular encoding of the domain based on *Multi-Timeline Preprocessing* (MTL), according to [12]. This scheme enables to express durative and concurrent actions, which is very relevant in robot's tasks planning.

Moreover, we allow, for every COMETS task, the possibility to deal with temporal constraints: these time constraints are related to wishes or requirements, expressed in missions requests. Four possible time constraints are enabled in this way: *start before, start after, end before, end after*. When a method generates sub-tasks during its decomposition, these sub-tasks inherit the time constraints.

We distinguish two kinds of operators: *actual operators* (AO), corresponding to explicit tasks in the generated plan, and *convenience operators* (CO), manipulating intermediary data, but not directly dealing with actual robot's tasks.

AOs have the following properties:

- An unique ID (generated during the planning process)
- A dependence list: dependencies dealing with other (previous) operators. This list built using the MTL properties is used when the MLE receives a plan to execute: the dependencies are then turned into preconditions.
- A relative starting time: a time interval where the task's starting should be triggered.
- A duration: provided by the *specialized refiners*.
- Time constraints, inherited from higher level methods decomposition, during planning.
- Some parameters, according to the operation's type.

These AOs mainly match the elementary tasks defined previously (e.g take-off, gotoxyz, etc.). AOs may also match highest level tasks which can not be refined in the only UAV's context : such tasks require multi-UAV refinements, which occur in a second step, trough the *interactions manager*. The duration of such a *Joint Task* (JT) is not necessarily relevant during plan building, since it may depend on the task refinement issue in the multi-UAV context: in this case, the duration is let "unknown" for this task.

On the other hand, the COs are related to intermediary operations, such as calling the specialized refiners during planning. Applying such a CO operator is required before applying any AO operator, since it provides a way to link symbolic knowledge with actual models of the world: environment, UAVs, communications, etc.

```
(:method (general-gotoxyz ?destloc ?time-constraints)
  ; preconditions
      ((uavloc flying ?startloc) (not (eval (eql '?startloc '?destloc))))
  ; subtasks
      (:ordered (!compute-gotoxyz ?startloc ?destloc)
                (!task-gotoxyz ?id ?dependences ?startloc ?destloc
                               ?waypoints ?start ?duration ?time-constraints)))
)
```

Fig. 3. A Shop method for the generation of a "gotoxyz" primitive

```
(:operator (!compute-gotoxyz ?startloc ?destloc)
  ; preconditions
     ((assign ?result (compute-data 'gotoxyz (list '?startloc '?destloc))))
  ; delete list
     ()
  ; add list
     ((eval-ok gotoxyz ?result))
  ; cost
     0
)
```

Fig. 4. CO example: calls the specialized refiners features

```
(:operator (!task-gotoxyz ?currentid ?dependences ?startloc ?destloc
            ?waypoints ?start ?duration ?time-constraints)
  ; preconditions
     (
       (eval-ok gotoxyz ?pre-computed-data) ; (1)
       (assign ?duration (get-duration '?pre-computed-data)) ; (2)
       (assign ?waypoints(get-waypoints '?pre-computed-data)) ; (3)
       ...
  ; delete list
       ...
  ; add list
       ...
  ; cost
     (get-cost '?pre-computed-data)) ; (4)
```

Fig. 5. AO example: the "gotoxyz" operator

2) Exploiting the specialized refiners during the planning process: Figures 3, 4, 5, illustrate a "gotoxyz" method (fig. 3) giving rise first to the computation (CO, fig. 4) of data related to "gotoxyz" task, then applying the primitive "gotoxyz" task (AO, fig. 5). The "compute-gotoxyz" operator sends a request to the specialized refiners for the refinement of the "gotoxyz" task, taking into account initial location and destination location, and the returned result is added in the current planning state (through the logical atom "eval-ok...", in the operator's "add list" field). Then the "gotoxyz" operator exploits the corresponding result (line (1) on fig. 5). Finally,

the result is parsed into the different relevant data, e.g. duration, waypoints and costs associated to the "gotoxyz" operation application (resp. lines (2), (3) and (4) on fig. 5).

Figure 6 illustrates an instance of "gotoxyz" task, as it appears in a final plan.

```
(TASKREQ
   TASK-GOTOXYZ 13
   (DEPENDENCES ((ENDED 12) (ENDED 11)))
   (PARAMS
      (WAYPOINTS ((WP 100.000000 -140.000000 100.000000 0 0 0 0 -1)
          (WP 100.000000 -130.000000 90.000000 14 1 -1 1 -1)
                              ⋮
          (WP 120.000000 -10.000000 50.000000 164 13 -1 20 -1)))
      (START-TIME 240 352.000000)
      (DURATION 88 113.000000)
      (TIME-CONSTRAINTS NIL NIL NIL NIL)
)
```

Fig. 6. "Gotoxyz" task, ready to be executed

Actually, the specialized refiners have the means to process data for much more complex tasks, such as tasks requiring both refinements for perceptions and path planning (e.g. TSP with planned perceptions, see section III-C).

3) Exploiting resulting plans - multi-UAV coordination issues: Only the AOs are notified in the final plan. Such a plan is ready to be executed **iff** it does not contain any task requiring coordination with other UAVs, i.e. JTs. However, if the plan contains JTs, the plan coordination is performed in a second step, through the *interaction manager*.

The interaction manager provides the means to coordinate UAVs activities, relying on distributed negotiation mechanisms. All the tasks requiring multi-UAV interactions (simple synchronization or more complex JTs) are processed in the interactions manager, so that the joint operations can be coordinated, for each involved UAV, in terms of space and time.

Detail related to the interactions manager is not provided here, since still ongoing work. Mainly three issues are tackled:

- Temporal coordination: achieved relying on UAVs synchronizations. We defined and implemented a scheme to enable incremental negotiations related to possible time intervals synchronization. As a result, a group of UAVs acknowledge a common time interval in which the synchronization should occur.
- Spatial coordination: we consider interactions models, to reason about the interactions requirements within the JTs. Afterward, during plan execution, collision avoidance can be safely achieved applying a Plan Merging Protocol [13] on the planned trajectories of UAVs.
- Tasks re-allocations: this issue consist in enhancing the global activity of the UAVs, allowing them to redistribute some tasks, when relevant. For each UAV, the relevance should be assessed w.r.t. the current tasks costs / utility in the current plan. Preliminary work dealing with this purpose has already been led in [14].

During coordination, the interaction manager may as well request computations / refinements related to the environment and UAVs models, i.e. relying on the *specialized refiners*.

As a result of these coordination processes, a coordinated, ready-to-be-executed (but not necessarily definite) sequence of tasks is provided and inserted in the current MLE's plan.

C. The specialized refiners tool-box: overview

The specialized refiners provide a wide set of features to support tasks decompositions and refinements during planning and coordination. They rely on different models (environment, UAV, etc) and states regularly updated during the UAV's activity, and offer through a common interface a set of services related to paths generation, perception planning and communication constraints satisfaction checking. UAVs tasks are mainly related to *perception* and *motion*: indeed the general philosophy of these refiners is "computing the right place (and related motions) to perform the right perception".

Techniques involved in these refiners are quite basic ones, they are instances of refinement means: any other mean could be substituted to these ones in the same way, assuming that consistent results are returned.

The returned results are intended to provide the symbolic planner and the interactions manager with relevant information to estimate the robot's ability to perform given tasks in a given context: they provide various costs and the means to weight the consequences of inserting a given task into the current UAV's plan. The Traveling Salesman Problem feature (TSP), for instance (see hereafter), can be exploited to choose the most relevant order of achievement among a set of goals. Hence such information should be sufficiently realistic and produced in an efficient way, in order to be useful during on-line execution. The overall process is incremental and is subject to frequent revisions.

1) Models: The environment model gathers ground surface and airspace information. The ground model is a 2D square cells array whose attributes are related to fire, mapping and alarm monitoring. A burning factor representing the burning risk is associated to each cell. The airspace model, a 3D array of voxels, gives relevant information for trajectories and perception planning. It indicates whether a voxel is free or not. Potential waypoints for trajectories planning are nodes located at the center of voxels' facets. Edges are labeled with the cost for the UAV to move from one node to another adjacent node.

A generic UAV model provides information about flight capabilities and available resources. The perception model contains characteristics such as the expected coverage of the perception device, and reports the sensors availability.

Finally, a (quite simple) line of sight-like communication model is exploited to estimate the "communicability" between two entities, and to compute the communication coverage.

According to these models, various "services" can be provided: the next section provides a few algorithmic details related to some of these features.

2) Algorithms:

a) Perception planning: Given a location to be perceived, the refiners compute the best locations for a given UAV to perform useful perceptions, according to the environment model (considering obstacles such as hills or no-flight zones) and the perception task model. A measure of the utility is compared over a discrete set of positions in a 3D radius around the location to be perceived.

b) Path planning and TSP: Path planning is performed in a simple way (A* based) to compute paths in the dis-

cretized 3D environment. Then we exploit this simple path finding to compute the shortest path between several points (TSP): an approximated solution of the TSP is computed using a simple stochastic algorithm with two operations: insertion and permutation of locations.

c) Mapping: The mapping task aims at covering a whole given area in the shortest time. In this problem, we try to minimize the number of turns, according to [15].

The principle of the algorithm is to select a favored direction (along the longest straight line inside the area), and then to apply a sweeping pattern accordingly, assuming that areas are (or can be divided into) convex polygons.

d) Detection: This activity requires the UAV to fly over an area during a certain time, trying to minimize the time between two flights over a given ground cell. In the COMETS context, different priority values are attached to the cells, according to their *burning risk factors*, and detection considers this terrain's burnability. We implemented for this purpose an potential fields-based algorithm. Each cell of the ground is associated to a potential, initiated to a maximum value, and decreasing with time according to the a specific law. Perception coverage depends on the perception device's aperture, and flying altitude. Perceiving areas makes the corresponding potential raise according to the perception model (i.e. well perceived regions have their potential raised to their max, whereas badly perceived regions' potential is only slightly raised). At each increment, the move follows the steepest gradient in the potential field. Even for very low risk areas, the potential slowly decreases until reaching a low value that eventually attracts the UAV after a lapse of time.

IV. Illustration of the planning scheme

In this section, we describe a possible COMETS mission and we provide a three UAVs scenario instantiation.

A. Mission and scenario

1) Mission: The general mission's goal is to perform fire detection and fire monitoring over a given area A. The initial task allocation is performed by a human operator (issues related to autonomous task allocation deal with higher autonomous decisional capabilities, w.r.t. the levels introduced in section II). Fire alarm detection should be performed by one UAV over A. Every located alarm should be confirmed by another UAV. If an alarm is confirmed, then 2 UAVs should perform coordinated monitoring around the fire.

2) Scenario: Three UAVs are introduced in this scenario: one blimp (K), not very maneuverable but well adapted to high altitude flights, and two copters (H and M), much more maneuverable, having hovering capabilities, well adapted to low altitude flight.

K is requested to perform detection over A. After a certain amount of time, a first fire alarm is raised over the location L1, then a second fire alarm is raised over the location L2: H is requested to make perceptions around L1, and M should make perceptions around L2. In L1, the alarm is infirmed (false alarm). In L2, the alarm is confirmed: H is requested to perform coordinated perceptions with M around L2, for monitoring purpose (requires a synchronization of the monitoring). During this time, K keeps on performing fire detection around L1 and L2. The monitoring activities performed by H and M should go on until K's detection activity is ended. After a certain amount of time, K stops its detection activity: a synchronization signal is sent to H and M. All the UAVs come back to the base station.

B. Running the scenario

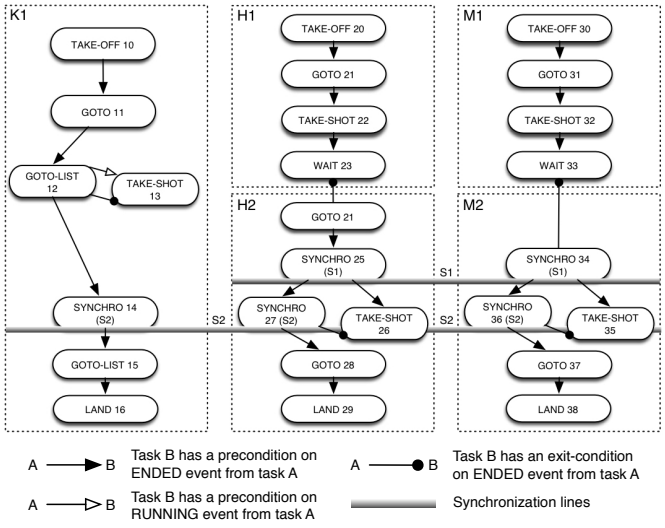


Fig. 7. Example of COMETS scenario: K,H and M's plans

Requests hereafter deal with high level Shop methods: once requested to Shop, they are decomposed into refined elementary tasks (resulting UAVs' refined plans are illustrated on fig. 7), exploiting the specialized refiners abilities. The sweeping pattern for fire detection is computed by the specialized refiners, as well as the most fitted perception locations close to L1 and L2 (for H and M), maximizing the perception utility (figure 8 depicts a simulated instance of this scenario).

1) **K blimp's mission:**

- K should perform detection over A during 15 minutes;
- THEN K should send sync.signal (S1) to H and M.
- THEN K should come back to the base station.

On fig. 7, task 11 is a "goto" task leading to area A. Task 12 is a "goto-list" task associated to the detection pattern computed by the specialized refiners. As task 12 is running, the perceptions are simultaneously triggered (task 13). Then once the synchronization is achieved, the "goto" task 15 makes K come back to the base station.

2) **H copter's mission (part 1: H1):** L1 alarm raised (trough K's perceptions): should be confirmed by H.

- H should make perceptions in L1 during 1 minute.
- THEN H should wait for further orders in secure mode.

Task 21 (fig. 7) is a "goto" task leading to L1.

3) **M copter's mission (part 1: M1):** L2 alarm raised (trough K's perception): should be confirmed by M.

- M should perform perceptions in L2 during 1 minute.
- THEN M should wait for further orders in secure mode.

Task 31 (fig. 7) is a "goto" task leading to L2.

4) **M copter's mission (part 2: M2):** L2 confirmed (M's perception): should perform coordinated monitoring.

- M should perform monitoring activity of L2 with H until receiving synchronization signal from K.
- THEN M should come back to the base station.

On fig. 7, task 34 is the synchronization with H for monitoring (task 35). Task 36 is the synchronization with K, which achievement stands as exit condition for task 35. Then task 37 is the "goto" task back to the base station.

5) **H copter's mission (part 2: H2):** L1 alarm is wrong, and L2 is confirmed (trough M's perception data processing): should perform coordinated monitoring.

- H should perform monitoring activity of L2 with M until receiving synchronization signal from K
- THEN M should come back to the base station.

On fig. 7, task 24 is a "goto" task leading to L2. Then task 25 is the synchronization with M for monitoring (task 26). Task 27 is the synchronization with K, which achievement stands as exit condition for task 26. Then task 28 is the "goto" task back to the base station.

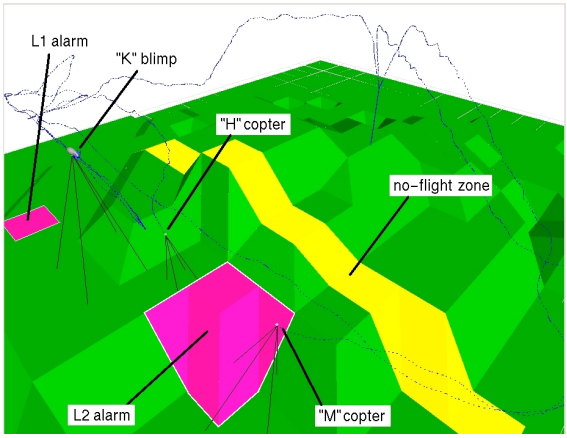


Fig. 8. Level 4 features in simulation: coordinated monitoring over L2

V. Results and future works

The whole architecture is developed considering the context of the COMETS project. In this frame, the 3 first levels of decisional autonomy have been tested both in simulation and in actual applications, with up to three UAVs (two copters and one blimp): during these tests, each MLE (section II) was receiving requests from a central, ground station.

The deliberative layer components that we introduced in this paper, and that aim at enabling higher levels (i.e. levels 4 and 5) of decisional autonomy have only been tested in simulation up to now.

The development of the deliberative layer's components is still ongoing work: the interaction manager, in particular, is worth to pay much attention, since it has a major role in the coordination of the UAVs activities. Preliminary tests deal with the negotiation of preferences for temporal synchronization of plans, considering synchronization tasks: such synchronizations are achieved afterward during the execution through the exchange of synchronization messages, as described in [11]. Some other directions have also previously been investigated, like MR dynamic task re-allocation [14], and more recently, multi-UAV zone covering, for mapping-like applications.

Fig. 9. 2 UAVs real test (low levels of decisional autonomy)

References

[1] P. Doherty, G. Granlund, K. Kuchcinski, E. Sandewall, K. Nordberg, E. Skarman, and J. Wiklund, "The witas unmanned aerial vehicle project," in *Proc. of the 14th European Conference on Artificial Intelligence (ECAI'00)*, Berlin, Germany, 2000, pp. 747–755.

[2] L. Wills, S. Sander, S. K. Kannan, A. D. Kahn, J. V. R. Prasad, and D. P. Schrage, "An open control platform for reconfigurable, distributed, hierarchical control systems," in *Proc of the 2000 AIAA Digital Avionics Conference*, 2000.

[3] S. Sukkarieh, E. Nettleton, J. Kim, M. Ridley, A. Goktogan, and H. Durrant-Whyte, "The anser project: Multi-uav data fusion," *International Journal on Robotics Research*, 2002.

[4] E. King, M. Alighanbari, Y. Kuwata, and J. How, "Coordination and control experiments on a multi-vehicle testbed," in *Proc. of the 2004 IEEE American Control Conference*, Boston, Massachusets, 2004.

[5] R. Vidal, S. Sastry, J. Kim, O. Shakernia, and D. Shim, "The berkeley aerial robot project (bear)," in *2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems IROS 2002, Workshop on Aerial Robotics*, Lausanne, Switzerland, 2002, pp. 1–10.

[6] A. Ollero *et al.*, "Architecture and perception issues in the comets multi-uav project," *IEEE Robotics and Automation Magazine, special issue on R & A in Europe: Projects funded by the Comm. of the E.U.*, 2004.

[7] Comets project official web page. [Online]. Available: http://www.comets-uavs.org/

[8] L. Parker, "Alliance: An architecture for fault-tolerant multi-robot cooperation," *IEEE Trans. on R.& A.*, vol. 14(2), pp. 220–240, 1998.

[9] R. Simmons, T. Smith, M. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot, "A layered architecture for coordination of mobile robots," in *Multi-Robot Systems: From Swarms to Intelligent Automata, Proc. of the 2002 NRL Workshop on Multi-Robot Systems*. Kluwer Academic, 2002.

[10] E. Gat, "Integrating planning and reacting in a heterogeneous asynchronous architecture for mobile robots," *SIGART Bulletin*, vol. 2, pp. 17–74, 1991.

[11] J. Gancet and S. Lacroix, "Embedding heterogeneous levels of decisional autonomy in multi-robot systems," in *Proc. of DARS'04*, 2004.

[12] D. Nau, T. Au, O. Ilghami, U. Kuter, W. Murdock, D. Wu, and F. Yaman, "Shop2: An htn planning system," *Artificial Intelligence Research*, vol. 20, pp. 379–404, 2003.

[13] R. Alami, F. Ingrand, and S. Qutub, "A scheme for coordinating multi-robot planning activities and plans execution," in *Proc. of the European Conf. on Artificial Intelligence (ECAI'98)*, 1998.

[14] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *Proc. of the Int. Conference on Robotic and Automation (ICRA'04)*, 2004.

[15] I. Mazza and A. Ollero, "Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Proc. of DARS'04*, 2004.