Journal of Intelligent and Robotic Systems manuscript No.
(will be inserted by the editor)

Multi-UAV cooperation and control for load transportation and deployment

I. Maza · K. Kondak · M. Bernard · A. Ollero

Received: date / Accepted: date

Abstract This paper deals with the cooperation and control of multiple UAVs with sensing and actuation capabilities. An architecture to perform cooperative missions with a multi-UAV platform is presented. The interactions between UAVs are not only information exchanges but also physical couplings required to cooperate in the joint transportation of a single load. Then, the paper also presents the control system for the transportation of a slung load by means of one or several helicopters. Experimental results of the load transportation system with one and three helicopters are shown. On the other hand, the UAVs considered in the platform can also deploy small objects, such as sensor nodes, on different locations if it is required. This feature along with the whole platform architecture are illustrated in the paper with a real multi-UAV mission for the deployment of sensor nodes to repair the connectivity of a wireless sensor network.

Keywords multiple UAVs · multi-UAV load transportation · sensor deployment · multi-UAV distributed decision

1 Introduction

The progress on miniaturization technologies, together with new sensors, embedded control systems and communication, have fuelled the development of many new small and relatively low cost Unmanned Aerial Vehicles (UAVs). However, constraints such as power consumption, weight and size plays an important role in UAVs, and particularly in small size, light and low cost UAVs. Then, the cooperation of many of these vehicles is the most suitable approach for many applications. A single powerful aerial vehicle equipped with a large array of different sensors of different modalities is limited at any one time to a

This work was partially funded by the European Union AWARE Project (IST-2006-33579), the CONET Network of Excellence (INFSO-ICT-224053) and the ROBAIR Project of the Spanish Research and Development Program (DPI2008-03847).

I. Maza · A. Ollero
Robotics, Vision and Control Group, University of Seville, 41092 Seville (Spain)
E-mail: imaza@cartuja.us.es,aollero@cartuja.us.es

K. Kondak · M. Bernard
Technische Universität Berlin, Einsteinufer 17, 10587 Berlin,
E-mail: kondak, bernard@cs.tu-berlin.de

A. Ollero
Center for Advanced Aerospace Technology (CATEC), Seville (Spain)
E-mail: aollero@catec.aero

single view point. However, a team of aerial vehicles can simultaneously collect information from multiple locations and exploit the information derived from multiple disparate points to build models that can be used to take decisions. Team members can exchange sensor information, collaborate to track and identify targets and perform detection and monitoring activities among other tasks [19]. Thus, for example, a team of aerial vehicles can be used for exploration [16], detection, precise localization, monitoring and measuring the evolution of natural disasters, such as forest fires. Furthermore, the multi-UAV approach leads to redundant solutions offering greater fault tolerance and flexibility.

Unmanned aerial vehicles are mainly concerned with environment sensing and perception activities, with limited ability to react against environment changes. The above mentioned surveillance and detection applications lie in this category. If the environment is considered known and static, or quasi static with low dynamic behaviour when comparing with the UAVs motion, then the coordination problem can be solved before the real-time execution of the mission. Then, the multi-UAV mission planning, task allocation and trajectory generation could be done before the mission execution. However, environment perturbations (e.g. in a disaster management scenario) or a lack of environmental data require real-time coordination efforts. Furthermore, applications in non segregated aerial spaces where other non-cooperative aircrafts may operate, also require collision detection and avoidance capabilities.

Other applications, such as tracking the motion of a mobile ground target, require the real-time computation of the trajectory from environment perception, and, in general, the closing of real-time control loops by means of environment perception in the aerial robot [20]. The same happens when considering the monitoring of dynamic events such as forest fires. In this case, the application of real-time coordination and cooperation methods is required.

However, notice that, up to now, in all the above mentioned applications, the aerial robots, as many other mobile robots, are mainly considered as platforms for environment sensing. Then, the aerial robots do not modify the state of the environment and there are no physical interactions between the UAV and the environment. Furthermore, the interactions between the UAVs are essentially information exchanges, without physical couplings between them.

This paper deals with the cooperation and control of multiple aerial robots with sensing and actuation capabilities for the deployment of loads, and particularly, sensor nodes. Furthermore, the paper considers the multi-UAV load transportation when an object is too heavy to be carried by one single UAV. The latter requires the consideration of physical interactions between the aerial robots.

On the other hand, Wireless Sensor Network (WSN) technologies have experienced an important development in the last ten years. Research work includes node mobility using robots and the navigation of mobile robots by using wireless sensor network information (see survey in [1]). Connectivity is a main issue in wireless sensor networks. Then, the application of robots for the deployment and connectivity repair of wireless sensor networks has been also proposed. In [6] the application of an UAV for deployment and connectivity repair of a wireless sensor network is proposed and demonstrated by using a single autonomous helicopter. In this paper, the deployment and connectivity repair of a wireless sensor network in a multi-UAV context is proposed and shown experimentally.

The transportation of a load by several ground robots has been an active subject of research and development for many years. The coordinated control of the motion of the vehicles should consider the involved forces. Thus, each robot could be controlled around a common compliance center attached to the transported object. Under the assumption that each robot holds the object firmly, the real trajectories of all of the robots are equal to the real trajectory of the object. Both centralized and decentralized compliant motion control algorithms have been proposed, including the consideration of non-holonomic constraints [14]. The method has been implemented in an experimental system with three tracked mobile robots with a force sensor. In [23] the decentralized control of cooperating mobile manipulators is studied with a designated lead robot being responsible for task planning. The control of each robot is decomposed (mechanically decoupled) into the control of the gross trajectory and the control of the grasp. The excessive forces due to robot positioning errors and odometry errors are accommodated by the compliant arms. In [4] the Omnimate system which uses a compliant linkage platform between two differential drive mobile robots is presented. In [9] distributed coordinated control of two rovers carrying a 2.5 meters long mockup of a photovoltaic tent is presented and demonstrated as an example of the CAMPOUT behavior-based control architecture.

The transportation of a single load by means of several helicopters has been also proposed in the literature. In experiments with two manned helicopters it was determined that the piloting of two coupled helicopters is a very challenging task, so at least, an automatic stabilization of the helicopters is required. The motivation for using two or more small helicopters instead of one with bigger load capacity are:

– Like in the case of real manned transport helicopters, the costs for two small helicopters are often less than for one with double load capacity.
– Whatever load capacity an existing helicopter has, there will be always a task where a higher load capacity is required. In this case the control software could allow to couple existing helicopters in order to form a system with sufficient load capacity.

Particularly, research on lifting and transportation of loads by means of two helicopters (twin lift) was presented in e.g. [18,21]. However, this research work has been done only in simulation. In this paper real flight experiments for load transportation with one and three helicopters are presented. We describe an automatic control system which allows the usage of one or multiple small size helicopters for slung load transportation. The number of helicopters should be configurable depending on helicopters' capabilities and the load to be transported.

The research and developments presented in this paper have been conducted in the framework of the AWARE project. The general objective of this project is the design, development and experimentation of a platform providing the middleware and the functionalities required for the cooperation among UAVs and a ground sensor-actuator wireless network, including mobile nodes. The platform will enable the operation in sites with difficult access and without communication infrastructure. Then, the project considers the self-deploying of the network by means of autonomous helicopters with the ability to transport and deploy loads (communication equipment and nodes of the ground network).

This paper is organized as follows. Section 2 presents the multi-UAV distributed architecture developed for the AWARE platform, that allows different levels of cooperation and coordination among the UAVs and between the UAVs and the environment, including both sensing and actuation. Section 3 is devoted to the transportation of a single load by means of multiple joined helicopters, and includes experiments with one and three helicopters. Section 4 presents the multi-UAV deployment of multiple loads and particularly the deployment of the sensor nodes of a wireless sensor network for connectivity repairing. The last sections are devoted to the Conclusions and References.

2 Architecture for aerial robots cooperation in the AWARE platform

This section addresses specifically the architectural aspects linked to the necessary coordination and cooperation issues induced by the operation of several AWARE subsystems, such as the aerial robots.

A global mission for the AWARE platform is specified by the user using the Human Machine Interface (HMI) software. Each mission $\mathcal{M}$ will consist in a set of *tasks* (possibly ordered) that must be executed by the platform. The task allocation process to the different UAVs could be done by the user or may be autonomously performed in a distributed way. The latter might be necessary in situations where large numbers of UAVs have to interact, where direct communication with a central station is not possible, or where the local dynamics of the situation require timely reaction of the UAVs involved in it.

The tasks that will be executed by the AWARE platform involve coordination, mainly for sharing space, and cooperation, for example for the surveillance at different altitudes or from different viewpoints of the same object, or when an UAV plays the role of a radio re-transmitter from/to other UAVs and the central station. Cooperation includes coordination, but there is role sharing between the subsystems to achieve a global common task.

In the following, a general framework that addresses the operation of the AWARE multi-UAV platform is described. Let us consider a team of UAVs that plan their motions according to a set of decentralized and cooperative rules $\mathcal{R}$. In particular, we assume that the set $\mathcal{R}$ defines k possible *tasks*

$T = \{\tau_1, \tau_2, \ldots, \tau_k\}$ that UAVs can perform, and specifies n *logical conditions* requiring a change of task in the current plan. Let $E = \{e^1, e^2, \ldots, e^n\}$ be a set of discrete events associated with such conditions. Each task has a set of m parameters $\Pi = \{\pi^1, \pi^2, \ldots, \pi^m\}$ defining its particular characteristics.

Aerial robotic systems composed of a physical plant and a decisional and control system implementing such kind of cooperation rules $\mathcal{R}$ can be modeled as hybrid systems [8,5,15]. The general components of a simplified hybrid model $\mathcal{H}$ are explained in the following. Let $q_i \in \mathcal{Q}$ be a vector describing the state of the i-th UAV in the configuration space $\mathcal{Q}$, and let $\tau \in T$ be the task that the UAV is currently executing. The i-th UAV's configuration q_i has a continuous dynamics

$$\dot{q}_i = f(q_i, c_i, \gamma_i), \tag{1}$$

where $c_i \in \mathcal{C}$ are control inputs and $\gamma_i \in \Gamma$ models the influence of the possible physical coupling with other UAVs and transported objects.

The i-th UAV's current task has a discrete dynamics $\delta : T \times E \rightarrow T$, i.e.

$$\tau^+ = \delta(\tau, e), \tag{2}$$

where $e \in E$ is an event (internal or external) requiring a change of task from τ to τ^+, both from the set of tasks T.

Event activation is generated by

$$e = \mathcal{A}(q_i, \varepsilon, X, \bar{\mu}), \tag{3}$$

where ε represents the internal events (such as changes in the execution states of the tasks), X is a vector with information about external events in the environment and $\bar{\mu}$ is a vector $\bar{\mu} = (\mu_1, \mu_2, \ldots, \mu_{N_m})$ containing the messages coming from N_m UAVs cooperating with the i-th UAV. Those messages are used for example in the negotiation processes involved in the intentional cooperation mechanisms and are generated in each robot by a decisional module $\mathcal{D}$. This module encompasses high level reasoning and planning, synchronization among different robots, negotiation protocols for task allocation and conflict resolution purposes, task management and supervision, complex task decomposition, etc.

In the following section, the particular details of the architecture adopted for the AWARE multi-UAV platform are described.

2.1 Architecture

The main objective in the design of the architecture was to impose few requirements to the execution capabilities of the autonomous vehicles to be integrated in the platform. Basically, those vehicles should be able to move to a given location and activate their payload when required. Then, autonomous vehicles from different manufacturers and research groups can be integrated in the AWARE architecture easily.

The global picture of the AWARE distributed UAV system is shown in Fig. 1. In each UAV, there are two main layers: the On-board Deliberative Layer (ODL) and the proprietary Executive Layer. The former deals with high-level distributed decision-making whereas the latter is in charge of the execution of the tasks. In the interface between both layers, the ODL sends task requests and receives the execution state of each task and the UAV state. For distributed decision-making purposes, interactions among the ODLs of different UAVs are required. Finally, the HMI software allows the user to specify the missions and tasks to be executed by the platform, and also to monitor the execution state of the tasks and the status of the different sub-systems of AWARE.

A more detailed view of the Deliberative Layer architecture is shown in Fig. 2. As it has been mentioned above, the ODL has interactions with its executive layer and with the ODL of other UAVs as well as with the HMI. The different modules shown in the ODL supports the distributed decision-making process involving cooperation and coordination. After presenting the task model in the next section, the operation of each module will be detailed.

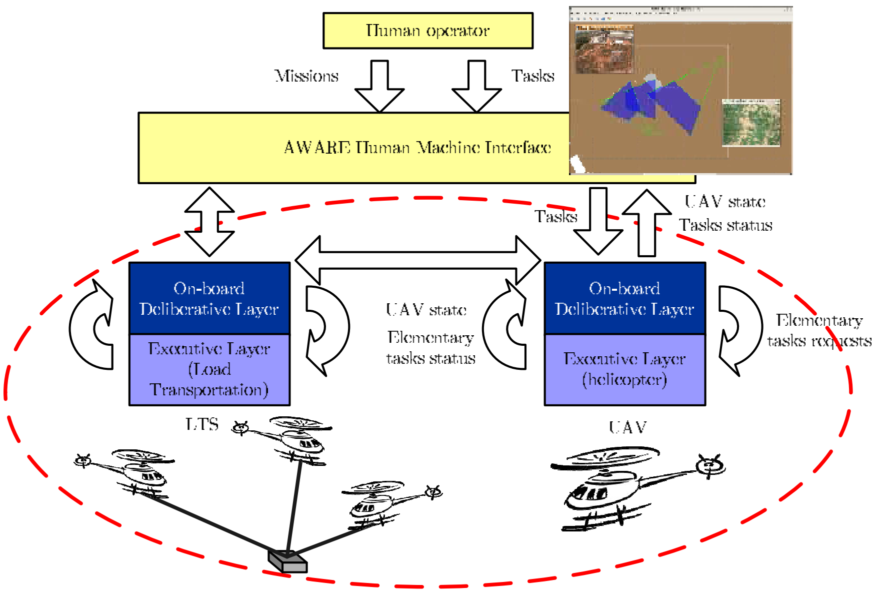

Human operator
Missions
Tasks
AWARE Human Machine Interface
Tasks
UAV state
Tasks status
On-board Deliberative Layer
Executive Layer (Load Transportation)
UAV state
Elementary tasks status
On-board Deliberative Layer
Executive Layer (helicopter)
Elementary tasks requests
LTS
UAV

Fig. 1 Global overview of the distributed UAV system architecture.

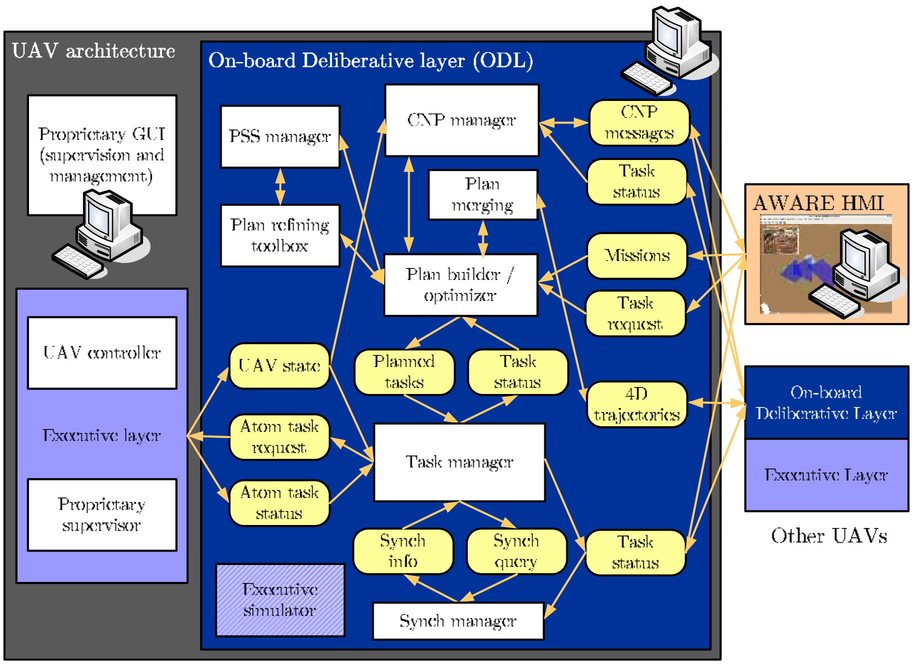


Fig. 2 Detailed view of the internal On-board Deliberative Layer (ODL) architecture.

2.2 Task model

Let us consider a mission $\mathcal{M}$ specified by the AWARE platform user. This mission is decomposed (autonomously or manually) in a set of partially ordered tasks $\mathcal{T}$. Those tasks can be allocated to the UAVs manually from the HMI or autonomously in a distributed way (see Sect. 2.5). Let us define a task with unique identifier k and type λ_k allocated to an UAV i as $\tau_k^i = (\lambda_k, ^-\Omega_k, \Omega_k^+, \varepsilon_k, \Pi_k)$, where $^-\Omega_k$ and Ω_k^+ are respectively the set of preconditions and postconditions of the task, and ε_k is the event associated to the task evolution (see Table 1). Finally, $\Pi_k = \{\pi_k^1, \pi_k^2, \ldots, \pi_k^m\}$ is the set of m parameters which characterizes the task. As an example, see Table 2, which shows the parameters considered in a task consisting in covering a given area for surveillance.

Table 1 Possible events considered in the status evolution of a task τ_k^i.

Event (ε_k)	**Description**
`EMPTY`	No task
`SCHEDULED`	The task is waiting to be executed
`RUNNING`	The task is in execution
`CHECKING`	The task is being checked against inconsistencies and static obstacles
`MERGING`	The task is in the plan merging process to avoid conflicts with the trajectories of other UAVs
`ABORTING`	The task is in process to be aborted. If it is finally aborted, the status will change to ABORTED, and otherwise will return to RUNNING.
`ABORTED`	The task has been aborted (the human operator has aborted it or the UAV was not able to accomplish the task properly)
`ENDED`	The task has been accomplished properly

Table 2 Parameters of a task with type λ_k = SURV.

Parameters (Π_k)	**Description**
π^1 (Polygon)	The set of vertices defining the polygon of the area to be covered by the UAV
π^2(Altitude)	Altitude (m) ellipsoid-based datum WGS84 for the flight
π^3(Speed)	Desired speed (m/s) for the flight

Regarding the type of task (λ_k) at the ODL level of the architecture, the list shown in Table 3 has been considered in the AWARE platform.

On the other hand, preconditions and postconditions are event-based mechanisms that can deal with events related to the evolution of the tasks states (see Table 1), to the reception of messages, to the detection of a given event by the Perception System (see Sect. 2.8), the elapsing of a certain time period, etc. Then, the execution of a task starts when all the associated preconditions are satisfied. On the other hand, it is also possible to specify postconditions, i.e. conditions which satisfaction triggers the abortion of a task. If a task does not have any precondition or postcondition, then $\tau_k^i = (\lambda_k, ^-\Omega_k = \emptyset, \Omega_k^+ = \emptyset, \varepsilon_k, \Pi_k)$.

An example of a precondition or a postcondition related to the evolution of the tasks is the "end of task" event of a different task. Furthermore, thanks to the synchronization manager module (see Sect. 2.3), it is possible to specify preconditions between tasks of different UAVs. Finally, it should be mentioned that perception events (not related to the execution of a task) such as the detection of a fire or a fireman in a disaster scenario, could be also the precondition of a task (i.e. a tracking task).

Table 3 Several type of tasks (λ_k) considered at the ODL level.

Type of task (λ_k)	**Description**
`TAKE-OFF`	The UAV takes off and stabilizes at a default safe height, then switches to a secured wait mode, waiting for further instructions.
`LAND`	The UAV starts landing procedures, lands, and is set to a ground secured mode.
`GOTO`	The UAV moves from its current location to a point P (or to its vicinity).
`GOTOLIST`	The UAV moves from its current location to each of the points of the waypoints list, following the order of the points.
`WAIT`	The UAV is set to a secured waiting mode: hover or pseudo-hover, during a given period.
`SURV`	The UAV covers a given area defined by a polygon at a certain altitude.
`DETECT`	The perception system of the UAV starts to operate in detection mode, providing an alert if a given event in the environment (fire, persons, etc.) appears.
`TRACK`	The perception system of the UAV starts to operate in tracking mode, providing (if possible) estimations of a given event location (fire, persons, etc.). The UAV moves accordingly to follow the event.
`HOME`	The UAV is commanded to return home.

The ODL processes the tasks received and generates simpler tasks, called elementary tasks, that are finally sent to the executive layer of the UAV. Let us define an elementary task with unique identifier k and type $\hat{\lambda}_k$ allocated to the i-th UAV as $\hat{\tau}_k^i = (\hat{\lambda}_k, \hat{\Pi}_k, \hat{\varepsilon}_k)$ where $\hat{\Pi}_k = \{\hat{\pi}_k^1, \hat{\pi}_k^2, \ldots, \hat{\pi}_k^{\hat{m}}\}$ is the set of $\hat{m}$ parameters which characterizes the elementary task and $\hat{\varepsilon}_k$ is the event associated to the elementary task evolution. It should be mentioned that `RUNNING` and `ENDED` are the only events considered for the elementary tasks. On the other hand, as an example, Table 4 shows the seven parameters that are considered in an elementary task consisting in visiting a given location.

Table 4 Elementary task with type $\hat{\lambda}$ = GOTO: list of parameters.

Parameters ($\hat{\Pi}$)	**Description**
$\hat{\pi}^1$(Latitude)	Latitude ellipsoid-based datum WGS84
$\hat{\pi}^2$(Longitude)	Longitude ellipsoid-based datum WGS84
$\hat{\pi}^3$(Altitude)	Altitude (m) ellipsoid-based datum WGS84
$\hat{\pi}^4$ (Speed)	Desired speed (m/s) along the way to the waypoint
$\hat{\pi}^5$(Force_heading)	1: force to the specified heading, 0: Not force
$\hat{\pi}^6$(Heading)	Desired heading (degree) along the way (N is 0 °, E is 90 °, W is −90 ° and S is 180 °)
$\hat{\pi}^7$(Payload)	1: to activate the payload around the location of the waypoint, 0: not to activate

Then, the vehicles to be integrated in the AWARE platform should be able to receive elementary tasks, report their associated execution events and execute them. A small set of elementary tasks have been considered in order to allow the integration of a broader number of vehicles from different manufacturers and research groups. Basically, those vehicles should be able to move to a given location and activate their payload when required.

In the next section, the module in charge of the management of the tasks following the model presented above is described.

2.3 Task and synchronization managers

The task manager module (see Fig. 2) receives the planned tasks from the plan builder module. Those tasks can have preconditions and/or postconditions and the task model assumed is based on elementary events processing, which are expected to occur whenever the states of the tasks and the environment evolve. The starting event is the only controllable event: all other kind of events related to a task are contingent, i.e. the system can neither guarantee that such an event will occur nor when exactly it may occur.

In each task request for the task manager, the operation to be applied should be one of the following two alternatives:

– Dynamic task insertion (`INSERT` operation): this allows to request tasks insertion in the UAV's current plan, according to the relative order specified for the newly inserted task, versus the current partial order of the tasks already scheduled. It allows to insert a task with preconditions and/or postconditions.
  Preconditions can be specified either as mandatory or optional. If it is mandatory and the precondition happens not to be satisfiable anymore, then the task is aborted. On the contrary, if it is specified as optional, the precondition is considered as satisfied (and hence removed from the task's list of preconditions) if it is actually satisfied or if its own satisfiability becomes unsatisfiable (and in this case the task is not aborted). An example of task precondition is the "end of task" event of a different task.
  On the other hand, it is also possible to specify postconditions, i.e. conditions which satisfaction triggers the abortion of a task. For example, it allows to interrupt a given task when another one is achieved: during a surveillance task, once a sequence of waypoints covering an area have been visited, we might want to interrupt also the `DETECT` task being carried out by the Perception Sub-System (PSS) on-board.
– Dynamic task abortion (`ABORT` operation): this mechanism allows to dynamically request task abortion in the current plan, while the plan is being executed. If the task is already running, then the abortion of the task is an interruption. If the task is not yet running, then the abortion is a cancellation (the task is de-scheduled). The abortion triggers a propagation mechanisms, that checks which of the scheduled tasks depends on the aborted task (i.e. the tasks having a precondition expecting an event from the aborted task, like an "end of task" event): if the dependence is a mandatory precondition, then this task is also aborted and so on. If it is an optional precondition, then the dependence is removed as if the precondition was satisfied, and the corresponding task is not aborted.

On the other hand, the task manager also interfaces with the executive layer of the UAV. It sends elementary tasks to the executive, which reports the state of both those tasks and the UAV itself.

The management of the preconditions and postconditions is carried out together with the task synchronization module. In the interface between both, there is a simple protocol based on messages. The synchronization manager is in charge of keeping the dependencies coherent among the different tasks in the current plan of the UAV, and also with the tasks of other UAVs.

When a new task request with preconditions and/or postconditions arrives to the task manager module, it sends a `REPORT_DEPS` message with the dependencies to the synchronization manager, which saves it in a local database.

The synchronization manager is always checking the state of the tasks in the distributed UAV system and, if there is any change, it updates the dependencies database. For instance, if a given task changes its status to `ENDED` and if this event is precondition for other tasks, those preconditions are changed to satisfied. On the other hand, if all the postconditions of a task are satisfied, an `INFO` message is sent to the task manager module that will proceed with the corresponding task abortion.

Before requesting an elementary task to the executive layer, the task manager sends a `QUERY` message with the sequence number of the task to the synchronization manager. The satisfiability is checked and the answer is sent back with an `INFO` message. If all the preconditions are not satisfied, the task manager will ask again periodically.

2.4 Plan builder / optimizer

In the plan building stage of the AWARE platform, there are two different possibilities:

- Offline planning: previous to the mission execution, the AWARE platform user can plan a mission to tune its parameters and check its feasibility using the EUROPA framework developed at NASA's Ames Research Center.
- Online planning: it is based on a plan builder / optimizer module integrated in the ODL architecture (see Fig. 2) and programmed to solve the specific planning problems of the UAVs in the AWARE platform.

Both options are described in the next subsections.

2.4.1 Plan builder / optimizer: offline planning

The EUROPA framework developed at NASA's Ames Research Center is available under NASA's open source agreement (NOSA) since 2007. NOSA is an OSI-approved software license accepted as open source but not free software. EUROPA (Extensible Universal Remote Operations Planning Architecture) is a class library and tool set for building planners (and/or schedulers) within a Constraint-based Temporal Planning paradigm and it is typically embedded in a host application. Constraint-based Temporal Planning (and Scheduling) is a paradigm of planning based on an explicit notion of time and a deep commitment to a constraint-based formulation of planning problems. This paradigm has been successfully applied in a wide range of practical planning problems and has a legacy of success in NASA applications.

As a simple application example in the AWARE project context, a deployment mission for an autonomous helicopter will be considered in the following. EUROPA is used offline before the mission execution to tune the parameters of the tasks and test its feasibility. The particular application domain model and problem definition has to be provided to the planner, which will generate a plan to solve the problem. In our example, the main entity application domain is the helicopter. The next decision is to identify the entities that will describe changes in the state of the helicopter as it moves around the environment performing a mission. Each entity is called a *timeline* in the EUROPA framework. Then, the following stage is to identify the states (called *predicates*) in which each timeline can be. Figure 3 shows the set of predicates identified on each timeline along with the variables in each state.

Analyzing the components of the helicopter produces the following breakdown of timelines (see Fig. 3) and states:

Navigator: controls the motion of the helicopter between locations and hovers at a location. States: The helicopter can be *hovering* at a location or *going* between locations.
Instrument: controls the instrument for dropping sensor nodes. States: the instrument can be *on* or *off*.
Commands: manages instructions from the HMI and checks the communication link after dropping a node. States: the helicopter can be instructed to *drop a node* and has to *check the communication link* in order to evaluate if the sensor network connectivity is repaired.

Then, the application domain description is encoded in NDDL (an acronym for New Domain Description Language) along with its initial state. The locations of the waypoints where the sensor nodes should be dropped, the initial location and battery level of the UAV and the different paths (with their associated costs computed by the plan refining toolbox) between the locations of interest are specified.

Finally, let consider that the goal of the mission is to deploy three sensor nodes in different waypoints:

- `wp1` at time 30.
- `wp4` at time 60.
- `wp3` at time 90.

It can be also easily encoded in NDDL in the following way:

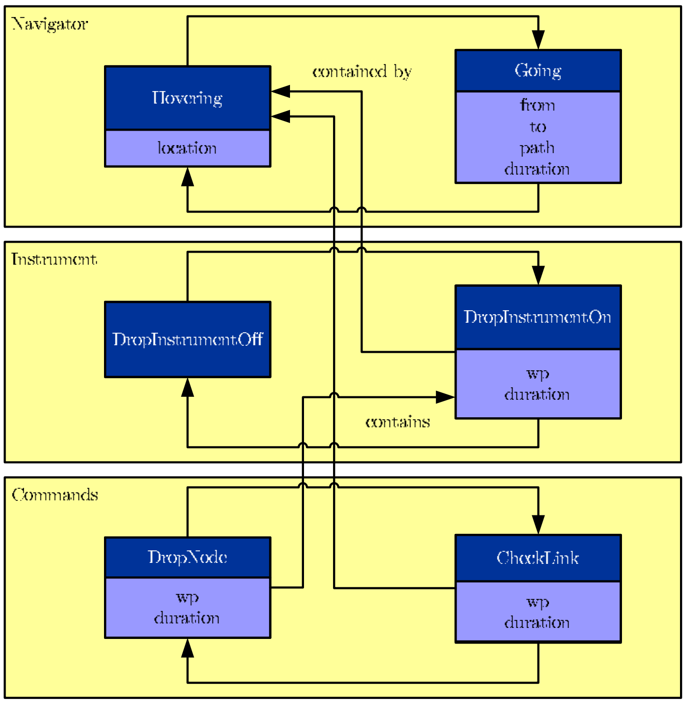


Fig. 3 Timelines and Predicates with Transitions between Predicates on each Timeline.

```
goal(Commands.DropNode drop_node_1); drop_node_1.start.specify(30);
drop_node_1.wp.specify(wp1);

goal(Commands.DropNode drop_node_2); drop_node_2.start.specify(60);
drop_node_2.wp.specify(wp4);

goal(Commands.DropNode drop_node_0); drop_node_0.start.specify(90);
drop_node_0.wp.specify(wp3);
```

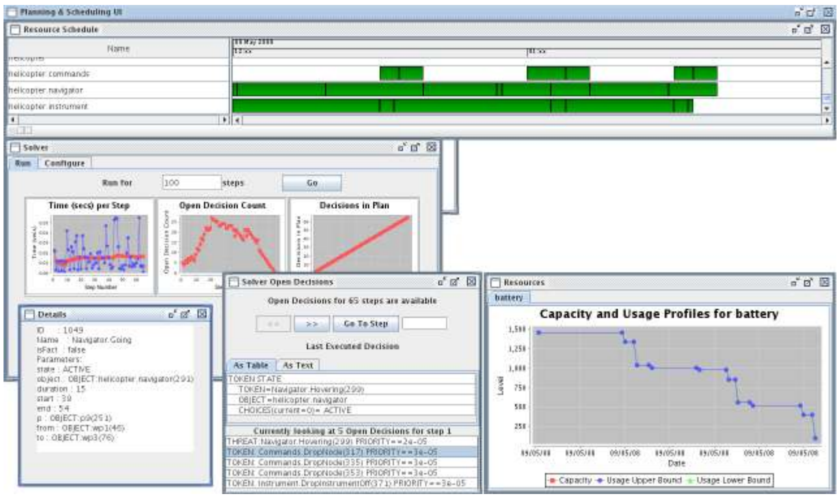
With the model and the initial state specified, the planner can be started to compute the solution. Figure 4 shows a screenshot with the visualization of the results obtained with PlanWorks.

From the solution obtained, the AWARE platform user can check the feasibility of the mission and tune its parameters properly.

2.4.2 Plan builder/optimizer: online planning

In general, the plan builder/optimizer module running during the mission execution generates a plan $\mathcal{P}$ as a set of partially ordered tasks. In the AWARE platform, the main function of the online planner will consist in ordering the motion tasks allocated to the UAV. Let us consider the i-th UAV with a set of n_m motion tasks to be executed. The planner will compute the order of the tasks $\{\tau_k^i / k = 1 \ldots n_m\}$ that minimizes the execution cost:

$$C^i = \sum_{k=1}^{n_m - 1} c_{k,k+1}^i \tag{4}$$



```
ID     : 1049
Name   : Navigator.Going
isFact : false
Parameters:
state : ACTIVE
object : OBJECT:helicopter.navigator(291)
duration : 15
start : 39
end : 54
p : OBJECT:p3(251)
from : OBJECT:wp1(46)
to : OBJECT:wp3(76)
```

```
TOKEN STATE
   TOKEN=Navigator.Hovering(299)
   OBJECT=helicopter.navigator
   CHOICES(current=0)= ACTIVE
Currently looking at 5 Open Decisions for step 1
THREAT:Navigator.Hovering(299) PRIORITY==2e-05
TOKEN: Commands.DropNode(317) PRIORITY==3e-05
TOKEN: Commands.DropNode(335) PRIORITY==3e-05
TOKEN: Commands.DropNode(353) PRIORITY==3e-05
TOKEN: Instrument.DropInstrumentOff(371) PRIORITY==3e-05
```

Fig. 4 Screenshot with the visualization of the results obtained with PlanWorks.

where $c^i_{k,k+1}$ is the motion cost between the locations associated to the tasks τ^i_k and τ^i_{k+1}. This problem is an instance of the *Travelling Salesmen Problem*, often referred as *TSP*, which is NP-hard. The simplest exact algorithm to solve it is based on a brute force search that tries all the ordered combinations of tasks. The running time for this approach lies within a polynomial factor of $\mathcal{O}((n_m - 1)!)$, so this solution is only feasible when a small number of tasks are allocated to the UAVs, which is the case in most of the AWARE platform missions.

But, if the user delegates the allocation process of tasks to the ODLs, each UAV will have to run many times the planning algorithm during the autonomous negotiation with other UAVs. Then, when the autonomous distributed allocation is launched, another algorithm with a lower computational cost is required. Each time a new task is received, the plan builder runs an algorithm that inserts the new task in all the possible and feasible locations in the current plan and chooses the order with lowest plan cost.

In the next Section, the module driving the autonomous distributed task allocation process is described.

2.5 CNP manager

The CNP manager module (see Fig. 2) operation is based on the Contract Net Protocol [22] and it manages the distributed task allocation process among the different UAVs.

The *multi-robot task allocation* (MRTA) problem has become a key research topic in the field of distributed multirobot coordination in recent years. In the AWARE project, three algorithms (SIT, SET and S+T) have been developed and implemented to solve the distributed task allocation problem among the UAVs in the platform. All of them apply a market-based approach in which the UAVs consider their local plans when bidding and multiple tasks can be allocated to a single UAV during the negotiation process. The second one (SET) is based on the negotiation of subset of tasks and can be considered as a generalization of the former (SIT), which only negotiates single tasks. Both algorithms have been tested in simulation with good results in multiple missions involving the execution of random GOTO tasks [25].

The latter algorithm (S+T) solves the MRTA problem in applications that require the cooperation among the UAVs to accomplish all the tasks. If an UAV cannot execute a task by itself, it asks for help and, if possible, another UAV will provide the required service. In the AWARE platform, tasks such as surveillance, that could involve transmitting data in real-time to a given location are considered.

For those tasks, communication relay services could be required if the communication range of a single UAV is not enough. On the other hand, the parameters of the S+T algorithm can be adapted to give priority to either the execution time or the energy consumption in the mission. The potential generation of deadlocks associated to the relation between tasks and services have been studied and a distributed algorithm that prevents them have been also implemented [26].

In the following, the SIT algorithm used during the mission execution described in Section 4 is detailed.

2.5.1 Dynamic Single Task Negotiations With Multiple Allocations To A Single Robot (SIT)

As our goal was to find solutions close to the global optimum (minimize the sum of all the individual costs assuming independent tasks), the approach presented in [7] was taken as a starting point. In the same manner, robots with a local plan and multiple tasks allocated to a single robot during the negotiation were considered. But in the implementation of the SIT algorithm, several differences with the work in [7] can be pointed out: revenues are not used, a different synchronization method is applied and there is an agent acting as an entry point for the tasks.

In the negotiation process, each UAV bids for a task with the cost of inserting this task in its local plan (marginal cost). Let us assume that the i-th UAV has a local plan $\mathcal{P}_i$ consisting in a set of n ordered tasks $\tau_1, \tau_2, \ldots, \tau_n$ with cost C^i and receives a new task. If this task is inserted at the position j in the plan $\mathcal{P}_i$, then a new plan $\mathcal{P}_i(\tau_j)$ with cost $C^i(\tau_j)$ is generated. In that case, the associated marginal cost μ_j is given by:

$$\mu_j = C^i(\tau_j) - C^i \qquad (5)$$

The plan builder module of each UAV will compute the optimal insertion point of the new task in its current plan. Taking into account the local plan of each UAV in the negotiation leads to better solutions as it will be shown later.

The SIT algorithm is based on two different roles played dynamically by the UAVs: auctioneer and bidders. In each auction there is only one auctioneer which is the UAV that has the token. The auction is opened for a period of time and all the bids received within it are considered. When the auction is finished and the task allocated, the auctioneer considers to pass the token to another UAV. If that happens, the auctioneer changes its role to a bidder role and the UAV with the token becomes the new auctioneer. These basic steps of the algorithms executed by the auctioneer and the bidders are given in Algorithms 1 and 2. In Algorithm 1, the best bid collected by the auctioneer is increased by a given percentage (usually 1%) to avoid transactions that will not significantly improve the solution.

Algorithm 1 SIT auctioneer algorithm

```
if there is any task to announce then
  announce task
  while timer is running do
    receive bids
  end while
  calculate best bid
  if best bid is smaller than the auctioneer bid then
    send task to best bidder
  end if
  delete task from announcement list
end if
```

The main difference with the basic CNP protocol is that the bid of each UAV depends on its current plan and every time the local plan changes, the negotiation continues until no bids improve the current global allocation. When the initial negotiation is over, the mission execution can start, but new tasks

Algorithm 2 SIT bidder algorithm

```
a new message is received
if new message is a task announcement then
    calculate the optimal position of the task in the local plan
    calculate bid (marginal cost)
    send bid to the auctioneer
else if new message is a task award then
    insert task in the local plan in the position calculated before
    introduce task in announcement list
end if
```

can be generated at any moment. Therefore, the negotiation is dynamic in the sense that new tasks are handled also during the mission execution. All the UAVs take part in the negotiation of those new tasks with the only restriction that the current tasks in execution are not re-negotiated.

The SIT algorithm has been tested in multi-UAV missions consisting in visiting waypoints and returning to the home location. In this case, the local plan cost for an UAV i visiting a set of n_w ordered waypoints $P_1, P_2, \dots, P_{n_w}$ can be expressed as:

$$C^i = d(P(x_i), P_1) + \sum_{l=2}^{n_w} d(P_{l-1}, P_l) + d(P_{n_w}, P(h_i)), \tag{6}$$

where $P(x_i)$ are coordinates corresponding to the current state of the i-th UAV, $P(h_i)$ is its home location and $d(A, B)$ is the Euclidean distance between the points A and B. In this particular missions, each UAV should build its own local plan visiting the waypoints in an order that minimizes the total distance travelled. This problem is equivalent to the TSP problem which is a well known *NP-hard* problem. In our implementation, a greedy approach has been applied to solve it, inserting the new task in the position which minimizes its insertion cost.

Hundreds of simulations with different number of UAVs and waypoints have been run to compare the SIT algorithm with the global optimal solution. Additionally, another algorithm has been implemented in order to evaluate the relevance of the local plans computed by the plan builder module in the quality of the solutions. This second algorithm, that will be called *NoP* (No local Plan), uses a basic CNP protocol where UAVs only participate in the auction when they are idle. Furthermore, a brute force algorithm has been used to compute the global optimal solutions when the sum of UAVs and tasks is below a certain value.

In particular, for each given number of UAVs and waypoints, one hundred missions have been run in a virtual scenario of 1000x1000 meters using random positions for the UAVs and the waypoints. Each mission has been simulated with the two algorithms implemented and Table 5 shows the different solutions compared with the global optimum. In each cell the first number is the arithmetic mean of the global cost for the 100 random missions, the value between brackets is its standard deviation (in meters) and the third number is the difference in percentage with the optimal solution. The global cost is given by the sum of the individual costs of the UAVs.

From the results, it should be noted that using a local plan during the auction process improves the solutions significantly. On the other hand, the SIT algorithm achieves very good results, with a maximum difference of 4.7% w.r.t. the optimal solution. Also, it is important to point out that the standard deviation values are high because missions are calculated at random, i.e., the global cost of the different random missions can differ very much among them.

2.6 Plan merging module

When considering the different plans of the UAVs, the main resource they share is the airspace. Therefore, a plan merging module (see Fig. 2) has been included in the architecture to detect potential conflicts

UAVs	Tasks	NoP	SIT	Optimum
3	3	$2371,22$ $(742,4)$ $65,18\%$	$1453,44$ $(369,63)$ $1,25\%$	$1435,4$ $(362,36)$
3	5	$4144,7$ $(923,42)$ $101,23\%$	$2097,83$ $(414,52)$ $1,74\%$	$2061,8$ $(396,3)$
3	7	$5073,2$ $(788,75)$ $114,73\%$	$2473,65$ $(385,46)$ $4,7\%$	$2362,6$ $(335,81)$
3	9	$6070,8$ $(850,46)$ $129,13\%$	$2816,59$ $(398,16)$ $6,3\%$	$2649,5$ $(332,58)$
5	3	$1764,36$ $(627,87)$ $39,49\%$	$1274,88$ $(365,74)$ $0,79\%$	$1264,78$ $(356,04)$
5	5	$3808,55$ $(921,45)$ $112,37\%$	$1842,96$ $(363,62)$ $2,76\%$	$1793,35$ $(337,56)$
5	7	$5407,59$ $(1238,38)$ $150,15\%$	$2225,67$ $(384,74)$ $2,96\%$	$2161,68$ $(365,82)$

Table 5 Solutions computed with the distributed task allocation algorithms implemented and the optimal result. The first number is the arithmetic mean of the global cost, the value between brackets is its standard deviation (both values in meters) and the third number is the difference in % with the optimal solution.

among the different trajectories and also to follow a policy in order to avoid them. Then, this module has to interact with the plan builder module and also with other UAVs to interchange the different 4D trajectories.

A distributed collision avoidance method (see Algorithm 3) for the helicopters of the AWARE platform has been designed and implemented. The method is based on the hovering capabilities of the helicopters and guarantees that each trajectory to be followed by the UAVs is clear of other UAVs before proceeding to the execution.

This algorithm has been applied in the execution of the mission described in Section 4.

2.7 Plan refining toolbox

The plan refining toolbox (see Fig. 2) provides services to the plan builder module and can also interact with the Perception Sub-System (see Sect. 2.8) depending on the UAV mission.

The main services provided are:

- Task decomposition rules.
- Path planning algorithms to avoid static obstacles.
- Task and mission costs computations for the planner module.

Those services are based on the decomposition of the tasks received by the ODL into an ordered set of elementary tasks. Then, once a task $\tau_k^i = (\lambda_k, {}^{-}\Omega_k, \Omega_k^{+}, \varepsilon_k, \Pi_k)$ is received, it is processed following a set of decomposition rules $\mathcal{D}$ and an associated set of n_e elementary ordered tasks $\{{}^1\hat{\tau}_k^i, {}^2\hat{\tau}_k^i, \ldots, {}^{n_e}\hat{\tau}_k^i\}$ is obtained. This set of tasks inherits as a whole, the preconditions ${}^{-}\Omega_k$ and postconditions Ω_k^{+} from the task τ_k^i.

In the AWARE platform, there are several predefined decomposition rules $\mathcal{D}$ for the different possible tasks. For example, for a surveillance task, consisting in covering a given area defined by a polygon at a certain altitude, an algorithm based in [16] is applied to decompose it into elementary GOTO tasks:

$$\tau_k^i = (\lambda_k = \mathrm{SURV}, {}^{-}\Omega_k, \Omega_k^{+}, \varepsilon_k, \Pi_k) \longrightarrow \{{}^j\hat{\tau}_k^i = (\hat{\lambda}_k = \mathrm{GOTO}, \hat{\varepsilon}_k, \hat{\Pi}_k)/j = 1 \ldots n_e\} \qquad (7)$$

Algorithm 3 Distributed collision avoidance algorithm running in the i-th UAV

```
while (true) do
  a new message is received from UAV $j$
  if new message is a path clearance ($\Delta_j$) request with timestamp then
    if a task is running with path $\Delta_i$ then
      if there is no conflict between paths $\Delta_i$ and $\Delta_j$ then
        send a path grant $\Delta_j$ message to UAV $j$
      else if then
        add the new request to the list of pending path clearance requests
      end if
    else
      if there is no conflict between current UAV $i$ location and path $\Delta_j$ then
        send a path grant $\Delta_j$ message to UAV $j$
      else if then
        add the new request to the list of pending path clearance requests
      end if
    end if
  else if new message is a $\Delta_k$ path grant message AND elementary task $\hat{\tau}_k$ is ready to be executed then
    if $\Delta_k$ path grant messages have been received from all the UAVs then
      execute corresponding GOTO elementary task $\hat{\tau}_k$
    else if then
      $\Delta_k$ path grant messages counter is increased
    end if
  end if
  check again the clearance state of the pending path requests previously received
end while
```

2.8 Perception subsystem (PSS) module

The main purpose of the Perception System (PS) of the AWARE platform is to build and update a consistent representation of the environment. A fully distributed probabilistic framework has been developed in order to achieve detection and tracking of events using the sensors provided by the AWARE platform: visual and infrared images from UAVs and ground cameras, scalar measures like temperature, humidity, CO or node signal strength from the sensor nodes of the WSN. It allows reducing the network bandwidth requirements on data transmission and dividing the processing load among different computers. As a result, the scalability of the whole system will be improved. In addition, the ability of separately process the information increases the robustness of the architecture.

Then, the whole PS system is divided in several software instances called perception sub-systems (PSS) modules, each attached to an AWARE platform element with perception capabilities. Then, there are PSS modules for the UAVs with cameras on-board (see Fig. 2), ground cameras, and the wireless sensor network (WSN). Each of them processes locally the environment information (images, sensors, ...) in order to reduce the amount of data transferred through the network. All the PSSs share their beliefs about specific events. The variety of events to be detected and tracked by the PSS is large but a common representation based on a probabilistic representation has been considered [24, 17]. The events are related with the real world by means of their position and velocity in a global coordinate frame. This information is always accompanied by the error estimation represented as an information matrix. In order to disambiguate among different events located in the same place (or close), general information about the event is also included: mean color, histogram, intensity and received signal strength indication (RSSI) when available. Finally, in order to fuse the different estimations, an Information Filter has been applied due to its properties for distributed implementation.

3 Multi-UAV load deployment

The transportation of loads by a single UAV is strongly limited by the payload constraint of the UAV. Then, when using small UAVs, this constraint may preclude the transportation and deployment of communication equipment or loads required for the application, as for example first aid supplies required for victims in search and rescue operations. The system designed in the AWARE project allows the transportation of a single load by means of several helicopters. The number of helicopters is configurable depending on helicopters capabilities and the load to be transported. The multi-UAV cooperation in the load transportation task is characterized by the physical coupling between the members of the team. Then, the individuals are connected by physical links transmitted through the common load. The dynamic behaviour of the state variables q_i of the i-th helicopter is given by equation (1).

In terms of motion planning and collision avoidance, all the members of the team and the load can be considered as a single entity. Let τ be the task being executed by this single entity, such as for example GOTO or GOTOLIST in the above section. Then, the control law can be expressed as

$$c_i = g(q_i, \bar{q}_i, \tau), \tag{8}$$

In general the control law u_i applied to each helicopter depends not only on q_i but also of the state variables $\bar{q}_i$ of the other N_c helicopters linked to the load. However, as will be shown in this section, in some cases the problem can be decoupled.

3.1 Modeling

The model of a small size helicopter is a key component for behavior description of the whole system composed of one or several helicopters coupled to a load by means of ropes. Here we use the model presented in our previous work [12,13]. The small size model helicopter shows some specific effects which are not presented or are negligible small in case of the full-size helicopter and vice versa. For that reason, it is impossible to use the models derived for full-size helicopters (see e.g. [10]) without any adaptation. As it was pointed out in our previous work the main differences between model and full size helicopters in respect of modeling and control are:

- a model helicopter has a much higher ratio of the main rotor mass to the fuselage mass
- the main rotor rotation speed of a model helicopter is higher compared to most full-size helicopters
- a very stiff main rotor without flapping hinges (for almost all commercially available helicopters)

Due to these differences, the inertial effects of the main rotor make a significant contribution to the rotational dynamics of the system and can not be neglected. Therefore, the mechanical model should be considered as composed of two rigid bodies: the main rotor and the fuselage. For the considered class of model helicopters, the dominant component is the main rotor and not the fuselage. Unfortunately, the main rotor is very often neglected in papers related to modeling and control of small-size helicopters and only one rigid body – the fuselage – is accounted for in the dynamical equations.

The complete model of a model helicopter is composed of two main components: the mechanical model and the model for generation of aerodynamic forces and torques. From experimental results with helicopters we concluded that the generation of aerodynamic forces and torques, at least for the considered class of helicopters, can be approximated with simple algebraic relations (corresponding time delay should be taken into account). Therefore, the dynamics of one small-size helicopter or of a system composed of several coupled helicopters are mostly determined by its mechanical model. The lifting forces $\mathbf{F}_3^{MR}$ and torques $\mathbf{T}_{1,2}^{MR}$ generated by the main rotor of each helicopter, see Fig. 5, and the forces $\mathbf{F}_2^{TR}$ generated by the tail rotors will be considered as abstract control inputs $\mathbf{c}$ in the above equation.

The important issue for the control of a helicopter coupled to a load is illustrated in Fig. 5. The mechanical model of the helicopter is composed of two rigid bodies: the fuselage F and the spinning main rotor MR. The load, denoted as mass point cm_L, is connected to the helicopters fuselage by means

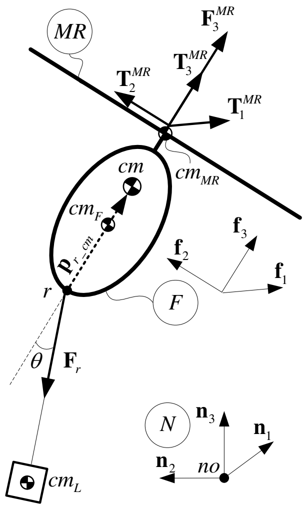


Fig. 5 Mechanical model of the helicopter connected to the load

of a rope in the point r. The motion of the whole system is considered in respect to a Newtonian frame N. The points cm_F, cm_{MR} and cm are the center of mass (CoM) of the fuselage, of the main rotor and of the complete helicopter respectively. For a real system it is difficult to place the point r at the helicopter CoM cm, so the vector $\mathbf{p}_{r-cm}$ connecting these two point is not zero and the rope force $\mathbf{F}_r$ causes a non zero torque $\mathbf{T}_r = \mathbf{F}_r \times \mathbf{p}_{r-cm}$ on the helicopter fuselage F. As we have shown in [12, 13] the rotation dynamics of the helicopter modeled as two rigid bodies is quite complicated, but is not coupled with the translation dynamics which means that equations for rotational dynamics depend only on generalized speeds describing the rotation of the helicopter. Because the lifting force $\mathbf{F}_3^{MR}$ generated by the main rotor is approximately perpendicular to the main rotor plane or to the fuselage plane $\mathbf{f}_1 - \mathbf{f}_2$, the translational accelerations are always given by the absolute value of $\mathbf{F}_3^{MR}$ and the orientation of the helicopter. Therefore, the relationship between rotation and translation dynamics for a single helicopter can be expressed as follows: rotation $\Rightarrow$ translation.

In the case where one or several helicopters are connected by means of rope to a load, the rope force $\mathbf{F}_r$ and, therefore, an additional torque $\mathbf{T}_r$ acts on the fuselage of the helicopter. This torque depends on orientation of the helicopter and its translational motion in the frame N (e.g. if helicopter and load are in free fall, $\mathbf{F}_r$ and $\mathbf{T}_r$ are zero). Here we have a more complicated relationship between rotation and translation dynamics for each helicopter: rotation $\Leftrightarrow$ translation. If several helicopters are connected to the load, the translational and rotational motion of one particular helicopter has direct influence on the rotational dynamics of all other helicopters in the compound. Even the translation with constant acceleration, e.g. to the right in the Fig. 5, can cause oscillation of the angle θ between the rope and the helicopter axis. Then, the coupling in equation (1) can be expressed as

$$\gamma_i = h(q_i, {}^p\bar{q}_i, {}^p\dot{\bar{q}}_i), \tag{9}$$

with vector ${}^p\bar{q}_i = (q_{i_1}, q_{i_2}, \ldots, q_{i_{N_c}})$ containing the configurations of the N_c neighbors physically connected to the i-th helicopter.

The strong mutual coupling between rotation and translation makes the usage of an orientation controller, which was designed for a single helicopter, problematic (even if the techniques for robust control design were used). Please note that in many practical cases the absolute value of torque $\mathbf{T}_r$ is similar or even larger than the values of torques needed to control the rotation of a single helicopter without the rope.

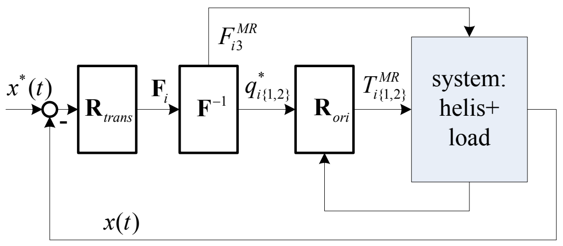


Fig. 6 General control scheme for control of coupled helicopters

For modeling we consider the general case where n helicopters are connected to one load. To get the dynamical equations of motion we used the Kane-method, see e.g. [11], which allows to generate the equations for systems with an arbitrary number of helicopters. The coupling between equations for different helicopters is established by adding of one motion constraint (the length of the rope does not change during the time) for each helicopter in the compound. The resulting equations are used for control design as well as for simulation. For more on modeling see [13,3]. The aerodynamics of the load and ropes are currently neglected, but will be considered in future work for flights with high velocity.

3.2 Controller design

The general scheme of the proposed control algorithm for one or several helicopters coupled with the load is composed of two loops: the outer loop for translation control and inner loop to control the orientation of each helicopter, see Fig. 6.

The input of the control scheme in Fig. 6 is the desired trajectory $\mathbf{x}^*(t)$ for helicopters or for the load. The translational motion of each helicopter is controlled in the outer loop by controller $\mathbf{R}_{trans}$. Using the deviations from the desired trajectories the controller $\mathbf{R}_{trans}$ calculates for each helicopter i the forces $\mathbf{F}_i$ which should be generated by its rotors. The helicopter i can realize the force $\mathbf{F}_i$ by adjusting the absolute value of the main rotor lifting force F_{i3}^{MR} and adjusting the orientation of the main rotor plane or fuselage $q^*_{i\{1,2\}}$. The desired orientation of the main rotor plane is controlled in the inner loop by controller $\mathbf{R}_{ori}$. The values F_{i3}^{MR} and $q^*_{i\{1,2\}}$ are calculated using algebraic relations in block $\mathbf{F}^{-1}$.

As mentioned above, the rope can not be connected to the center of mass of the helicopter and the force in the rope produces torques imposed on the helicopter fuselage. Therefore the influence from the load or from the other helicopters transmitted to the helicopter fuselage by the rope, makes the orientation control a challenging task [12,13]. The developed approach for control of coupled helicopters is based on following three ideas:

– The design of the orientation controller (for each helicopter) accounts for the complete dynamics (translational and rotational) of the whole system, all coupled helicopters and the load. This is required due to the strong mutual coupling between translation and rotation of each helicopter and the load.
– The design of the translation controller is based on simplified model and accounts only for the translational dynamics of the whole system. In this simplified model the helicopters are modeled as mass points which can produce forces in direction perpendicular to current orientation of the main rotor plain.
– The usage of the force sensor in the ropes simplifies the design of orientation controller and makes it robust against variations of system parameters and disturbances.

The usage of the force sensor in the rope is a key issue in the proposed controller design. The force measured in the rope can be easily recalculated into resultant force and torque acting on the fuselage

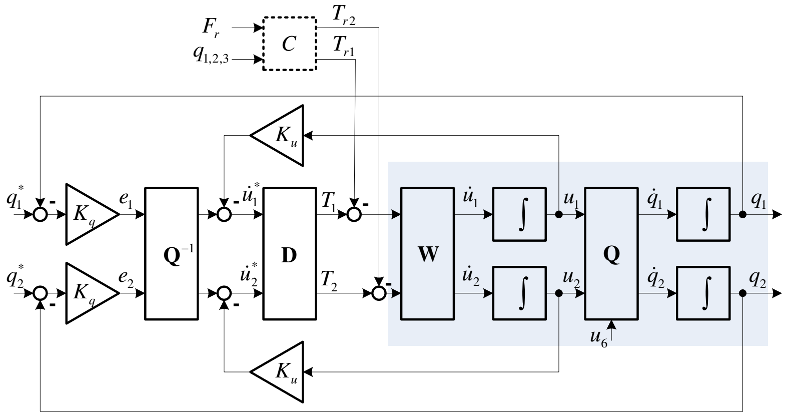


Fig. 7 Scheme for the orientation control

from the rest of the system. These two values are used in the feedback loop of orientation controller. The usage of the force sensor signal in the feedback loop has three main advantages:

1. the closed loop system becomes very robust against variation of system parameters and disturbances
2. the orientation controller for such complicated systems becomes quite simple
3. the orientation controller does not depend on the number of helicopters connected to the load.

As shown in [13], the rotation dynamics for a single helicopter, represented by two rigid bodies for the fuselage and the main rotor, can be expressed by the following equations:

$$T_1^{MR} + K_{12}u_2 + K_{11}\dot{u}_1 = 0 \tag{10}$$

$$T_2^{MR} + K_{21}u_1 + K_{22}\dot{u}_2 = 0 \tag{11}$$

Where $T_{1,2}^{MR}$ are the torques generated around the longitudinal and lateral axes of the fuselage, $u_{1,2}$ are rotation speeds of the fuselage and the coefficients K_{xx} are constant parameters of the helicopter and of the main rotor speed. We assume here that the influence of the rotation around the vertical axis (rotation speed u_3) on $u_{1,2}$ is small and can be considered as disturbance. This assumption is true if u_3 is hold on zero or on constant values using additional tail rotor controller with a time constant much smaller compared to the time constant of the orientation controller for $u_{1,2}$. Equations (10, 11) are coupled through $u_{1,2}$. This coupling leads to oscillations (for the parameters of typical small size helicopter) once the system has been stimulated. The scheme for the control of roll and pitch angles $q_{1,2}$ for a single helicopter is shown in Fig. 7, where the gray block denotes the model of helicopter rotational dynamics and kinematics. The controller is composed of blocks $\mathbf{Q}^{-1}$, $\mathbf{D}$ and two feedback loops with gains K_u, K_q for rotation speeds $u_{1,2}$ and orientation angles $q_{1,2}$ respectively. The rotation dynamics described by Eqs. (10), (11) are represented in Fig. 7 by the block $\mathbf{W}$. The block $\mathbf{D}$ of the controller is used to decouple the plant between $T_{1,2}^{MR}$ and $u_{1,2}$. The decoupling can be performed by means of known techniques from linear control theory, e.g. using matrix composed of compensating transfer functions. This orientation controller shows a good performance and robustness in simulation and real flight experiments with different types of helicopters as we have shown in [12,13].

The rotational dynamics of a helicopter coupled to the load by means of a rope are strongly influenced by the motion of the whole system. To account for this influence, block $\mathbf{D}$ should be replaced by the inverse rotational dynamics $\tilde{\mathbf{D}}$ not of a single helicopter, but of the whole system (considering both, the rotation and translation of each helicopter). With this new block $\tilde{\mathbf{D}}$, the orientation controller for helicopter coupled to the load shows equal performance as the orientation controller with block $\mathbf{D}$ for a single helicopter without load in case for nominal values of all system parameters ρ_i:

$$c_i = g(q_i, \bar{q}_i, \rho_i), \tag{12}$$

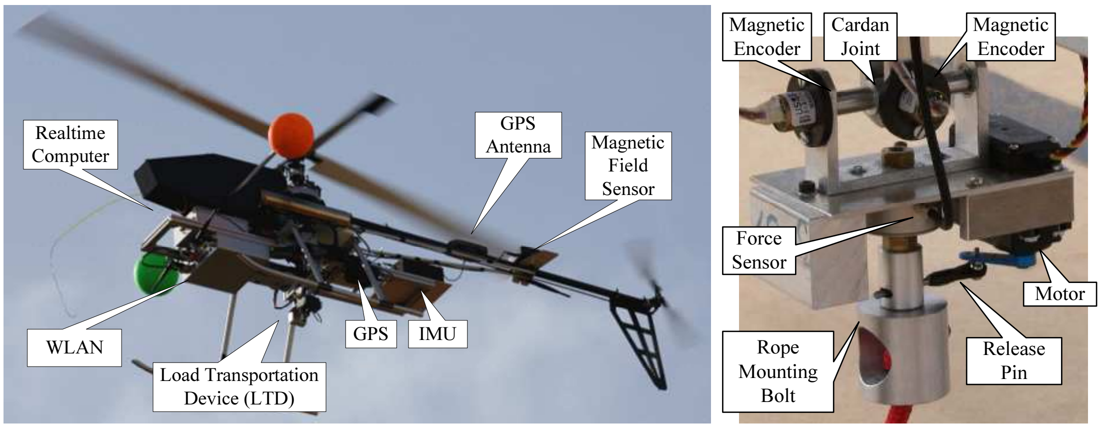


Fig. 8 UAV component diagram

The simulation experiments have shown that, unlike in the case of a single helicopter, the orientation controller with inversion block $\tilde{\mathbf{D}}$ for coupled helicopter is quite sensitive to variation of the system parameters ρ_i(5% variation could be critical). To overcome this problem, we propose to use a force sensor in the rope. The force $\mathbf{F}_r$, measured in the rope, will be used to calculate the influence on the rotational dynamics of the coupled helicopters from the remainder of the system. This influence is expressed by means of torque $\mathbf{T}_r = \mathbf{F}_r \times \mathbf{p}_{r-cm}$, where $\mathbf{p}_{r-cm}$ is the position vector connecting rope attaching point r and helicopter CoM. The resulting orientation controller

$$c_i = g(q_i, F_r), \tag{13}$$

is composed of the orientation controller for a single helicopter and the compensator block $\mathbf{C}$, see Fig. 7, where $\mathbf{T}_r$ is calculated and subtracted from torques calculated in $\mathbf{D}$. The usage of the compensator $\mathbf{C}$ allows us to decouple the control of each helicopter from the rest of the system and to use the same controller independent of the number of helicopters coupled together via the ropes.

There are two reasons for the robustness of the proposed orientation controller: First, the actual influence of the load on the fuselage is measured through $\mathbf{F}_r$ and, therefore, the compensation becomes independent from the mass of the load and the length of the rope. Second, as long as the orientation of the helicopter is known, the calculated compensation torque is always in the correct phase.

The details of the presented control algorithms can be found in [12,13,3,2].

3.3 System description

In Fig. 8 one of the TUB UAVs, used for the slung load transportation experiments, is shown during flight. The UAVs are based on commercially available small size helicopters. The helicopters have a rotor diameter of 1.8 m, a main rotor speed of approximate 1300 rpm and are powered by a 1.8 KW two-stroke engine. The UAVs can carry about 1.5 kg of additional payload, whereas the weight of the UAV itself is 12.5 kg. The different components necessary to achieve autonomous flight capabilities are mounted to the helicopters, using a frame composed of strut profiles. Through the use of these profiles, the location of hardware components can be altered and new hardware can be installed easily. This allows quick reconfiguration of the UAVs for different applications, easy replacement of defective hardware and alteration of the position of different components to adjust the UAVs center of gravity. The components necessary for autonomous operation are shown in Fig. 8: A GPS, an IMU, a control computer and a communication link. Due to the strong magnetic field of the engine a magnetic field sensor is mounted on the tail. All UAVs are equipped with Load Transportation Devices (LTD), which are specially designed

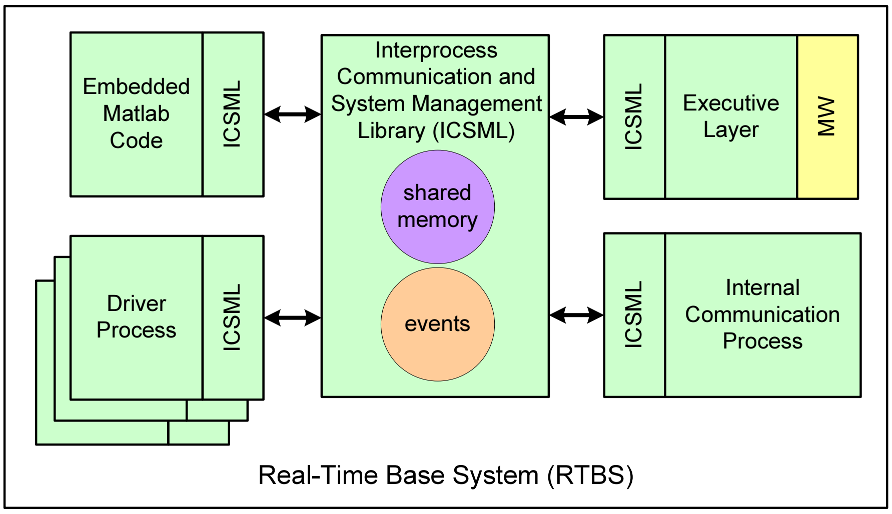

Embedded Matlab Code
ICSML
Interprocess Communication and System Management Library (ICSML)
shared memory
events
ICSML
Executive Layer
MW
Driver Process
ICSML
ICSML
Internal Communication Process
Real-Time Base System (RTBS)

Fig. 9 Real-Time Base System (RTBS)

for the transportation of slung loads using one or more UAVs (see Fig. 8). The LTD is composed of a two axis cardan joint with two magnetic encoders attached to each axis. After the joint a force sensor is mounted. After the force sensor a release mechanism for the rope is attached, which is composed of a bolt, inserted into a tube. The bolt is fixed in the tube through a pin, which can be pulled out by a small motor, to release the load. The release mechanism can be used for emergency decoupling of the UAV from the load (or the other coupled UAVs), but also to release the load after successful transportation. The magnetic encoders allow to measure the rope orientation relative to the UAV fuselage. With this information and the measured force in the rope, it becomes possible to calculate the torque and force imposed on the UAV through the load (and/or the other coupled UAVs), which are used in the feedback loop of the compensator block **C**, see Sec. 3.2.

In Fig. 9 the concept of the software system running on the UAV board computers is shown. The Real-Time Base System is of composed multiple separate modules, which are connected by a Interprocess Communication and System Management Library (ICSML). The ICML offers, among other features, convenient interprocess communication methods, using a shared memory design and a event system for interprocess notifications (e.g. on the change of shared memory variables). The driver modules are used to communicate with the peripheral devices of the UAV, such a compass, IMU or GPS. The device specific protocol is encapsulated in the driver process and the sensor data are presented to the remaining system through ICSML as generalized objects. The controller of the system was designed and tested in simulation using Matlab/Simulink. The RTBS provides a generic wrapper module, to embed C-Code generated by Matlab/Simulink. This allows fast implementation of new control algorithms and an error free transition from Simulink models to real-time controller code. The internal communication process is used for development and testing purposes and for experiments without the AWARE system. As explained in Sec. 2 the executive layer is the proprietary interface to the AWARE On-Board Deliberative Layer (ODL). The executive layer module provides the basic functionalities needed by the ODL. This includes the commanding of the UAVs, reporting of the command execution progress and the UAV state. Additionally the executive layer module provides a certain level of hardware abstraction: The relative configuration of the UAVs during the load transportation is part of the control loop and shouldn't be altered by the ODL. Therefore, during the load transportation the coupled UAVs are presented to the ODL as a single entity. This way the commanding interfaces for coupled and uncoupled UAVs are equal.

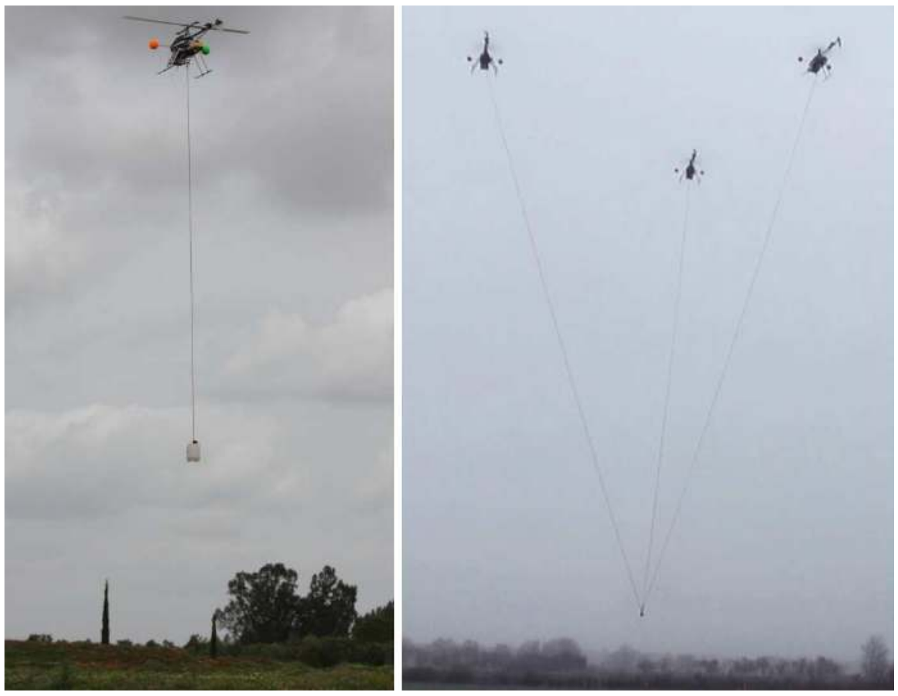


Fig. 10 Single and multiple UAV slung load transportation

3.4 Experimental results

The load transportation system has been used several times in different experiments, as stand alone system as well as a part of AWARE platform. In Fig. 10 two load transportation tasks are shown. The picture on the left with one helicopter transporting the load was taken in Utrera (Spain), April 2008. The first successful experiment was conducted by the authors in Berlin, November 2007 (http://www.-youtube.com/watch?v=J3gmzn0bSa4). The picture on the right with three helicopters transporting the load was taken on our test ground near Berlin, December 2007 (http://www.youtube.com/watch?v=-tl6DYWNe9ac). For these flight experiments three identical helicopters as described in Sec. 3.3 were used. The helicopters are equipped with a multi-UAV modular autopilot system developed at TUB. The rope is attached to the helicopter by means of the Load Deployment Device (LDD) which is mounted between the landing skids.

In the first experiment one helicopter transported a load of 1 kg using a rope of 5 m length. Fig. 11 (left) shows part of the system state during this experiment. All coordinates are given in an ground fixed frame with the origin at the helicopter take-off position. Coordinate x_3 represents the helicopter altitude. Until the load was dropped the coordinates $x_{1,2}$ show the position of the load and after the load dropping, the position of the helicopter itself. The dropping is indicated by a vertical slashed line. The gray line denotes the desired position given to the controller. We consider the performance of the controller as quite good, despite the stormy weather conditions on that day. Even with steady wind of 30 km/h and wind gusts up to 40 km/h the controller was able to stabilize the helicopter and damp upcoming oscillation of the load.

In the second experiment a load of 4 kg was transported by means of three helicopters. In this experiment ropes with a length of 13 m were used. The helicopters were arranged as a equilateral triangle on the ground, with a distance of 8 m between the helicopters. In Fig. 11 (right) the coordinates of all three helicopters during the whole flight are shown. The coordinates of the helicopters are shown in different ground fixed frames, which have the same orientation, but different origins (take-off position of

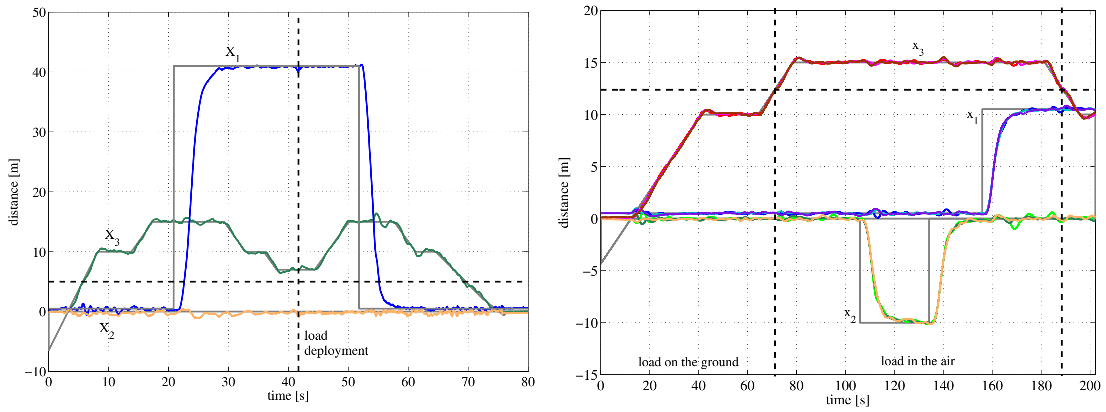


Fig. 11 Motion of the load (1 kg) during transportation using one helicopter (left) and motion of three helicopters transporting one load of 4 kg (right)

each helicopter), therefore there are no offsets between the helicopter trajectories. The load was lifted when the helicopters reached approximate 12.4 meters. The weight of the load was not considered in the controller and therefore a small disturbance in the x_3 trajectories can be observed at the moment the load was lifted from the ground as well as during strong acceleration in $x_{1,2}$-direction. A position error of each helicopter in hovering was approx. ± 0.3 m During the whole flight the triangular formation of the helicopters with a precision of about ± 0.3 meters and the load was moved very smoothly. To our knowledge these were the very first successful flight experiments to transport a load using multiple autonomous helicopters.

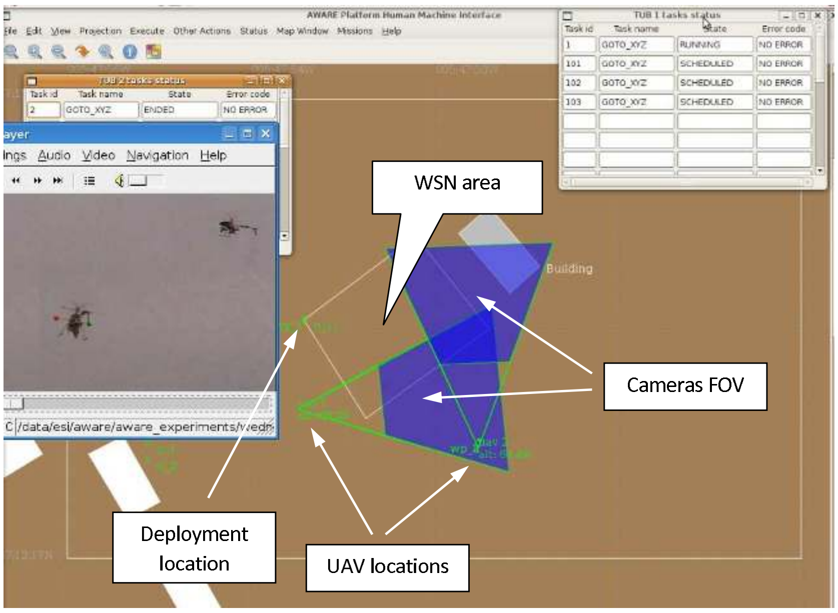
4 Multi-UAV deployment of multiple loads: experimental results

In this section, in order to illustrate the architecture of the AWARE platform, the execution of a real mission consisting in the deployment of sensor nodes and the monitoring of the scene by the helicopter team is described. In this case, the helicopters are not physically coupled but at least one helicopter carries a device that is able to deploy several sensor nodes.

This mission was performed in 2008 during the second year of the AWARE project (see www.aware-project.net/videos/videos.shtml). The objective was to deploy a sensor node from an UAV in a given location to repair the WSN network connectivity, whereas another UAV supervised the operation with the on-board camera and also monitored the area for the deployment and a nearby building. During the mission, a third UAV took-off to monitor the whole operation. It should be mentioned that all the UAVs of the AWARE platform are autonomous helicopters.

The following systems were involved in the execution:

– HMI (Human Machine Interface) station: allowed to specify the mission (waypoints with their corresponding headings, speeds, altitudes, etc). After that, there was a mission briefing with the operators of the UAVs using a simulation of the execution in the HMI to tune several parameters of the mission using EUROPA. Once an agreement was met, the mission was ready to be executed. During the execution, the HMI allowed to supervise the execution state of the different tasks allocated to the UAVs. Figure 12 shows a screenshot of the HMI application during the experiment.
– ODL (On-board Deliberative Layer) software of the UAVs: performing the function described in Sect. 2.
– UAV supervisor and executive software components: allowed to supervise the elementary tasks sent by the ODL in order to have another check point of their consistency. Finally, the executive software on-board the helicopter was in charge of the final execution of the elementary tasks.


AWARE Platform Human Machine Interface
File Edit View Projection Execute Other Actions Status Map Window Missions Help
ayer
ings Audio Video Navigation Help
C/data/esi/aware/aware_experiments/vedr
Building
WSN area
Cameras FOV
Deployment location
UAV locations

TUB 2 tasks status

Task id	Task name	State	Error code
2	GOTO_XYZ	ENDED	NO ERROR

TUB 1 tasks status

Task id	Task name	State	Error code
1	GOTO_XYZ	RUNNING	NO ERROR
101	GOTO_XYZ	SCHEDULED	NO ERROR
102	GOTO_XYZ	SCHEDULED	NO ERROR
103	GOTO_XYZ	SCHEDULED	NO ERROR

Fig. 12 The HMI screen during the experiment: visualization of the locations and status of the TUB helicopters with their allocated elementary tasks.

- WSN gateway: In this experiment, the gateway was in charge of autonomously repairing the connectivity of the network once a node was placed between two isolated networks. The system was able to automatically re-compute the data routing and to send all the information from the nodes to the HMI.
- Middleware: allowed to communicate the tasks status, telemetry and images, and also to send high level commands from the HMI to the ODL and commands from the ODL to the UAV executive layer.

4.1 Mission description

The mission was specified by the AWARE platform user using the HMI software. It was a node deployment mission to repair the WSN connectivity involving three UAVs:

- UAV 1: equipped with a fixed camera aligned with the fuselage of the helicopter and pointing downwards 45° .
- UAV 2: it has a device on-board for node deployment (see Fig. 13) and also a camera.
- UAV 3: equipped with a camera mounted inside a mobile gimbal to film the whole mission from different point of views, transmitting images to the HMI through the middleware.

The AWARE platform user specified the following tasks to be executed (see Table 6):

- τ_1: deploy a node in the waypoint `wp_1` with GPS geographical coordinates (-5.79816709, 37.20502859) after τ_2 is completed. The goal was to repair the connectivity with the node with identifier 28 deploying the node 11 in `wp_1`.
- τ_2: visit waypoint `wp_2` (-5.79790241 37.20488147) to monitor a nearby building.
- τ_3: During the node deployment, the corresponding WSN area should be also monitored with the camera from `wp_2`.
- τ_4: After node deployment, the building should be monitored again from `wp_2`.

In the next section, the role of the different modules in the ODL architecture is described.

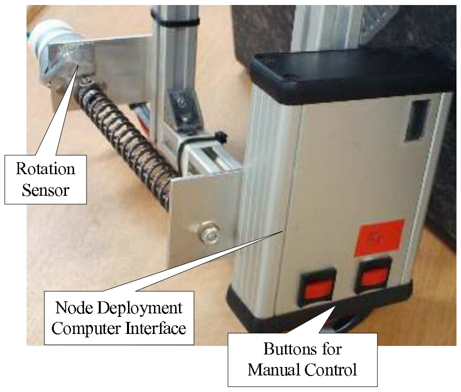

Rotation Sensor
Node Deployment Computer Interface
Buttons for Manual Control

Fig. 13 Detail of the device on-board the helicopter for the node deployment operation.

Table 6 Tasks specified for the mission. The values of the parameters (Π_k) are detailed in Table 7.

τ_k	λ_k	$^{-}\Omega_k$	Ω_k^+	Π_k
τ_1	DEPLOY	END(τ_2)	$\emptyset$	Π_1
τ_2	GOTO	$\emptyset$	$\emptyset$	Π_2
τ_3	GOTO	START($^{101}\hat{\tau}_1$)	$\emptyset$	Π_3
τ_4	GOTO	END($^{103}\hat{\tau}_1$)	$\emptyset$	Π_4

Table 7 Values for the tasks parameters (Π_k).

Parameters (Π_k)	Π_1	Π_2	Π_3	Π_4
π^1	-5.79816709	-5.79790241	-5.79790241	-5.79790241
π^2	37.20502859	37.20488147	37.20488147	37.20488147
π^3	67.0	67.0	67.0	67.0
π^4	1.1	1.1	1.1	1.1
π^5	1	1	1	1
π^6	90	0	-45	0
π^7	1	0	0	0

4.2 ODL modules during the mission

As the user did not allocate those tasks manually, the distributed task allocation process started from the HMI software. The negotiation involved the CNP manager modules of UAV 1 and UAV 2, and due to the different devices on-board each UAV, task τ_1 was allocated to UAV 2 whereas the rest of tasks were allocated to UAV 1 (bids with infinite cost for task τ_1).

In this mission, the plan builder role was trivial: for both UAVs a take-off was required before executing the allocated tasks and the ordering of the tasks was fixed by the preconditions.

Then, the plan refining module of UAV 2 decomposed τ_1 as follows:

$$\tau_1 \longrightarrow \{^1\hat{\tau}_1, ^{101}\hat{\tau}_1, ^{102}\hat{\tau}_1, ^{103}\hat{\tau}_1\} \tag{14}$$

with:

- $^1\hat{\tau}_1$: visit wp_1.
- $^{101}\hat{\tau}_1$: descend to an altitude of 3 meters above the ground.
- $^{102}\hat{\tau}_1$: activate the device for node deployment.
- $^{103}\hat{\tau}_1$: ascend back to the initial altitude.

whereas in UAV 1, it computed the headings required for monitoring the WSN area and the node deployment operation with the fixed camera from wp_2.

During the mission, the interface with the executive layer was the task manager. For example, once the executive layer changed the status of ${}^{1}\hat{\tau}_1$ from `RUNNING` to `ENDED`, the task manager sent the next task (${}^{101}\hat{\tau}_1$) for its execution. Moreover, the dependencies between tasks of different UAVs were also handled by the task manager with the assistance of the synchronization module. On the other hand, the plan merging modules running the algorithm shown in Section 2.6 did not detect any conflict between the planned 4D trajectories of the UAVs and no change in the plan was inserted.

The resulting evolution of the tasks during the mission is shown in Figure 14, where the different preconditions have been represented by arrows.

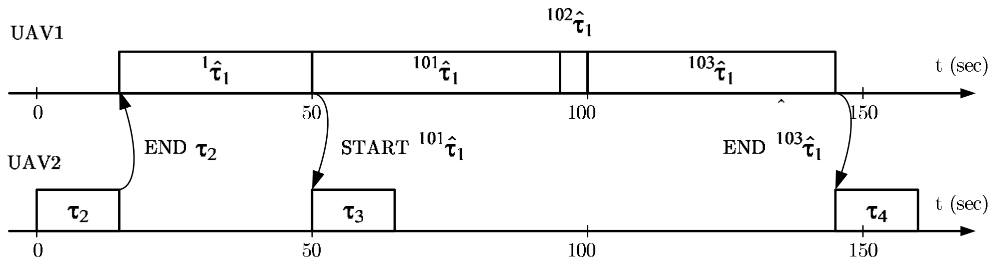


Fig. 14 Tasks executed by each UAV. The arrows represent the different preconditions summarized in Table 6.

Finally, it should be mentioned that this experiment represents the first test of the AWARE platform integrating the three helicopters and the wireless sensor network in the same mission. Regarding the WSN, the connectivity with node 28 was achieved a few seconds after node 11 was deployed. Figure 15 shows several photographs taken during the experiment.

Fig. 15 Coordinated flights during the sensor node deployment mission (see http://www.aware-project.net/videos/videos.shtml).

5 Conclusions

The cooperation of multiple autonomous aerial vehicles is a very suitable approach for many applications, including detection, precise localization, monitoring and tracking in emergency scenarios. In these applications the UAVs do not modify the state of the environment and there are no physical interactions between the UAV and the environment. Furthermore, the interactions between the UAVs, are essentially information exchanges without physical couplings between them. This paper demonstrates the possibilities of the cooperation and control of multiple aerial robots with sensing and actuation capabilities allowing load deployment (in particular sensor nodes deployment).

Furthermore, the paper presents the multi-UAV load transportation, which requires the consideration of physical interactions between the aerial robots. The paper has presented a multi-UAV architecture developed in the AWARE project that allows different level of interaction among the UAVs and between the UAVs and the environment, including both sensing and actuation. The interaction between all the systems in this architecture can be represented by means of an hybrid system formulation coping with both discrete events associated to the tasks to achieve a given mission and the continuous dynamic interactions between physically coupled UAVs.

Particularly, the paper has presented results obtained in the AWARE project demonstrating the lifting and transportation of a slung load by means of one helicopter and also by three coupled helicopters, which has been the first demonstration of this challenging application. Finally, the paper also presents a multi-UAV mission consisting of the deployment of the sensor nodes of a WSN.

The proposed methods open many different new opportunities in missions involving the cooperation of multiple UAVs for applications such as search and rescue and interventions in disaster management and civil security. The transportation of loads by means of the UAVs can be also considered as a first step toward the cargo transportation by means of UAVs.

Acknowledgements The authors would like to thank the AWARE Project reviewers Fernando Lobo Pereira and Ørnulf Jan Rødseth for their very constructive advice and orientation in the project.

References

1. Banâtre, M., Marron, P., Ollero, A., Wolisz, A.: Cooperating Embedded Systems and Wireless Sensor Networks. John Wiley & Sons Inc. (2008)
2. Bernard, M., Kondak, K.: Generic slung load transportation system using small size helicopters. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3258–3264 (2009). DOI 10.1109/ROBOT.2009.5152382
3. Bernard, M., Kondak, K., Hommel, G.: Load transportation system based on autonomous small size helicopters. In: 23rd Int. Unmanned Air Vehicle Systems Conf. (2008)
4. Borenstein, J.: The Omnimate: A guidewire- and beacon-free AGV for highly reconfigurable applications. International Journal of Production Research **38**(9), 1993–2010 (2000)
5. Chaimowicz, L., Kumar, V., Campos, M.F.M.: A paradigm for dynamic coordination of multiple robots. Autonomous Robots **17**(1), 7–21 (2004)
6. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3602–3608 (2004)
7. Dias, M.B., Stenz, A.: Opportunistic optimization for market-based multirobot control. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2714–2720. Lausanne, Switzerland (2002)
8. Fierro, R., Das, A., Spletzer, J., Esposito, J., Kumar, V., Ostrowski, J.P., Pappas, G., Taylor, C.J., Hur, Y., Alur, R., Lee, I., Grudic, G., Southall, B.: A framework and architecture for multi-robot coordination. International Journal of Robotics Research **21**(10–11), 977–995 (2002)
9. Huntsberger, T.L., Trebi-Ollennu, A., Aghazarian, H., Schenker, P.S., Pirjanian, P., Nayar, H.D.: Distributed control of multi-robot systems engaged in tightly coupled tasks. Autonomous Robots **17**(1), 79–92 (2004)
10. Johnson, W.: Helicopter Theory. Dover Publications, Inc. (1980)
11. Kane, T., Levinson, D.: Dynamics: Theory and Applications. McGraw-Hill Book Company (1985)
12. Kondak, K., Bernard, M., Losse, N., Hommel, G.: Elaborated modeling and control for autonomous small size helicopters. In: ISR/ROBOTIK 2006 Joint conference on robotics (2006)

13. Kondak, K., Bernard, M., Meyer, N., Hommel, G.: Autonomously flying VTOL-robots: Modeling and control. In: IEEE Int. Conf. on Robotics and Automation, pp. 2375 – 2380 (2007)
14. Kosuge, K., Sato, M.: Transportation of a single object by multiple decentralized-controlled nonholonomic mobile robots. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, vol. 3, pp. 1681–1686 (1999)
15. Li, H., Karray, F., Basir, O., Song, I.: A framework for coordinated control of multiagent systems and its applications. IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans **38**(3), 534–548 (2008)
16. Maza, I., Ollero, A.: Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In: Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems, pp. 211–220. Toulouse, France (2004)
17. Merino, L.: A cooperative perception system for multiple unmanned aerial vehicles. Application to the cooperative detection, localization and monitoring of forest fires. Ph.D. thesis, University of Seville (2007)
18. Mittal, M., Prasad, J.V.R., Schrage, D.P.: Nonlinear adaptive control of a twin lift helicopter system. IEEE Control Systems Magazine **11**(3), 39–45 (1991)
19. Ollero, A., Maza, I.: Multiple heterogeneous unmanned aerial vehicles. Springer Tracts on Advanced Robotics. Springer-Verlag (2007)
20. Ollero, A., Merino, L.: Control and perception techniques for aerial robotics. Annual Reviews in Control **28**(2), 167–178 (2004)
21. Reynolds, H.K., Rodriguez, A.A.: H_∞ control of a twin lift helicopter system. In: Proceedings of the 31st IEEE Conference on Decision and Control, pp. 2442–2447 (1992)
22. Smith, G.: The Contract Net Protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers **29**(12) (1980)
23. Sugar, T., Kumar, V.: Decentralized control of cooperating mobile manipulators. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 4, pp. 2916–2921 (1998)
24. Sukkarieh, S., Nettleton, E., Kim, J.H., Ridley, M., Goktogan, A., Durrant-Whyte, H.: The ANSER Project: Data Fusion Across Multiple Uninhabited Air Vehicles. The International Journal of Robotics Research **22**(7-8), 505–539 (2003). DOI 10.1177/02783649030227005
25. Viguria, A., Maza, I., Ollero, A.: SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3339–3344. Rome, Italy (2007)
26. Viguria, A., Maza, I., Ollero, A.: S+T: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3163–3168. Pasadena, California, USA (2008)