Journal of Software Engineering for Robotics 5(2), September 2014, 1-12
ISSN: 2035-3928

Control Software Architecture for Cooperative Multiple Unmanned Aerial Vehicle-Manipulator Systems

Gianluca Antonelli[1] Khelifa Baizid [1] Fabrizio Caccavale[2]
Gerardo Giglio[2] Giuseppe Muscio[2] Francesco Pierri[2]

[1] Dipartimento di Ingegneria Elettrica e dell'Informazione, Università di Cassino e del Lazio Meridionale, Via Di Biasio 43, 03043 Cassino, Italy
[2] Scuola di Ingegneria, Università degli Studi della Basilicata, via dell'Ateneo Lucano, 10, 85100 Potenza, Italy

Abstract—In this paper a Control software Architecture for cooperative multiple unmanned aerial Vehicle-manIpulator Systems (CAVIS) is presented. In particular, Unmanned Aerial Vehicles (UAVs) equipped with a robotic manipulator eventually involved in a cooperative object transportation mission are considered. Such robotic systems involve specific constraints given by the interaction of the degrees of freedom of the arms; for example, obstacle avoidance, joint limits, kinematic singularities need to be properly handled together with the main objective of controlling the arm's end-effector to transport the object. To this end, a proper behavioral approach has been tailored and implemented with the aim to provide modularity and flexibility of use to the architecture. A case study shows the performance of the proposed software architecture.

Index Terms—Control software architecture, Unmanned aerial vehicle manipulator systems, multi-robot systems, coordination.

1 Introduction

In the last two decades multi-robot systems have played an important role in several robotics applications [1] [2] [3]. They can be composed by different types of robotic units, such as aerial, ground or marine vehicles. In particular, Unmanned Aerial Vehicles (UAVs) can be used in several domains, such as, e.g., monitoring, aerial mapping, terrain inspection, traffic surveillance, rescue missions and cadastral applications [4], [5], [6], [7]. Manipulation capabilities of aerial robots can be provided by a robotic arm mounted on the floating base (Unmanned Aerial Vehicle-Manipulator Systems, UAVMSs) [8]. In addition, systems composed of multiple UAVMSs can be adopted to carry out more complex missions such as, e.g., cooperative transportation of large and/or heavy payloads, cooperative assembly of structures in remote and/or hazardous environments, cooperative inspection and mapping. A notable example is given by the EU-funded ARCAS (Aerial Robotics Cooperative Assembly System) project [9], aimed at developing one of the firsts cooperative free-flying robot system for assembly and structure construction (Fig. 1).

Figure 1. Application scenario envisioned in the framework of the ARCAS project.

Since multi-robot systems are employed in a wide range of applications, the design and implementation of Control Software Architectures (CSAs) dedicated to such systems is a

Regular paper – *Manuscript received December 06, 2013; revised Jun 29, 2014.*

- *This work was supported European Communitys 7^{th} Framework Program (No.287617) (IP project ARCAS - Aerial Robotics Cooperative Assembly system).*

- *Authors are in alphabetical order.*

challenging issue, given the complexity and the heterogeneity of such systems. Indeed, several efforts have been spent on the design and the development of software/hardware platforms supporting multi-robot systems, leading to notable results such as URBI [10], MRDS [11], Player/Stage [12], ROS [13] and OROCOS [14].

Considering aerial robotics, recently, many software applications have been developed, such as Hector Quadrotor [15], ARDron [16] (and the related ARDrone API [17]) for robust navigation in dynamic environment, TeleKyb [18] for UAV control. Some works addressed multi-UAV systems, e.g., [5], [6] and [19], although the presence of a manipulator was not considered.

Evaluation of the above mentioned software architecture is not easy, due to the huge variety of application scenarios [20], [21]. Many of them, certainly, may be considered versatile, but the implementation of such CSAs might require development of new software components, to cope with additional specifications imposed by particular application scenarios and robotic systems, such as multi-UAVMSs. On the other hand, embedding a suitable control methodology in the CSA is a critical step for systems having a relatively large number of Degrees Of Freedom (DOFs) to be handled, especially when safety issues and physical interaction between robots (and/or with the external environment) have to be considered. Moreover, tuning the control software through a reliable simulation environment is another issue to be tackled in the design of the CSA, since there are many dynamic and kinematic constraints to be considered in the design phase of the controller, see e.g., [22]. Indeed, the development of reliable and effective CSAs for these systems that consider all above issues, is a very challenging topic, which must be well addressed to the earlier stages of the software design [5], [6], [21]. On the other hand, task management and mission planning for cooperative multi-UAVMSs must be taken into account as well [5]. Recently, several research works adopted concepts such as hierarchical decomposition of missions planning [23], where the mission can be decomposed into a set of basic behaviors (i.e., a library of software functions). When performing complex tasks, however, combining many simple behaviors might be necessary.

Therefore, it can be concluded that the current state of the art in CSAs for multi-robot systems does not provide complete solutions to cope with all the requirements needed by multiple UAVMSs. In the following, a Control software Architecture for cooperative unmanned aerial Vehicle-manIpulator Systems (CAVIS) is developed. CAVIS is aimed at supporting the design and the implementation of behavioral control schemes for multiple UAVMSs. To deal with CSA design complexity, the concept of *decomposition of the system into components* has been applied, which is a critical issue for software architecture design [24], [25]. In detail, the architecture is based on a hierarchical arrangement of the main components. Namely, at the lowest level, a set of basic tasks, named *elementary behaviors*, is defined; those are then combined into complex tasks, named *compound behaviors*, according to the Null-Space based Behavioral (NSB) approach [2], [23], [26]. The developed tasks' library allows to cover most aerial manipulation scenarios. At a higher abstraction level, compound behaviors are grouped into *actions*, while a *mission* can be composed by a temporal sequence of actions.

The current version of the CSA has been developed under Matlab/Simulink® VPL (Visual Programing Language) and profit from SimMechanics® for simulation and scene rendering. A preliminary version of the developed architecture has been ported to the experimental set-up available at FADA-CATEC in Sevilla [27] by means of the Real Time Workshop® [28] [29], and tested experimentally on a single UAVMS. Some videos of the experiments are available at http://webuser.unicas.it/lai/robotica/video.html#UAVMS.

The paper is organized as follows: the second section provides a general description of the proposed software architecture; the third section provides a bottom-up description of the main modules; the fourth section provides an application example, involving a cooperative transportation of an object by two UAVMSs; lesson learned and conclusions end the paper. Finally, some mathematical details of the control algorithms developed in CAVIS are provided in the Appendix.

2 CAVIS: A GENERAL DESCRIPTION

2.1 Main features

Figure 2 illustrates the main CAVIS's features, i.e., the possibility to handle multiple heterogeneous UAVs/UAVMSs, eventually in a collaborative scenario, by providing a library of behaviors properly handled. To summarize, the CAVIS features are:

1) it provides a flexible and modular environment for UAVMSs applications;
2) it allows to cope with multiple entities, eventually heterogeneous and/or cooperative UAVs/UAVMSs;
3) it provides simulation framework for the design and the tuning of the control software;
4) hardware-in-the-loop tests before porting to the hardware; can be carried out in the same framework used to develop the controller.

2.2 Definitions and main components

The proposed CSA is composed by several modules (Fig. 3), briefly described below:

- **Interface**: it is a Graphical User Interface that allows the user to input all the mission's details such as, for example, the initial and final configuration for the UAVMSs and the intermediate configurations (see Fig. 4).
- **Planner**: it generates off-line the plan for a given mission.
- **Display**: it allows to define and customize all 3D graphic objects, including the environment objects, UAVs and manipulators.

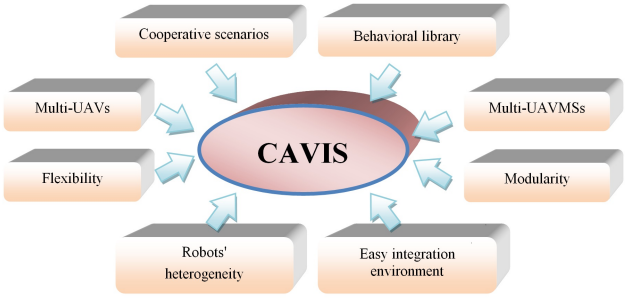


Figure 2. Main features of CAVIS architecture.

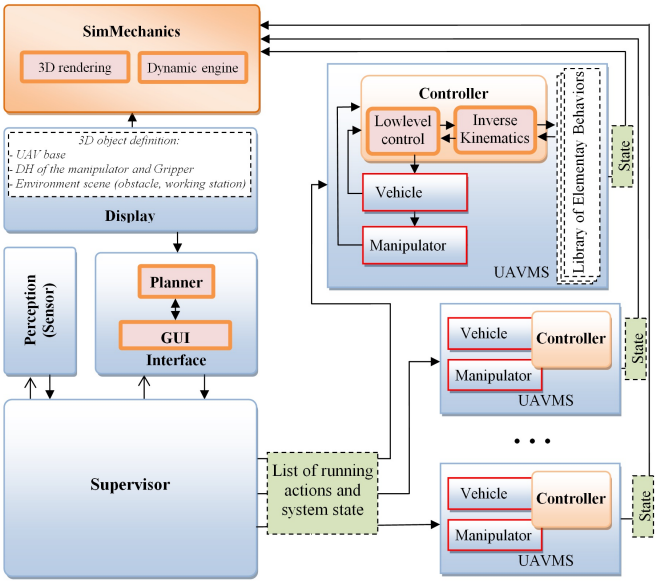


Figure 3. CAVIS's main software components.

- **Supervisor**: it implements one of the core concepts handled in this paper, since it provides task selection, coordination and synchronization.
- **Controller**: it implements the equations related to the selected compound behavior according to the Null-Space-based Behavioral control described in the Appendix. Its output is the reference velocity for each DOF to be sent to the low level (dynamic) control.
- **Perception**: it provides the state of the UAVMSs and information about the environment. Depending on the specific case, it can be centralized or partly implemented on-board each vehicle.

3 Bottom-up description of the components

Each UAVMS has to handle several control objectives simultaneously: for example, it has to move the end-effector and, at same time, to avoid both obstacles and mechanical joint limits. Each of this control objectives is defined as an *elementary behavior*. Within the framework of Null-Space-based Behavioral control, the elementary behaviors may be properly composed within *compound behavior* in a priority order. At a higher abstraction level, a set of compound behaviors can be grouped in an *action*, in such a way to represent a basic motion of the UAVMS. For each vehicle, a *sub-mission* lists the sequence of actions to be run. Finally, at the top, the *mission* represents the overall goal for the multi-agent system.

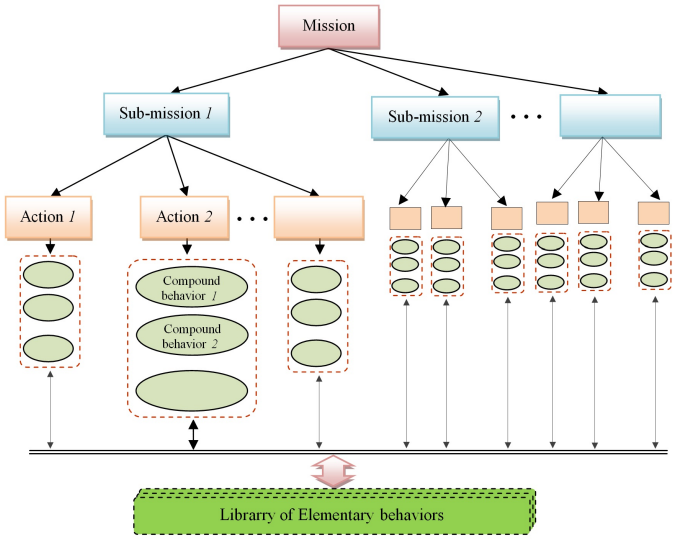


Figure 5. Hierarchical decomposition of *Mission*, *Sub-mission*, *Action* and *Compound behavior*.

Figure 5 represents the hierarchy that describes the relationship among the above-defined concepts. Notice that the elementary behaviors are embedded in compound behaviors.

3.1 Elementary behaviors

An elementary behavior is expressed through a function that relates the system's DOFs to the variable to be controlled. The proposed approach handles behaviors at the kinematic level, according to details given in the Appendix. It is possible to classify the elementary behaviors according to two aspects, one related to the DOFs involved, namely: the sole vehicle, the arm joint-space, the arm end-effector and the coordination/cooperation of the handled object. The second classification, based on the type of controlled variables, is in 3 groups: *safety behaviors*, namely those related to the safety of the system, *functional behaviors*, namely behaviors aimed at assigning a motion to the system, and *optimization behaviors*, namely those aimed at optimizing some indices (e.g., robot manipulability).

In the following, a possible list of elementary behaviors is provided with a brief description, while the expression of the task functions can be found in the Appendix. It is worth noting that this is not an exhaustive overview and new elementary behaviors can be defined on the basis of the designer's needs.

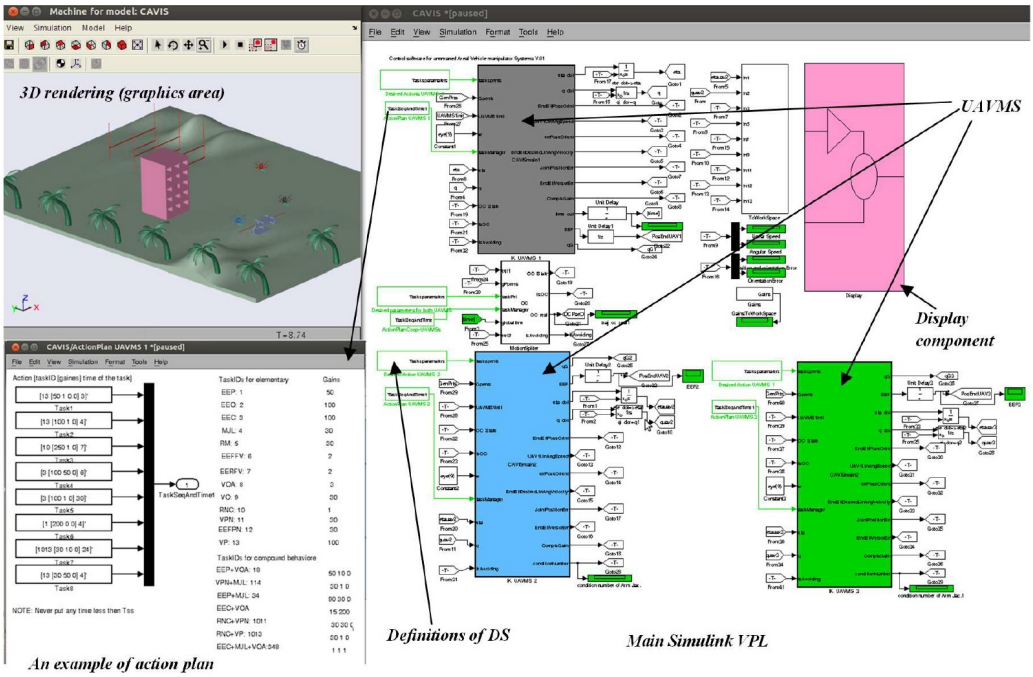


Figure 4. Main Matlab/Simulink® windows.

- Vehicle Position (VP): This elementary behavior acts at the vehicle level and controls the vehicle position. It is useful when the arm's motion is not of concern, since it is kept in a given home configuration (e.g., folded on itself).
- Vehicle Yaw (VY): This elementary behavior is aimed at controlling the yaw angle,
- Vehicle Obstacle Avoidance (VOA): This behavior is designed to prevent undesired collisions between the vehicle and obstacles present along the planned path. It controls the distance between the vehicle center of mass and the obstacle.
- Mechanical Joint Limit (MJL): Any manipulator exhibits mechanical limits for the joints mobility. It is appropriate to define a behavior that keeps the arm's configuration far from such limits.
- Robot Manipulability (RM): Dexterity of a robotic manipulator can be defined in many ways as a function (manipulability measure) of the joints configuration. This elementary behavior encodes the chosen manipulability measure and tries to maximize it.
- Robot Nominal Configuration (RNC): In some cases it is required to set the robotic arm to a specific configuration, e.g., the arm folded on itself during take-off and landing. This behavior is aimed at controlling the arm position in the joint space.
- End-Effector Position (EEP): This elementary behavior controls the position of the manipulator end-effector.
- End-Effector Orientation (EEO): This elementary behavior controls the orientation of the manipulator end-effector.
- End-Effector Configuration (EEC): This elementary behavior is aimed at controlling the manipulator end-effector position and orientation, simultaneously.
- Inter-Vehicle Distance (IVD): In order to avoid collisions between UAVMSs, it is very useful to keep a safety distance between each couple of UAVMSs.
- Object Configuration (OC): This behavior is aimed at controlling the motion of an object grasped by two or more UAVMSs along an assigned trajectory.
- Object Obstacle Avoidance (OOA): In the presence of obstacles during the cooperative transportation, the whole UAVMSs team should be able to hold the shared object and avoid the obstacle. Hence, this behavior controls the distance between a point of the system and the obstacle.

Figure 6 shows the matrix corresponding to the classification by resorting to the above-defined acronyms.

3.2 Compound behaviors

In case that the DOFs of the UAVMS are more than those required by the task function, the system is kinematically redundant and the redundant DOFs can be exploited to achieve additional tasks by resorting to a task-priority approach such as the NSB control [2], [26].

The elementary behaviors can be arranged in a defined priority order, according to the needs of the mission to be accomplished by the multi-UAVMS. The hierarchical combination of a set of elementary behaviors is defined as *compound behavior* (see Fig. 7). The priority order for elementary

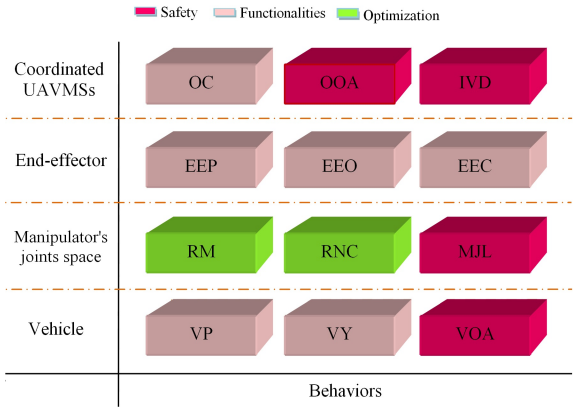


Figure 6. Classifications of some elementary behaviors according to the main control levels of the robotic system.

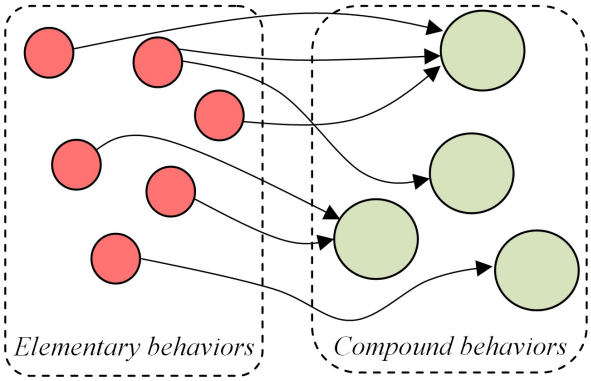


Figure 7. A scheme illustrating the relationship between elementary and compound behaviors.

behaviors depends on practical considerations (e.g., safety behaviors, as obstacles avoidance, should have always higher priority) or on a design choice. It is worth noticing that two compound behaviors can differ only for the priority order of its elementary behaviors.

An elementary behavior with a not-highest priority can be achieved only if it is compatible with the higher-priority ones, i.e., a low-priority behavior is achieved only for those components not conflicting with the higher-priority ones. A rigorous analysis of the compatibility issues of elementary behaviors can be achieved on the basis of the concepts of Jacobians orthogonality and independency [30].

From the software perspective, each elementary behavior corresponds to a function. The user only needs to know the list of compound behaviors with corresponding input-output. The function of an elementary behavior may be called by several functions of compound behaviors, thus implementing code re-use and improve the debugging phase.

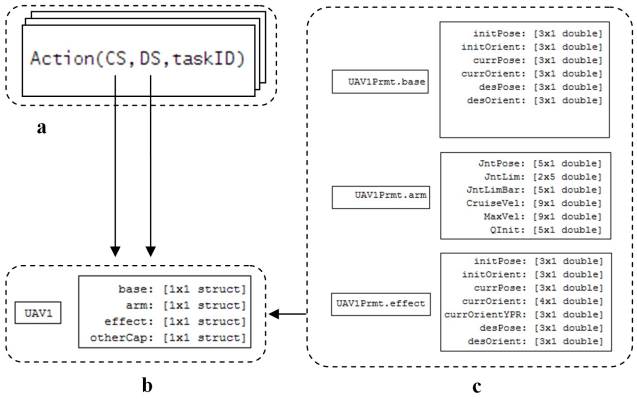


Figure 8. Action call syntax implemented in CAVIS.

3.3 Actions

Actions are used to raise the level of abstraction of the problem description and simplify the communication among developers. An *action* is a set of compound behaviors that are *logically* related to the current objective. A certain compound behavior may belong to different actions.

From a practical perspective, the Supervisor first selects the current action and then, among the compound behaviors belonging to the action, the proper one to be used in the NSB equation.

The actions are logically grouped within categories similar to the ones presented for the elementary behaviors. Thus, there are actions grouping all the compound behaviors to move the sole vehicle, actions collecting the compound behaviors related to the motion of the vehicle and the arm in free space, i.e., with no interaction nor cooperation, and, finally, actions concerning the cooperative/coordinated motion.

The output of the action is the desired velocities for the vehicle and the arm, to be sent to the low level dynamic controller. Its input is the Current State (CS) and the Desired State (DS), as schematically shown in Fig. 8 (*taskID* corresponds to the compound behavior selected by the Supervisor).

3.4 Mission/Sub-mission

Each mission is decomposed into several sub-missions, each one dedicated to a single UAV/UAVMS. A sub-mission is composed by a sequence of actions. For example, a mission that involves two UAVMSs transporting an object is composed by two sub-missions, each of them includes four successive actions: move the vehicle toward the bar, pre-grasp, i.e., reconfigure the system, grasp, move the object.

The actions of different UAVMSs are synchronized by means of a semaphore-like mechanism. Those requiring physical interaction between team-mates need to be executed simultaneously, otherwise a *weak* temporal synchronization is implemented. For example, two team-mates can reach the pre-grasp configuration independently and the first waits for the

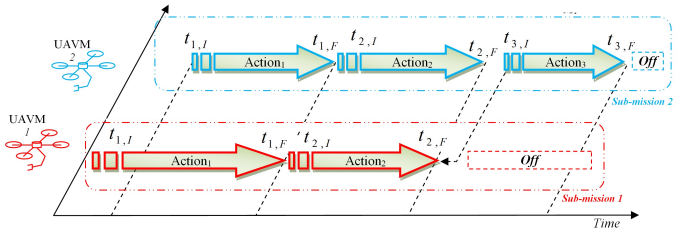


Figure 9. Mission to be executed by two UAVMSs.

second. However, once they have grasped the object to be transported, there is physical interaction between the cooperating UAVMSs, and thus the actions need to be synchronized at the higher possible frequency.

Figure 9 shows a mission that is composed by two sub-missions, each assigned to one UAVMS, where, the action 3 of the second UAVMS (with blue color) has time dependency to the second action of the first UAVMS.

3.5 Supervisor

The main role of the Supervisor is the execution of the mission generated by the Planner. During the mission, execution of the original plan may be changed, according to the actual state of the robotic team and the environment, by selecting the proper compound behavior to be performed.

Continuity of the control law must be preserved during the switching phase as done, e.g., in [31], [32].

In detail, the Supervisor switches among the compound behaviors according to the following criteria:

(i) the functionality behaviors are considered concluded if the error, $\widetilde{\sigma}(t)$, between the desired final configuration and the actual one, is below a suitable defined threshold $\widetilde{\sigma}_m$;

(ii) the proper safety behaviors are activated when the distance between the obstacle and the object or a single UAVMS, $d(t)$, is below a certain safety value d_m;

(iii) the mechanical joint limits of the manipulator are activated if (at least) one of the joints is close to violate its upper, $\bar{q}$, or lower, $\underline{q}$, limit. Namely, by defining a certain threshold, $\bar{\Delta}q$, the behavior is activated if $q(t) > \bar{Q}$ or $q(t) < \underline{Q}$, given that $\bar{Q} = \bar{q} - \Delta q$, $\underline{Q} = \underline{q} + \Delta q$ (refer to Appendix for a complete review);

(iv) to avoid deadlocks, each action has a certain maximum duration and it is deactivated after the final time, t_F, is achieved.

For example, if the Supervisor is running the compound behavior *VP*, i.e., vehicle position control, it will switch to *VOA+VP*, i.e., vehicle obstacle avoidance + vehicle position control, when an obstacle is detected.

Figure 10 shows the Supervisor flowchart. Each of the above listed criteria activates a flag indicating that it needs

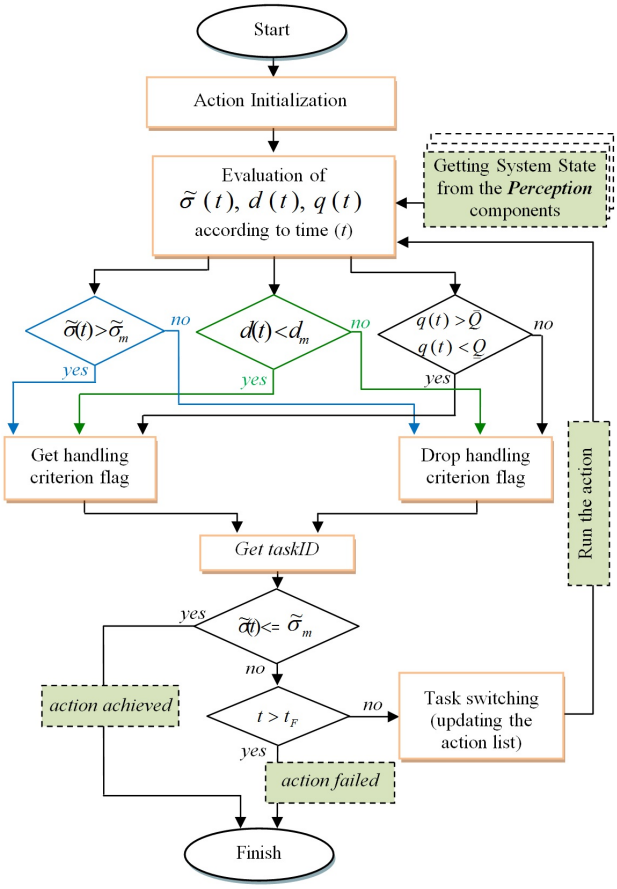


Figure 10. Flowchart implemented in the Supervisor for managing actions.

to be handled. A logical choice among safety, mechanical joint limits and task error influences selection of the compound behavior to be activated at the each sampling time.

4 Case Study

To test the proposed architecture, a case study has been developed by considering a system composed of two quadrotors, each equipped with a 5 DOFs robotic arm. The simulation model has been developed under Matlab/SimMechanics®; in detail, the vehicle parameters are those of the ASCTEC PELICAN quadrotor, while the arm parameters are those of the arm currently under development in the PRISMA laboratory of the University of Naples within the ARCAS project [9].

The objective is to perform a cooperative mission using two UAVMSs: they are required to approach an object, grasp it and move it along a desired trajectory. Each UAVMS has to perform one sub-mission, composed by a sequence of actions (see Fig. 11):

- Takeoff: Since the goal is to move the vehicle along the z axis, while keeping the arm still, only the compound behavior *VP* is activated.

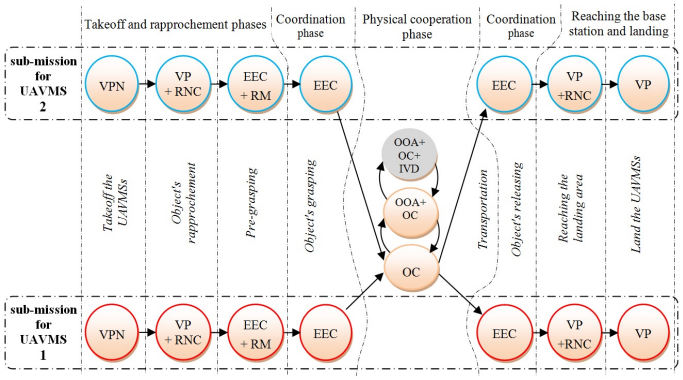


Figure 11. The sequence of actions to be managed by the Supervisor in the transportation scenario.

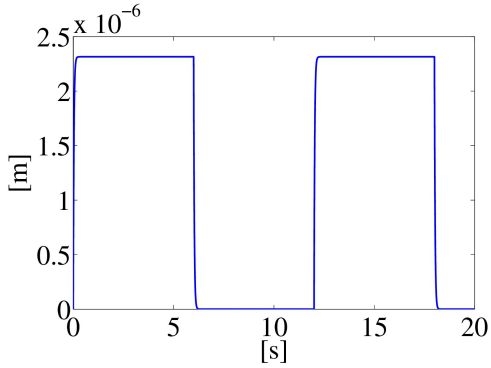
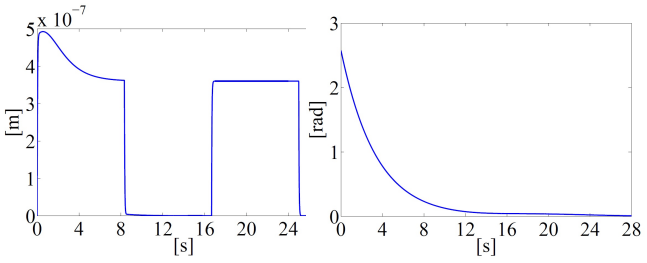


Figure 13. Norm of the vehicle position during the action "Takeoff".



(a) vehicle position error (b) joints position error

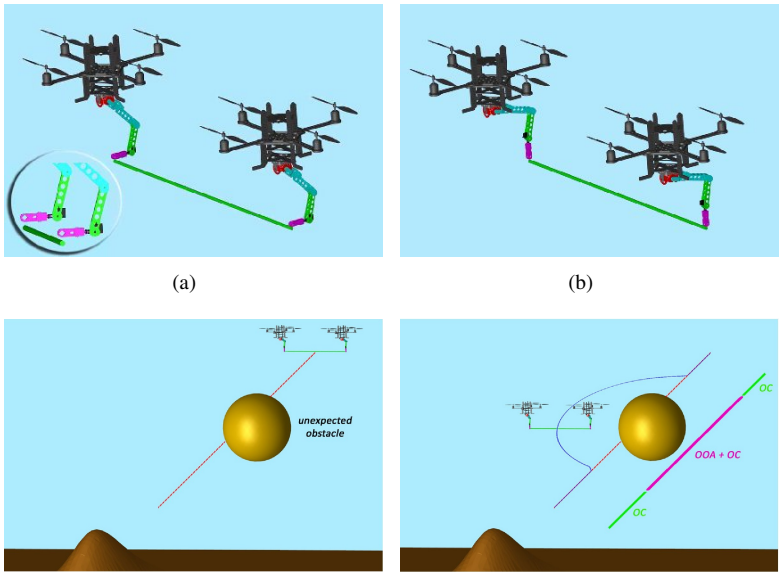
Figure 14. Task function errors during the action "Reach the object".

- Reach the object: Moves the vehicle near the object and, at the same time, set the arm in a particular configuration; this action includes the compound behavior *VP+RNC*, where, the vehicle position have the higher priority and the robot nominal configuration is the secondary behavior.
- Reach pre-grasping configuration: Moves the arm toward the best configuration for grasping (i.e., a configuration that maximizes the robot manipulability) and move the end-effector to the grasp position. Therefore, the compound behavior *EEC+RM* is activated, where, the *EEC* has the higher priority.
- Perform the grasp: Closes the grasp tool.
- Transports the object: Transports the object along a desired trajectory (a linear path). At the beginning the compound behavior *OC* is activated (i.e., the goal is to track the assigned trajectory for the object); then, during the motion execution, an obstacle obstructs the object's path. When the distance between the obstacle and the object is below a safety value ($d_m = 1.5$ m), the Supervisor switches from *OC* to *OOA+OC*, i.e., the object obstacle avoidance is executed with highest priority. Then, when the obstacle has been overcame, the Supervisor switches back to the compound behavior *OC*. During the transportation phase, in case the distance between the cooperating team-mates is below a certain threshold, the Supervisor activates the compound behavior *OOA+OC+IVD*. In the proposed simulation this doesn't happen since the object is relatively long.
- Release the object: Opens the grasping tool to release the object.
- Reach the station: Drives each UAVMS toward the base station, the implemented compound behavior is *VP+RNC*.
- Landing: For landing the UAVMS, the compound behavior *VP* is used.

Figure 12 shows some snapshots of the mission. In detail, Fig. 12(a) shows the approach to the object (the zoom on the bottom left corner shows details of the end-effectors and the object), Fig. 12(b) shows the arm reconfiguration, Fig. 12(c) shows the object in motion along the planned trajectory, while Fig. 12(d) shows the system during obstacle avoidance phase.

Figures 13 to 16 show the task function errors. For the sake of brevity only figures that refer to the first UAVMS are reported. Figure 13 reports the time history of the norm of the vehicle position error during the takeoff phase. Figure 14 shows the task function errors during the action "Reach the object"; in detail, in Fig. 14(a) the error related to the primary behavior (i.e., the norm of the vehicle position error) is reported, while in Fig. 14(b) the norm of the error between the desired arm joint positions and the actual ones is reported. It can be noticed that both task function errors converge to zero; therefore, both compound behaviors are correctly executed. Figure 15 shows the performance of the behavior executed during the pre-grasping; in detail, in Fig. 15(a) the norm of the position and orientation error of the end-effector are reported, while in Fig. 15(b) the manipulability index of the robotic arm, normalized to its maximum allowed value, w_{max}, is reported. The manipulability measure has been computed as in [33].

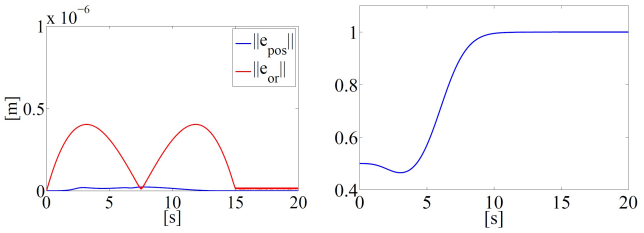
Figure 16 shows the performance obtained during the transportation of the shared object; the trajectory is reported in Fig. 16(a): it can be noticed that, during the obstacle avoidance



(a) (b)

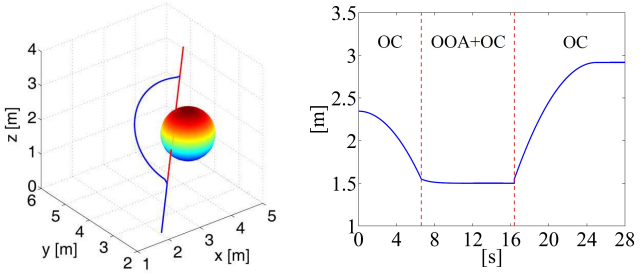
(c) (d)

Figure 12. Some snapshots of the mission of transporting a bar using two UAVMSs.



(a) end-effector position and orientation error norm (b) normalized manipulability index

Figure 15. Performance during the action "Reach pre-grasping configuration".



(a) planned (red) and actual (blue) object frame trajectory (b) distance between the vehicle center of mass and the obstacle

Figure 16. Object transportation performance.

phase, the object goes away from the planned trajectory (in red) which is reached back after the obstacle has been overcame; moreover, the safety distance of 1.5 m is kept, as shown in Fig. 16(b).

A video showing a mission of Coordinated transportation of a bar using two UAVMSs can be downloaded from this link http://webuser.unicas.it/lai/robotica/video.html#UAVMS, where also some preliminary experimental results are provided. From the aerial station that is located close to the working area, two UAVMSs take-off, reach a location close to the bar (object to be transported), grasp and move the bar toward a working area (building under construction). During the transportation phase, the programmed path goes through an obstacle (a building); the system is able to circumnavigate the building safety and attains the destination. Then, the UAVMSs release the bar and reach back the station.

5 Lessons Learned

From the software perspective, the challenge embedded in the experimental validation of a complex robotic mission is significant. Although this activity related to research is generally not new or original, it is time consuming and prone to *dramatic* errors.

Moreover, joint research projects usually involves heterogeneous working groups from different countries and different background (as example the ARCAS project [9] involves 8 Partners of 5 different countries). Thus, it is often necessary to *decompose* the overall problem in simpler concepts, easily manageable from the perspective of producing software in different laboratories by researchers with different backgrounds. In a sense, it is possible to describe the logical structure of the software with a bottom-up approach: we first defined the *elementary behaviors* and produced a library with well defined

input-output and the corresponding documentation. On the top of this we made the same with the *compound behaviors* where the atomic information of the elementary behaviors is not necessary anymore.

In a consequential way we agreed and implemented the concepts of *action*, *sub-mission* and *mission* with a progressive higher level of abstraction of the concepts. Thanks to this modularity each researcher, thus, *enters* the discussion at his level of pertinence by ignoring the lower-level concepts.

6 Conclusion

In this paper a new Control Software Architecture, CAVIS, has been presented, aimed at driving missions performed by cooperative Unmanned Aerial Vehicles Manipulator Systems (UAVMSs). The main objective of the architecture is to support a large range of possible cooperative scenarios using multiple UAVMSs. The architecture is designed around components that handle the current states of the involved UAVMSs, and provide basic functionalities. CAVIS implements the decomposition of the overall control problems in simpler sub-problems.

Future developments will be focused on the integration of CAVIS within a ROS environment as well as on extensive experimental testing.

Appendix

Kinematic control

Let us consider a system composed of N UAVMSs. The motion of each UAVMS can be described by the following variables:

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{x}_V \\ \boldsymbol{q} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_V \\ \boldsymbol{\phi}_V \\ \boldsymbol{q} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{x}_V \in \mathbb{R}^6$ represents the pose of the vehicle, given by the position $\boldsymbol{p}_V$ and the orientation (usually expressed via a roll-pitch-yaw triple of angles) $\boldsymbol{\phi}_V = [\phi_V \; \theta_V \; \psi_V]$ of the vehicle, and $\boldsymbol{q} \in \mathbb{R}^{n_M}$ represents the manipulator joint position vector, being n_M the number of DOFs of the arm.

The kinematic control problem for such a robotic system is to find a reference value, $\boldsymbol{\zeta}_r$, to be fed to the vehicle and arm motion controllers starting from the desired trajectory for the assigned tasks. Therefore, a kinematic control, usually, is based on two stages: first, an inverse kinematics algorithm computes the motion references for the vehicle and joint variables, then a motion control algorithm is in charge of tracking the motion references (see figure 17).

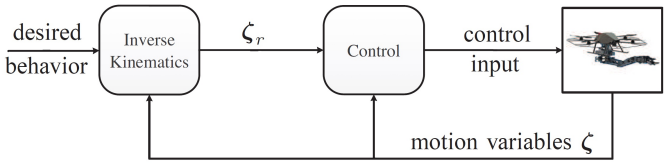


Figure 17. Scheme of the kinematic control.

Elementary behaviors

A specific elementary behavior assigned to the UAVMS can be analytically described through a task variable $\boldsymbol{\sigma} \in \mathbb{R}^m$ to be controlled. Let $\boldsymbol{f}$ be the configuration-dependent task function, representing the relationship between the task variable and the state vector $\boldsymbol{\zeta}$, defined in (1), i.e.,

$$\boldsymbol{\sigma} = \boldsymbol{f}(\boldsymbol{\zeta}). \tag{2}$$

The task Jacobian matrix $\boldsymbol{J}_\sigma \in \mathbb{R}^{m \times (6+n_M)}$ can be defined via the differential relationship

$$\dot{\boldsymbol{\sigma}} = \frac{\partial \boldsymbol{f}(\boldsymbol{\zeta})}{\partial \boldsymbol{\zeta}} \dot{\boldsymbol{\zeta}} = \boldsymbol{J}_\sigma(\boldsymbol{\zeta}) \dot{\boldsymbol{\zeta}}. \tag{3}$$

Let $\boldsymbol{\sigma}_d$ is the desired value of the task variable. In order to compute the reference for the motion controllers from (3), it is possible to derive a closed-loop inverse kinematics algorithm [33] as

$$\dot{\boldsymbol{\zeta}}_r = \boldsymbol{J}_\sigma^\dagger (\dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda} \tilde{\boldsymbol{\sigma}}), \tag{4}$$

where $\boldsymbol{J}_\sigma^\dagger = \boldsymbol{J}_\sigma^{\mathrm{T}} \left(\boldsymbol{J}_\sigma \boldsymbol{J}_\sigma^{\mathrm{T}} \right)^{-1}$ is a right pseudo-inverse of $\boldsymbol{J}_\sigma$, $\boldsymbol{\Lambda}$ is a suitable constant positive-definite matrix of gains and $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}$ is the task error.

Remark 1: Some DOFs of the system could be not actuated, as in the case of pitch and roll angles for standard quadrotor-arm systems. Hence, in order to extend the use of the inverse kinematics algorithm to this case, the (4) is solved only with respect to the actuated velocities and only the part of the Jacobian matrix referred to these velocities is considered. Further details can be found in [22].

In the following the task functions and the relative Jacobian matrices of the elementary behaviors reported in Section 3.1 are presented.

- Vehicle Position (VP):

  $$\boldsymbol{\sigma}_{VP} = \boldsymbol{p}_V \in \mathbb{R}^3, \quad \boldsymbol{J}_{VP} = [\boldsymbol{I}_3 \; \boldsymbol{O}_{3\times 3+n_M}] \in \mathbb{R}^{3\times 6+n_M},$$

  where $\boldsymbol{I}_\alpha$ and $\boldsymbol{O}_{\alpha\times\beta}$ are the $(\alpha \times \alpha)$ identity matrix and the null matrix of dimension $(\alpha \times \beta)$, respectively.
- Vehicle Yaw (VY):

  $$\boldsymbol{\sigma}_{VY} = \psi_V \in \mathbb{R}, \quad \boldsymbol{J}_{VY} = [\boldsymbol{0}_5 \; 1 \; \boldsymbol{0}_{n_M}] \in \mathbb{R}^{1\times 6+n_M},$$

  where $\boldsymbol{0}_\alpha$ is the null vector of dimension α.
- Vehicle Obstacle Avoidance (VOA): Let $\boldsymbol{p}_{ob} \in \mathbb{R}^3$ denote the obstacle position, the task function and the corresponding Jacobian matrix can be defined as

  $$\sigma_{VOA} = \frac{1}{2} \|\boldsymbol{p}_V - \boldsymbol{p}_{ob}\|^2 \in \mathbb{R},$$

  $$\boldsymbol{J}_{VOA} = (\boldsymbol{p}_V - \boldsymbol{p}_{ob})^{\mathrm{T}} [\boldsymbol{I}_3 \; \boldsymbol{O}_{3\times 3+n_M}] \in \mathbb{R}^{3\times 6+n_M}.$$

- Mechanical Joint Limit (MJL): A possible choice of the task function could be

$$\boldsymbol{\sigma}_{MJL} = \sum_{i=1}^{n_M} l_i(q_i)$$

where

$$l_i(q_i) = \begin{cases} \frac{(\underline{q}_i - q_i)^2}{2n_M}, & \text{if } q_i \leq \underline{q}_i, \\ 0, & \text{if } \underline{q}_i < q_i \leq \overline{q}_i, \\ \frac{(\overline{q}_i - q_i)^2}{2n_M}, & \text{if } q_i > \overline{q}_j, \end{cases}$$

with $\overline{q}_j$ and $\underline{q}_j$ are the lower and upper joint limit respectively. The task Jacobian is $\boldsymbol{J}_{MJL} = [\boldsymbol{0}_{1\times 6}\ \boldsymbol{J}_l] \in \mathbb{R}^{1\times 6+n_M}$ where

$$\boldsymbol{J}_l = \left[\frac{\partial l_1}{\partial q_1}, \frac{\partial l_2}{\partial q_2}, \ldots, \frac{\partial l_{n_M}}{\partial q_{n_M}}\right] \in \mathbb{R}^{n_M}.$$

- Robot Manipulability (RM): To define the task function, the manipulability measure for robot manipulators defined as $w(\boldsymbol{q}) = \sqrt{\det(\boldsymbol{J}_{E,V}^{\mathrm{T}} \boldsymbol{J}_{E,V})}$ could be adopted, where $\boldsymbol{J}_{E,V}$ is the Jacobian of the arm with respect to the vehicle center of mass. The task is fully characterized by the task function and the Jacobian

$$\sigma_{RM} = w(\boldsymbol{q}) \in \mathbb{R},$$

$$\boldsymbol{J}_{RM} = [\boldsymbol{O}_{1\times 6}\ \boldsymbol{J}_w] \in \mathbb{R}^{1\times 6+n_M},\ \boldsymbol{J}_w = \frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{1\times n_M}.$$

- Robot Nominal Configuration (RNC): This behavior can be described by the task function

$$\boldsymbol{\sigma}_{RNC} = \boldsymbol{q} \in \mathbb{R}^{n_M}$$

with the Jacobian

$$\boldsymbol{J}_{RNC} = [\boldsymbol{O}_{n_M\times 6}\ \boldsymbol{I}_{n_M}] \in \mathbb{R}^{n\times 6+n_M}.$$

- End-Effector Position (EEP): The task function is

$$\boldsymbol{\sigma}_{EEP} = \boldsymbol{p}_E \in \mathbb{R}^3,$$

where $\boldsymbol{p}_E$ is the position vector of the manipulator end-effector. The corresponding Jacobian $\boldsymbol{J}_{EEP} \in \mathbb{R}^{3\times 6+n_M}$ is the matrix such that $\dot{\boldsymbol{p}}_E = \boldsymbol{J}_{EEP}\dot{\boldsymbol{\zeta}}$, whose expression can be found in [22].
- End-Effector Orientation (EEO): The task function is

$$\boldsymbol{\sigma}_{EEO} = \boldsymbol{\phi}_E \in \mathbb{R}^3,$$

where $\boldsymbol{\phi}_E$ is the orientation vector (expressed in roll-pitch-yaw angles) of the manipulator end-effector. The corresponding Jacobian $\boldsymbol{J}_{EEO} \in \mathbb{R}^{3\times 6+n_M}$ is the matrix such that $\dot{\boldsymbol{\phi}}_E = \boldsymbol{J}_{EEO}\dot{\boldsymbol{\zeta}}$, whose expression can be found in [22].
- End-Effector Configuration (EEC): The task function is given by

$$\boldsymbol{\sigma}_{EEO} = \boldsymbol{x}_E = \begin{bmatrix} \boldsymbol{p}_E \\ \boldsymbol{\phi}_E \end{bmatrix} \in \mathbb{R}^6,$$

with the Jacobian

$$\boldsymbol{J}_{EEC} = \begin{bmatrix} \boldsymbol{J}_{EEP} \\ \boldsymbol{J}_{EEO} \end{bmatrix} \in \mathbb{R}^{6\times 6+n_M}.$$

- Inter-Vehicle Distance (IVD): This behavior is characterized by the task function

$$\sigma_{IV_{ij}} = \frac{1}{2}\|\boldsymbol{p}_{V_i} - \boldsymbol{p}_{V_j}\|^2 \in \mathbb{R},$$

$$\forall\,(i,j) \in \{1,\ldots,N\}\times\{1,\ldots,N\},\ i \neq j$$

and Jacobian

$$\boldsymbol{J}_{IV} = \begin{bmatrix} \boldsymbol{J}_{x_i}\ \boldsymbol{J}_{x_j} \end{bmatrix}$$

where

$$\boldsymbol{J}_{x_i} = -\boldsymbol{J}_{x_j} = (\boldsymbol{p}_{V_i} - \boldsymbol{p}_{V_j})^{\mathrm{T}}\,[\boldsymbol{I}_3\ \boldsymbol{O}_{3\times 3+n_M}] \in \mathbb{R}^{3\times 6+n_M}.$$

- Object Configuration (OC):
This elementary behavior requires from the planner module a desired position ($\boldsymbol{p}_{o,d}$) and a desired orientation ($\boldsymbol{R}_{o,d}$) for a coordinate frame attached to the object ($\mathcal{F}_o$). In addition to the object motion, the elementary behavior requires to define also the grasp geometry, namely the desired relative position ($\boldsymbol{p}^o_{Ei,o}$) and orientation ($\boldsymbol{R}^o_{Ei,o}$) of each robot end-effector with respect to the object frame. Notice that the superscript o denotes that the quantities are referred to the frame $\mathcal{F}_o$.
On the basis of the desired object motion and the desired grasp geometry, it is possible to compute the desired trajectories, in terms of position and orientation, for the robot end-effectors, via

$$\begin{cases} \boldsymbol{p}_{Ei,d} = \boldsymbol{p}_{o,d} + \boldsymbol{R}_{o,d}\boldsymbol{p}^o_{Ei,o} \\ \boldsymbol{R}_{Ei,d} = \boldsymbol{R}_{o,d}\boldsymbol{R}^o_{Ei}\,. \end{cases} \tag{5}$$

The desired linear and angular velocities for each robot can be obtained by deriving the (5), as

$$\begin{cases} \dot{\boldsymbol{p}}_{Ei,d} = \dot{\boldsymbol{p}}_{o,d} - \boldsymbol{S}(\boldsymbol{R}_{o,d}\boldsymbol{p}^o_{Ei,o})\boldsymbol{\omega}_{o,d} + \boldsymbol{R}_{o,d}\dot{\boldsymbol{p}}^o_{Ei,o} \\ \boldsymbol{\omega}_{Ei,d} = \boldsymbol{\omega}_{o,d} + \boldsymbol{R}_{o,d}\boldsymbol{\omega}^o_{Ei,o}, \end{cases} \tag{6}$$

where $\boldsymbol{S}(\cdot)$ is the (3×3) skew-symmetric matrix operator performing the cross product [33]. If the robots grasp a rigid object in a rigid way, the relative variables, $\boldsymbol{p}^o_{Ei,o}$ and $\boldsymbol{R}^o_{Ei,o}$, are to be kept constant, therefore the (6) becomes

$$\begin{cases} \dot{\boldsymbol{p}}_{Ei,d} = \dot{\boldsymbol{p}}_{o,d} - \boldsymbol{S}(\boldsymbol{R}_{o,d}\boldsymbol{p}^o_{Ei,o})\boldsymbol{\omega}_{o,d} \\ \boldsymbol{\omega}_{Ei,d} = \boldsymbol{\omega}_{o,d}\,. \end{cases} \tag{7}$$

Thus, the OC elementary behavior can be viewed as the elementary behavior EEC applied to the transporting robots with desired pose and generalized velocity given by (5) and (7), respectively.
- Object Obstacle Avoidance (OOA): The task function of this behavior is defined as

$$\sigma_{OOA} = \frac{1}{2}\|\boldsymbol{p}_o - \boldsymbol{p}_{ob}\|^2 \in \mathbb{R},$$

where $\boldsymbol{p}_o$ is the actual position of the origin of $\mathcal{F}_o$ and $\boldsymbol{p}_{ob}$ is the (constant) location of an obstacle. The task Jacobian can be computed as

$$\boldsymbol{J}_{OOA} = (\boldsymbol{p}_o - \boldsymbol{p}_{ob})^{\mathrm{T}} \in \mathbb{R}^{1\times 3}.$$

Compound behaviors

By adopting the Null Space-based Behavioral control approach, the overall system velocity is obtained by properly merging the velocity vectors computed for each behavior as if it was acting alone; then, before adding the task contribution to the overall velocity command, a lower-priority behavior is projected onto the null space of the higher-priority behaviors. Hence, the overall system velocity can be computed according to the following

$$\dot{\boldsymbol{\zeta}}_r = \dot{\boldsymbol{\zeta}}_1 + \sum_{k=2}^{N_t} \boldsymbol{N}_{1,k-1}\dot{\boldsymbol{\zeta}}_k, \tag{8}$$

$$\boldsymbol{N}_{1,k} = \left(\boldsymbol{I} - \boldsymbol{J}_{1,k}^{\dagger}\boldsymbol{J}_{1,k}\right), \tag{9}$$

where the subscript k denotes the task priority, N_t is the number of behaviors to be fulfilled, $\boldsymbol{N}_{1,k}$ is a projector onto the null space of the augmented Jacobian $\boldsymbol{J}_{1,k}$ defined as

$$J_{1,k} = [J_1 J_2 \cdots J_k]^T \tag{10}$$

7 Conclusion

In this paper a new Control Software Architecture, CAVIS, has been presented, aimed at driving missions performed by cooperative Unmanned Aerial Vehicles Manipulator Systems (UAVMSs). The main objective of the architecture is to support a large range of possible cooperative scenarios using multiple UAVMSs. The architecture is designed around components that handle the current states of the involved UAVMSs, and provide basic functionalities. CAVIS implements the decomposition of the overall control problems in simpler sub-problems.

Future developments will be focused on the integration of CAVIS within a ROS environment as well as on extensive experimental testing.

References

[1] S. Nestinger and H. Cheng, "Mobile-R: A reconfigurable cooperative control platform for rapid deployment of multi-robot systems," in *Proc. IEEE International Conference onRobotics and Automation (ICRA)*, 2011, pp. 52–57. 1

[2] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The NSB control: a behavior-based approach for multi-robot systems," *Paladyn Journal of Behavioral Robotics*, vol. 1, no. 1, pp. 48–56, 2010. [Online]. Available: http://webuser.unicas.it/arrichiello/papers/AntArrChi_paladyn2010.pdf 1, 1, 3.2

[3] K. Baizid, C. Ryad, and H. Traveler, "Virat: An advanced multi-robots platform," in *Proc. Industrial Conference on Electronics and Applications (ICIEA)*, 2011, pp. 564–569. 1

[4] Y.-R. Tang and Y. Li, "The software architecture of a reconfigurable real-time onboard control system for a small UAV helicopter," in *Proc. 8th International Conference on Ubiquitous Robots and Ambient Intelligence URAI*, 2011, pp. 228–233. 1

[5] J. Gancet, G. Hattenberger, R. Alami, and S. Lacroix, "Task planning and control for a multi-UAV system: architecture and algorithms," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, 2005, pp. 1017–1022. 1, 1

[6] A. Ortiz, F. Bonnin-Pascual, E. Garcia-Fidalgo, and J. Beltran, "A control software architecture for autonomous unmanned vehicles inspired in generic components," in *Proc. 19th Mediterranean Conference on Control Automation*, 2011, pp. 1217–1222. 1, 1

[7] M. Manyoky, P. Theiler, D. Steudler, and H. Eisenbeiss, "Unmanned aerial vehicle in cadastral applications," in *Conference on Unmanned Aerial Vehicle in Geomatics*, 2011, pp. 1–6. [Online]. Available: http://www.geometh.ethz.ch/uav_g/proceedings/manyoky 1

[8] V. Lippiello and F. Ruggiero, "Exploiting redundancy in cartesian impedance control of UAVs equipped with a robotic arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2012, pp. 3768–377. 1

[9] "ARCAS - Aerial Robotics Cooperative Assembly System," 20/10/2013. [Online]. Available: http://www.arcas-project.eu 1, 4, 5

[10] J. C. Baillie, "Towards a universal robotic body interface," in *Proc. EEE/RAS International Conference on Humanoid Robots*, vol. 1, no. 1, November 2004, pp. 33–51. 1

[11] "Microsoft robotics developer studio," 09/11/2013. [Online]. Available: http://www.microsoft.com/robotics/ 1

[12] A. H. Brian P. Gerkey, Richard T. Vaughan, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. International Conference on Robotics and Automation*, 2003. 1

[13] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009. 1

[14] H. Bruyninckx, in *Open robot control software: the OROCOS project*, vol. 3, 2001, pp. 2523–2528. 1

[15] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *3rd Int. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, vol. 7628, 2012. 1

[16] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, Oct 2012, pp. 2815–2821. 1

[17] "Ardroneapiref," 20/05/2014. [Online]. Available: https://projects.ardrone.org/boards/1/topics/show/5203 1

[18] V. Grabe, M. Riedel, H. Bulthoff, P. Giordano, and A. Franchi, "The telekyb framework for a modular and extendible ros-based quadrotor control," in *Mobile Robots (ECMR), 2013 European Conference on*, Sept 2013, pp. 19–25. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6698814 1

[19] S. K. J. Fink, N. Michael and V. Kumar, "Planning and control for cooperative manipulation and transportation with aerial robots," *International Journal of Robotics Research*, vol. 30, no. 3, 2010. 1

[20] "Robot standards and reference architecture," 09/11/2013. [Online]. Available: http://wiki.robot-standards.org/index.php/Main_Page 1

[21] A. Oreback and H. I. Christensen, "Evaluation of architectures for mobile robotics," *Autonomous Robots*, vol. 14, no. 1, pp. 33–49, January 2003. [Online]. Available: http://link.springer.com/article/10.1023/A:1020975419546 1

[22] G. Arleo, F. Caccavale, G. Muscio, and F. Pierri, "Control of quadrotor aerial vehicles equipped with a robotic arm," in *Proc. 21st Mediterranean Conference on Control Automation (MED)*, 2013, pp. 1174–1180. 1, 1, 6

[23] F. Keith, N. Mansard, S. Miossec, and A. Kheddar, "Optimization of tasks warping and scheduling for smooth sequencing of robotic actions," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 1609–1614. 1

[24] I. Nesnas, A. Wright, M. Bajracharya, R. Simmons, and T. Estlin, "CLARAty and challenges of developing interoperable robotic software," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2003, pp. 2428–2435 vol.3. 1

[25] T. Kaupp, A. Brooks, B. Upcroft, and A. Makarenko, "Building a software architecture for a human-robot team using the orca framework," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 3736–3741. 1

[26] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Experiments of formation control with multirobot systems using the null-space-based behavioral control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1173–1182, September 2009. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4814533 1, 3.2

[27] "Centro Aanzado de Tecnologias Aeroespaciales," 07/05/2014. [Online]. Available: http://www.catec.com.es/ 1

[28] R. Grepl, "Adaption of mathworks real-time workshop for an unsupported embedded platform," in *Mechatronics (ICM), 2011 IEEE International Conference on*, April 2011, pp. 881 – 886. 1

[29] O. Netland and A. Skavhaug, "Adaption of mathworks real-time workshop for an unsupported embedded platform," in *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference on*, Sept 2010, pp. 425–430. 1

[30] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic system," *IEEE Transactions on Robotics*, vol. 25, pp. 985–994, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04814683 3.2

[31] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Transactions on Robotics and Automation*, vol. 23, no. 1, pp. 60–72, 2007. 3.5

[32] J. Lee, N. Mansard, and J. Park, "Intermediate desired value approach for task transition of robots in kinematic control," *Robotics, IEEE Transactions on*, vol. 28, no. 6, pp. 1260–1277, Dec. 2012. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6334480 3.5

[33] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Verlag, 2009. 4, 6, 6

Gianluca Antonelli is an Associate Professor at the "University of Cassino and Southern Lazio". His research interests include marine and industrial robotics, multi-agent systems, identification. He has published 34 international journal papers and more than 90 conference papers, he is author of the book "Underwater Robots" (Springer-Verlag, 2003, 2006, 2014) and co-authored the chapter "Underwater Robotics" for the Springer Handbook of Robotics, (Springer-Verlag, 2008, 2015). He has been scientific responsible of the STREP Co3AUVs, Associate Editor for the IP project ECHORD and researcher for the IPs ARCAS and EUROC. He served both as independent expert and reviewer for the European FP calls several times since 2006. He is chair of the IEEE RAS Chapter of the IEEE-Italy section, he has been Chair of the IEEE Robotics and Automation Society (RAS) Technical Committee in Marine Robotics. He served in the Editorial Board of the IEEE Transactions on Robotics, IEEE Transactions on Control Systems Technology, Springer Journal of Intelligent Service Robotics, he is Editor for the RAS Conference Editorial Board.

Khelifa Baizid received his B. Sc. and Engineering degrees in mechanical engineering from the University of M'hamed Bougara of Boumerds (Algeria) in 2001 and 2004 respectively. In 2007, he received his Magister degree from the Polytechnic Military School, Algiers. He prepared his PhDs degree (2010) in Robotics at the Italian Institute of Technology with collaboration with the University of Genova, Italy. He was a postdoctoral researcher at Italian Institute of Technology in 2011, and at Brno University of Technology in 2012, Czech Rep. Currently, he is a research assistant in the DIEI at University of Cassino and Southern Lazio.

Fabrizio Caccavale received the *Laurea* degree and the Ph.D. degree in Electronic Engineering from the University of Naples in 1993 and 1997, respectively. From 1999 to 2001 he has been Assistant Professor at the Department of Computer and Systems Engineering of the University of Naples. He is currently Associate Professor at the School of Engineering of the University of Basilicata. His research interests include manipulator inverse kinematics techniques, cooperative robot manipulation, fault diagnosis and nonlinear control of mechanical systems. He has published more than 100 journal and conference papers. He is co-author of the book "Control and Monitoring of Chemical Batch Reactors" (Springer, 2011), co-editor of the book "Fault Diagnosis for Mechatronic Systems: Recent Advances" (Springer, 2002) and co-author of the chapter "Cooperative Manipulators" for the Springer Handbook of Robotics, (Springer, 2008). He has been in the program committee of several international conferences and workshops. He has served as Associate Editor of the journals *Robotica* and *IEEE Transactions on Control Systems Technology*. He is *Senior Member* of IEEE.

Gerardo Giglio received the Bachelor degree in 2007 and the Master degree cum laude in 2011, both in Mechanical Engineering, from the University of Basilicata. From January to December 2013 he was a contract researcher at the School of Engineering of the University of Basilicata. Since January 2014 he is a Ph.D. student in Innovation Engineering and Sustainable Development from the School of Engineering of the University of Basilicata. His main research interests are focused on dynamic simulation of robotic systems and aerial robotics.

Giuseppe Muscio received the Bachelor degree cum laude in 2006 and the Master degree cum laude in 2010, both in Mechanical Engineering. In February 2014 received the Ph.D. degree in Industrial and Innovation Engineering from the School of Engineering of University of Basilicata, with a thesis, titled "Modeling and control of multi-arm systems equipped with robotic hands", funded by the Italian Space Agency (ASI). From October 2012 through April 2013 he had been visiting scholar at Computer Science Departement of the Rensselaer Polytechnic Institute, Troy, New York. Since November 2013 he is with the Revoind Industriale. His research activity is focused on control of dual arm/hand dexterous manipulators.

Francesco Pierri received the *Laurea* Degree cum laude in Mechanical Engineering and the Ph.D. degree in Environmental Engineering from the University of Basilicata in 2003 and 2007, respectively. From March to September 2006 he has been a visiting scholar at the Department of Automatic Control of the Lund University, Sweden. Since December 2008 he is Assistant Professor at the School of Engineering of the University of Basilicata. His research interests include process control, fault diagnosis and fault tolerant control for nonlinear systems, and robotics. He is co-author of more than 30 journal and conference papers and co-author of the book "Control and Monitoring of Chemical Batch Reactors" (Springer, 2011). He is associate editor of the International Journal of Robotics and Automation. He is *Member* of IEEE since 2004.