Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 147 – 163

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

On Autonomous Robotic Cooperation Capabilities Within Factory and Logistic Scenarios

G. Casalino*, E. Simetti, F. Wanderlingh

K. Darvish, B. Bruno, F. Mastrogiovanni

DIBRIS-University of Genoa, via Opera Pia 13, 16145, Genoa, Italy

Abstract

The paper presents the development of a unified functional, algorithmic and Software (Sw) architecture, which can be adopted as a standard for controlling, at action level only, *any* robotic structure within a given wide class of them; even of reconfigurable type within the class. Such control architecture is therefore deemed very suitable for operating within factory and/or logistic, possibly reconfigurable, scenarios. Moreover, for the few cases of cooperative activities to be established between agents not allowed to be cable connected, an effective coordination policy, based on the exchange of a reduced information set, only regarding the cooperation goals, is developed; and relevant simulative and experimental trials are briefly outlined. Moreover, the advantage of having, in whatever operative condition, the possibility of commanding the involved structures only in terms of the ultimate goals of each action, also seems to be the right basis for having non-negligible improvements within their integration with automated action planning, and even learning, techniques.

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: Cooperarating robots; Reconfigurable robotic systerm; Cooperative manipulation and transportation; bi-manual manipulation;Vehicle manipulator systems; Action based control architectures.

1. Introduction

* Corresponding author.
E-mail address: pino@dist.unige.it

Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.219

The employment of cooperating robots (or collaborative or coordinating) for executing complex activities within factory and/or logistic scenarios is currently foreseen as a valid perspective toward non-negligible improvements of the operational activities in such fields.

This may imply the coordination between two arms within a dual-arm work-cell, or the coordination of a mobile platform with supported arms, within vehicle-manipulator systems used for de-localized interventions; up to the coordination between different vehicle-manipulator systems employed for transporting large-space and/or large-weight items.

Basic robotic structures, or basic units (i.e. vehicles and arms), could in fact be employed either as individual agents or as coordinately operating parts within more complex organizations of them, characterized by enhanced overall operational capabilities. In this perspective this should also lead to higher operational versatility (e.g. expandability/scalability via reconfiguration); as well as to reduced sets of basic robot typologies to be adopted; and also to improved reliability aspects; other than to reduced purchase/maintenance costs per units.

To the aim of fully exploiting robotic cooperation capabilities, each basic unit (arm or vehicle) should be supported by a unified functional and algorithmic control architecture, to be organized in such a way to satisfy the following requirements: a) automatically guaranteeing all safety and operational-enabling unit-specific conditions; b) contemporarily best-accomplishing to externally provided mission commands; c) operating without structural changes also when assembled with other ones; d) in this case becoming a synergic part of a resulting enlarged control architecture; that however should result it also capable of governing the resulting system within the same requirements expressed at points a) to c) for individual units. Moreover, other than allowing for almost seamlessy transitions from cooperative to non-cooperative situations, such unified control architecture should also allow, in whatever situation (single units or composed ones), for receiving high-level commands only (for instance only relevant to the object-frame of a cooperatively transported item); because the structural motions that will allow for best-obeying to them, while also satisfying to all the operational and safety conditions of each composing unit, and also between the units (for instance avoiding their collisions) should be reactively generated by the overall resulting merged control framework. Such aspect should in turn greatly facilitate the integration with methodologiea for automatic reasoning and/or learning-based task planning, specifically devoted to robotic systems, whose application at task-space level only should in fact allow for non-negligible improvements in both their implementation and operability.

In the sequel, the paper will focus on a recently improved (by part of some of the authors) control methodology and on the related organizational and implementation aspects; which have been purposely developed by the authors research for allowing the the achievement of all the above-mentioned requirements. To this aim, the paper is therefore organized as follows: Section two will provide some useful conceptual definitions and related preliminary considerations; Section 3 will show how the unified control structure can be developed for any member of a given class of robots (single units or composed ones resulting from reconfigurations); while section 4 will show how the same control architecture can be extended to the case of cooperating separate units. Finally, Section 5 will outline some simulations and experimental trials relevant to cooperating robotic systems, to the aim of confirming and supporting the applicability of the proposed unified control architecture. Some conclusions, reported in Section 5, will finally complete the work.

2. Centralized versus decentralized control architectures

A somewhat generic factory or logistic scenario, employing different mechanically organized *robotic structures*, ranging from the simplest ones, like vehicles and fixed-base single-manipulators, till the most complex ones, as for instance vehicle/dual-arm systems, will be assumed; where clusters of them can also be employed for executing coordinated activities beyond those allowed to individual robotic structures. To this regard, the following conceptual definitions seem therefore worth to be introduced.

An *Individual Robotic Agent* (for short an *Agent*) is intended to be any robotic structure having its coordinated motion activities governed by an associated, suitably designed, *Centralized Agent-control System* (for short *Agent-Controller*). The design and resulting performances exhibited by an Agent-controller are assumed achievable via the exploitation, without any restriction, of: a) *all* information relevant to the whole geometric and kinematic model of

the underlying robotic structure; b) the real-time measurements of *any* needed variable from the inside of the robotic structure; c) the real-time acquisition of any additional environmental information provided by endowed sensors; as well as the operational commands to which the agent has to obey at best.

A *Multi Robotic Agent* (for short a *Multi-agent*) is instead intended as the aggregation, or collection, of *different* Agents, which results when more-then-one of these lasts are all together employed for executing, cooperative activities; via the aid of a suitably implemented *Coordination Policy*, (*Coordinator* for short) established among their *different* agent-controllers; in turn supported by a real-time information exchanges with them.

In the Multi-agent case, it generally makes sense assuming the information exchange traceable to a *proper subset* of the complete one that would correspond to a full-exchange of the individual information of each agent; thus avoid re-entering into the centralized case, where the coordinator could de-facto subsume the separate agents, by becoming itself the unique centralized Agent-controller of the resulting composed structure. Moreover, provided possible, such information exchange should remain confined within the set of variables strictly needed for representing the evolution of the required cooperative activity (i.e. by as much as possible excluding details regarding the very many internal variable supporting the mechanical motion of each involved agent). The difference between the two concepts can be easily appreciated by comparing the potential performances for instance achievable by a vehicle/dual arm system centralized governed as in fig. 1a, versus those instead achievable when the same robotic system is considered as a Multi-agent decentralized controlled as in Fig. 1b. In the second case in fact, just due to the assumed non-complete availability, by part of the coordinator, of the merged individual information sets, we have that its potentially expressible performances cannot ever be better than those relevant to its centralized counterpart.

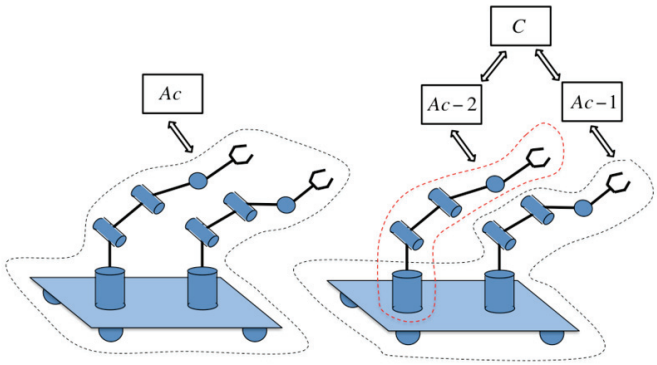

Fig. 1. Single-agent (a) and Multi-agent (b) control

The convenience in resorting to decentralized solutions as in Fig. 1b, might however be appreciated within more advanced scenarios requiring for *reconfigurable* robotic structures; where these last can transform their mechanical structure, while however remaining within an assumed *population* (or equivalently a *class*) of them. Even in these cases, however, resorting to coordination policies for managing in-the-class transformations can actually be avoided. Where the reason of this just relies on the fact that the control techniques to be proposed in the next section 3 (based on the so-called *task-priority Control* approach) for developing the Agent-controllers of each class member, actually lead to control architectures always exhibiting the *same* functional, algorithmic, and SW organization (i.e. unified in their structural elements); thus only requiring specific parameterizations in correspondence of each class member (of permanent type and/or resulting from mechanical reconfigurations within the class). Thus allowing controlling any resulting reconfiguration by simply selecting (and downloading) the corresponding Agent-controller (efficient and rapid ways for executing such context switches should be therefore devised, as a must for reconfigurable scenarios)

However, since the same arguments of Section 3 will also evidenced how the agent-controller of a given class-member can also directly manage any downscaled use of its underlying robotic structure (e.g. stopping one arm of a vehicle/dual-arm system, for having it behaving like a vehicle/single-arm; and so on); and since such downscaled

behaviours can be obtained by simply commanding the agent-controller to primarily obey to the constraints qualifying each downscaled use (e.g. in case of stopping one arm of a vehicle/two-arm system, by simply primarily imposing a specific rest position for the non-used arm); it immediately follows that, at least for the special scenario of reconfigurations achievable within classes constituted by *identical basic robotic structures* (i.e. identical vehicles and identical arms), the same agent-controller developed for the most complex robotic structure in the class (e.g. the vehicle/two arms one) could therefore serve also for any other class-member; including those resulting from reconfigurations within the class; thus without requiring any specific agent-controller selection (and relevant Sw downloading) in case of reconfigurations; while also achieving the non-negligible advantage of minimizing any context switching time. Nevertheless, since the exploitation of such possibility would however require a non-negligible standardization effort, mainly for mechanically designing homogeneous basic units, with related non-negligible economic investments; such possibility does not seem yet ready to be welcomed with sufficient belief.

Accordingly with the above considerations, what however remains worth to be developed, is the problem related with the definition of the coordination policies that instead, *unavoidably*, must be adopted for few remaining special cases of aggregations that cannot be included as members of the considered class. These are represented by all cases where a Multi-agent organization de-facto arises in consequence of cooperation activities to be established among robotic structures not allowed for cabled-data-exchange among them (as instead it has been implicitly assumed for any mechanical reconfigurations within the class). This is for instance the case of the "occasional" cooperation that could be required to a couple of separate vehicle/one-arm systems (for instance passing an object grasped by one of them, to the other; or manipulating a shared grasped object; or even possibly performing on-fly assembly tasks) where the reduced-bandwidth of a wireless data-exchange, certainly prevents the intensive rates that would instead be necessary for establishing a centralized Agent-controller also for such resulting "occasional" situation. As stated in the introduction, such case will be considered in in section 4 in some details, where an effective coordination policy will be proposed.

3. Centralized Agent-control Architectures

In this section, the assumed most complex case of a vehicle/dual-arm system within the class will be considered; for which the requirements of its agent-controller will be identified; to be then followed by its structuring into the announced *unified* functional and algorithmic control architecture, capable of eventually guaranteeing (i.e. for increasing time) the best fulfilment of *any* set of control objectives the agent could be commanded to achieve, in a *fully concurrent* way. Such unified structuring will be the result of the employment of the so-called task-priority based control technique [1,2], which has been recently extended by some of the authors [3,4,5] for also including the so-called inequality-control objectives, in addition to the traditional equality ones. And this within both unconstrained and constrained motion conditions. With these lasts (constrained motion conditions) they also characterized by interaction-control requirements (i.e. force/torque control objectives at the interaction zones) specifiable as equality or inequality-type objectives, to be eventually achieved.

In the following, due to space limitations, we will however not consider interaction control aspects. Nevertheless the development of Agent-controllers for vehicle/dual-arm systems, even if restricted to motion control aspects only, will however suffice for clarifying how their downgrading toward Agent-controllers for simpler structures within the class, can always be directly implemented, as it has been already stated. Finally it should be however kept into account that the control architecture that we are now going to discuss, actually constitutes the so-called *Kinematic Control Layer* (KCL) of an overall agent-control architecture. That is the control layer in charge of real-time providing, to the underlying *Dynamic Control Layer* (DCL), the most appropriate reference system-velocity signals to be best-tracked by the corresponding actuators of the underlying robotic structure; each one just controlled by the DCL itself. Within this work, however, we shall assume the DCL of any considered robotic structure, as given; and adequately performing with respect to the KCL-provided real-time requests.

3.1. The Real-time Agent Information System

In the following, the so-called *Agent-configuration* (or *System-configuration*, or also *configuration* for short) will be assumed represented by the vector c, constituted by the stacking of vectors q_1, q_2 of the current joint positions of

each arm; plus the vector p; in turn constituted by the stacking of sub-vectors p_1, p_2, respectively representing the absolute position and attitude of the vehicle-attached frame $< v >$ with respect to the world frame $< 0 >$. Then, quite similarly, the so-called *agent-velocity vector* (or *system-velocity vector*) will be represented by the overall vector $\dot{y}$ constituted by the stacking of vectors $\dot{q}_1, \dot{q}_2$ of the current joint rates of each arm; plus vector v, in turn constituted by the stacking of sub-vectors v_1, v_2, respectively representing the linear and angular velocities of the attached vehicle frame $< v >$, both represented with component on the vehicle frame $< v >$ itself. Such vectors are also assumed real-time acquired by the agent-controller; and used for real-time evaluating all the so-called relative, and then also absolute, *basic transformation matrices*; that is the transformation matrix of each link-frame of the arm with respect to that of its antecedent, including the end-effector frames, and then, on their basis, also their absolute ones (i.e. with respect to the world frame $< 0 >$). Then, from the basic transformation matrices, also all the so-called *basic Jacobian matrices* are they also real-time evaluated (each one representing the existing linear relationship between the system velocity vector, as input, and the absolute linear and angular velocity vectors of each link-frame, including the end-effectors frames; all intended with components on a common frame, as for instance the world one $< 0 >$. As it is well known, the above basic transformation and Jacobian matrices represent the fundamental support for possibly also real-time evaluating any further scalar variables and vectors definable within the system, and/or between the system ad the environment; together with their relevant Jacobian relationships with the system velocity vector $\dot{y}$; that might actually be real-time required by the agent, for accomplishing to its control functionalities. Such overall set of real-time evaluations is here denoted as the *real-time agent information system* (for short *agent information system*, or simply *information system*).

3.2. Objectives, Tasks, Activation functions

Let us start by formerly consider a generic, configuration dependent, *scalar variable* $x(c)$ among those definable within the system, or between the system and the environment, via the aid of the information system. For such scalar variable we shall say that it is required to achieve an *equality control objective* (for short *equality objective*) or an *inequality control objective* (for short *inequality objective*), when one of the following conditions is *desired* to be eventually achieved

$$t \to \infty \quad \Rightarrow \quad x(c) = x_0 \quad or \quad [\, x(c) \geq x_m \;\; and/or \;\; x(c) \leq x_M \,] \;\; ; \;\; x_m \leq x_M \tag{3.2.1}$$

Note that if m different variables $x_i(c)$ are considered, each one of them representing the *i*-th component of a certain vector h, then it becomes possible requiring the whole vector achieving, component-by-component, equality or inequality objectives (i.e. what will be here termed as a *multiple objective*, or *M-objective* for short); thus meaning that starting considering scalar objectives does not influence the generality of the approach. Furthermore, in case the scalar $x(c)$ is for instance the norm of a certain vector h, then it can be for instance used for requiring a particular value for it (e.g. zeroing it), or instead for requiring it to eventually stay below and/or above a given threshold. In particular, just in case the zeroing of a vector h is required, such requirement can be formalized in both terms of zeroing its norm, or either zeroing each one of its components; however with the difference that, while in the first case the employment of a sole Cartesian degree of freedom is required; in the second, m of them must be engaged. *Always* associated to each objective, the so-called *reference feedback-rate* $\dot{\bar{x}}$ (for short *feedback-task* or also *reactive-task*) will be considered, corresponding to what has to be real-time provided by a suitably defined *reference* or *reactive* feedback law that, in case it could be directly applied, would consequently asymptotically drive its associated variable $x(c)$ toward an arbitrary point x^* located inside the interval where the objective is satisfied (obviously $x^* = x_0$ in case of an equality objective); as for instance the following simple one

$$\dot{\bar{x}}(x) \doteq \gamma(x^* - x) \quad ; \quad \gamma > 0 \tag{3.2.2}$$

Where γ is a positive gain proportional to the desired convergence rate for the considered variable.

Moreover, *always* associated to an objective (and relevant feedback-task) there will also be the so-called *activation function*

$$a = a(x) \in [0,1] \tag{2.2.3}$$

Simply constituted by a continuous sigmoidal function of the objective variable $x(c)$, which assumes zero values within a complete inner sub-interval of the interval of validity of the associated objective; that is by the simple forms indicated in the following Fig. 2 (for equality and left-inequalities only).

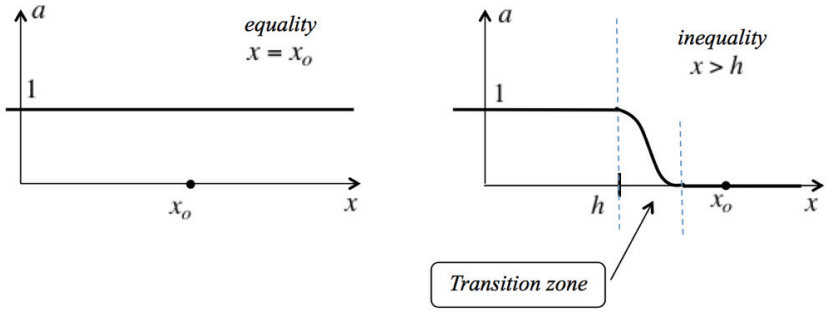

Fig. 2. General shape of the Activation functions

Such functions will be used in the sequel as a valid mean for automatically *smoothly* de-activate/activate the corresponding reactive-task, in dependency of the achievement or not of its related objective, at current time. Finally note that, since we are assuming that the existence of an objective *always* implies also the existence of its associated feedback-task and related activation function; throughout the sequel we will be therefore authorized to use the sole word "objective" as *always* being inclusive of all its three items.

3.3. The Task-priority Based Agent-controller Architecture

The set of objectives to be *concurrently* accounted for, can be formerly grouped into a number of clusters of them, each one termed as a *multiple objective* (for short *M-objective*); with each one of them in turn characterized by its own *priority-level*; where such terminology is used for denoting the fact that each control activity devoted to the *concurrent* achievement of *all* objectives within an M-objective located at a certain priority level, must however not interfere with those relevant to all the M-objectives located at higher priority levels.

Thus, accordingly with this, in correspondence of an M-objective located at the generic *k*-th priority level (with $k = 1,2,...,N$ intended in a descending priority ordering) the following organization into the *k*-th *task-vector* of the associated *multiple feedback-task*, or equivalently a *multiple reactive-task* (*M-feedback-task* or *M-reactive-task* for short) extractable from the considered *k*-th M-objective, can be considered

$$\dot{\bar{x}}_k \doteq col\,[\dot{\bar{x}}_{1k}, \dot{\bar{x}}_{2k}, ..., \dot{\bar{x}}_{n_k k}] \tag{3.3.1}$$

Together with the associated *k*-th *activation matrix*

$$A_k \doteq diag\,[a_{1k}, a_{2k}, ..., a_{n_k k}] \tag{3.3.2}$$

Then, the here proposed *task-priority control* procedure turns out to be represented, at each sampling interval and downstream the evaluation of the current status of *all* objectives within the whole prioritized list of M-objectives, by the implementation of the following, so-called, *task-priority Inverse Kinematic* procedure

- For $k = 1,2,...,N$ do (with $S_0 \doteq R^n$; $n = \#$ of dof's of the robotic structure)

$$S_k \doteq \left\{ \dot{y} = \underset{\dot{y} \in S_{k-1}}{arg\,R\text{–}min} \left\| A_k (\dot{\bar{x}}_k - J_k \dot{y}) \right\|^2 \right\} \tag{3.3.3}$$

With $\dot{x}_k \doteq J_k \dot{y}$ the current Jacobian relationship (to be evaluated via the information system) expressing the current actual rate-of-change of *k*-th *M-objective-vector*

$$x_k \doteq col\,[x_{1k}, x_{2k}, ..., x_{n_k k}] \tag{3.3.4}$$

Constituted by al scalar objective-variables characterizing the *k*-th M-objective. And where the introduced notation *R-min* is used for indicating an actually *regularized minimization*: which results strictly necessary for managing, with a suitable continuity, the algorithmic singularities (or quasi-singularities) that can arise when at least a task (i.e. a row in the argument of the above quadratic forms) is in its *transition zone*; that is when its associated activation function assumes values in between of its extreme ones, 0 and 1 (see [5]]). As it can be seen, procedure (3.3.3) is just structured (via the impositions $\dot{y} \in S_{k-1}\ ;\ k = 1,2,...,N$ at each priority level) accordingly with the philosophy of at each time-instant having the activities directed toward the achievement of all objectives within the *k*-th M-one (i.e. providing the best approximation of the currently provided task vector $\dot{\bar{x}}_k$) *without interfering* with those relevant to all others M-objectives exhibiting higher priorities.

Moreover, as it can also be noted, since the achievement of any one of the inequality-objectives included within a generic *k*-th M-objective, implies in (3.3.3) the smooth deactivation (accordingly with the smooth modulating-action performed by activation matrix A_k) of the corresponding reactive-task, and of the associated request for its best approximation; this has the nice consequence of releasing a corresponding degree of mobility; of which the remaining feedback-tasks within the same M-objective can directly benefit; as well as, even if less directly, also those included in the M-objectives with lower priority.

As regard the algorithmic translation of the above-introduced task-priority based control procedure (actually mainly represented by its inverse kinematic part) since being at each sampling interval based on a sequential minimization of quadratic cost functions, it consequently leads to a very simple *standard* algorithmic form; hereafter reported without proof (see however [4,5]) just for highlighting this aspect.

$$\rho_0 \doteq 0 \quad ; \quad \bar{G}_0 \doteq 0 \quad ; \quad Q_0 \doteq I$$

$$for \quad k = 1, 2, ... N \quad (G_k \doteq A_k J_k \ ; \ \dot{\bar{\sigma}}_k \doteq A_k \dot{\bar{x}}_k)$$

$$\bar{G}_k \doteq G_k Q_{k-1}$$

$$\left.\begin{array}{l} \dot{\bar{\rho}}_k = (I - Q_{k-1} \bar{G}_k^{\#} G_k)\, \dot{\bar{\rho}}_{k-1} + Q_{k-1} \bar{G}_k^{\#} \dot{\bar{\sigma}}_k \\ Q_k = Q_{k-1} (I - \bar{G}_k^{\#} \bar{G}_k) \end{array}\right\} \Rightarrow \quad S_k = \left\{ \dot{y} = \dot{\bar{\rho}}_k + Q_k \dot{z}_k \ ; \ \forall \dot{z}_k \right\}$$

---------- $\Rightarrow \quad S_N = \left\{ \dot{y} = \dot{\bar{\rho}}_N + Q_N \dot{z}_N \ ; \ \forall \dot{z}_N \right\}$

Where the notation $\bar{G}_k^{\#}$ is used for denoting the regularized pseudo-inversion of matrix $\bar{G}_k$, to be however performed accordingly with the methodology developed in [5]; which translates algorithmically the need for regularization that had been previously mentioned, at the time minimizations (3.3.3) was introduced

At this point it should also be easy to ascertain how, although we were explicitly referring to the most complex case of a vehicle/dual-arm system, as a matter of fact the *same*, structurally identical, procedure, leading to the *same*, structurally identical, algorithmic structure, can actually be applied, *without* any structural modification, also for developing the agent controller of any simpler robotic structure within the assumed class of them. Being the resulting differences only due to their generally differing Jacobian matrices. Such difference is however sufficient to justify the already remarked fact that, in case of reconfigurable scenarios populated by *heterogeneous* basic units, it is always preferable selecting, and then downloading, the agent controller relevant to the resulting reconfigured robotic structure, than instead resorting to coordination policies. However, as it also has been already remarked, in case instead of *homogeneous* basic units, as we shall better see at the next point 3.6, the sole agent-controller developed for the most complex case of the vehicle/dual arm type system, can actually suffice for also controlling all simpler ones within the class; thus almost totally avoiding context-switching efforts in correspondence of reconfigurations remaining in the class. At this point, before duly providing some examples, we simply conclude the present sub-section by introducing the terminological agreement of always denoting any *prioritized list* of M-objectives, more concisely as an *Action*.

3.4. Worked examples

Free-motion actions

For a vehicle/two-arms system requested to grasp, via its two arm-grippers, two objects (or two parts of a same object as well) located within the environment; whose respective transformation matrices ${}^{e1}_{g1}T, {}^{e2}_{g2}T$ of their fixed frames $< g_1>, < g_2>$ with respect to the end-effector ones $< e_1>, < e_2>$ are assumed at each time instant known; then a possible structuring for an action capable of driving the system toward the execution of such grasps, while however assuring maintaining its so-called safety and/or operational conditions always satisfied, could be for instance the one hereafter high-level described (1.e. substantially in a verbose form)

Action: Grasping Objects

M-Obj. 1: system with stable posture

M-Obj. 2: vehicle distance from obstacle above threshold

$$M\text{-}obj.3: \begin{cases} arms\ with\ joints\ inside\ limits \\ arms\ manipulability\ above\ threshold \end{cases}$$

M-Obj. 4: end-effectors toward the goals

M-Obj. 5: arms with preferred postures

Where the safety requirement (M-obj. 1) of having the system with a stable posture (i.e. the vertical projection of the system structural centre of gravity inside the vehicle convex-hull) has been located at the highest priority level, since deemed to be the most important one for obvious reasons. Then, the safety requirement (M-obj. 2), of having the vehicle with no risks of hitting possible obstacle (obviously provided they can be detected, possibly by fusing sensing and map information) follows as a requirement judged to be immediately less important than the highest priority one. Then the two safety-operability conditions for both arms (M-obj. 3) follow as the third ones; located at the same priority level and immediately prior the grasp objective (M-obj. 4), qualifying the ultimate action goal. With this last finally followed by the least priority one (M-obj. 5), simply introduced for avoiding, in case of redundant arms, possible drifting internal arm-motions, once the grasp is completed. Obviously the above choice is just one of many possible ones, because others M-objectives (mainly of safety/operational-enabling type) could also be added, at suitable priority levels, within the list (for instance the requirement of also avoiding the arms self hitting, and/or also having them avoiding external obstacles, could also be added; for instance as M-obj. 2.1 within the above list). In any case, as it can be inferred from the above-exemplified action structuring, the safety/operational enabling M-objectives generally have to precede the ones (but generally a sole one) related with the ultimate action-goal (in the above example M-obj. 4: grasping the object). And although such type of prioritized choice is not strictly necessary from the algorithmic point of view, for quite obvious reasons it has to be considered as preferable.

Once an action has been high-level organized, in order to complete its structuring, the following steps must be then executed: a) identify each one of the *N* M-objective-vectors (3.3.4), qualifying each M-objective in the priority list; b) associate each one of them with the corresponding M-feedback-task (3.3.1) and relevant activation matrix (3.3.2); c) identify for each one of them the corresponding Jacobian matrix. For instance, for the considered example, it is not difficult to see how the M-objective vector x_4 for the above M-obj. 4 (grasping the objects), can be defined as follows (if convergence is desired component-by-component)

$$x_4 \doteq col(x_{1,4}, x_{2,4}) \quad with \quad x_{1,4} \doteq col(\rho_1, r_1) \quad ; \quad x_{2,4} \doteq col(\rho_2, r_2) \tag{3.4.1}$$

Where ρ_1, r_1 represents the misalignment/distance vectors of goal frame $< g_1>$ with respect to the end-effector one $< e_1>$; and similarly for vectors ρ_2, r_2; all with components on a chosen frame (for instance the vehicle or the world one). Moreover, as it is well known, the real-time knowledge of such couple of double-vectors directly follows from the assumed real-time knowledge of transformation matrices ${}^{e1}_{g1}T, {}^{e2}_{g2}T$, respectively, via simple well known algorithms.

Then, since for the above vector x_4 its eventual zeroing is desired (let us assume component-by-component);

this makes M-obj. 4 formally resulting into the global equality objective

$$t \to \infty \quad \Rightarrow \quad x_4 = 0 \tag{3.4.2a}$$

It follows that its associated M-feedback-task 4 can be therefore just set as

$$\dot{\bar{x}}_4 \doteq -\gamma x_4 \quad ; \quad \gamma > 0 \tag{3.4.2b}$$

Obviously with associated activation matrix $A_4 = I$.

Moreover, since it is well known that *desiring* the above, just corresponds to *desire* the end-effector frames $< e_1 >, < e_2 >$ having their generalized absolute velocities (i.e. the stacking of the absolute angular and linear velocity vectors of each one of them; with components on the chosen frame) respectively equal to the half-upper and half-lower part of the above (3.4.2b), it then follows that the Jacobian relationship to be accounted for the M-reactive-task 4, just results into the absolute one of both the end-effectors, that is the following

$$\dot{x}_4 \doteq J_4 \dot{y} \quad ; \quad J_4 \doteq col(J_{e1}, J_{e2}) \tag{3.4.3}$$

Where J_{e1}, J_{e2} are respectively the absolute Jacobian matrices of the end-effector frames $< e_1 >, < e_2 >$, when separately considered; and with their output vectors projected on the same chosen frame; which are in turn, such Jacobians, they also directly provided by the real-time agent information system.

Thus, in summary, everything above in such a way to have the 4-th priority level of the task-priority inverse kinematic procedure, specified by the following set evaluation

$$S_4 \doteq \left\{ \dot{y} = \underset{\dot{y} \in S_3}{arg\, R\text{–}min} \left\| \dot{\bar{x}}_4 - J_4 \dot{y} \right\|^2 \right\} \tag{3.4.4}$$

As it regards instead the execution of the same, above-indicated, structuring steps a), b), c) also for the remaining others M-objectives in the list; and apart those directly involving the arms joints (first part of M-obj. 3 and M-obj. 5, which are de-facto straightforward), since space limitations unfortunately prevent to adequately illustrate them, reference is consequently made to the work [6], where a *general methodology* for the subject has been developed with due details.

Constrained-motion actions

For the same robotic structure underlying the considered systems, it is also of interest analysing the successive action phase, when, once a shared object has been eventually *firmly grasped* by both end-effectors, the system is then required to do its best for transporting the object (i.e. moving its attached frame $< b >$) to a location represented by a given goal-frame $< g >$.

In this situation, after having conveniently located both the arms tool-frames $< t_1 >, < t_2 >$ on the object frame $< b >$, we must formerly keep into account that the firm grasp assumption imposes, and will continue to impose at future times, the following *geometric constraints*

$$col(\rho_{b/t_1}, r_{b/t_1}) = col(\rho_{b/t_2}, r_{b/t_2}) = col(\rho_{t_1/t_2}, r_{t_1/t_2}) = 0 \tag{3.4.5a}$$

Where $\rho_{b/n}, r_{b/n}$ represents the misalignment/distance vectors of frame $< b >$ with respect to tool frame $< t_1 >$; and similarly for $\rho_{b/t2}, r_{b/t2}$; and also similarly for $\rho_{n/t2}, r_{n/t2}$, between the two tool-frames; all with components on a chosen frame within the system or the environment.

Obviously, among the above, only two of them can be chosen as independent ones, with the remaining other directly implied by the formerly chosen two.

Moreover the above geometric constraint necessarily also imposes the following *differential constraint* (from which also the related, hereafter reported, developments)

$$\dot{x}_b = J_n \dot{y} = J_{t2} \dot{y} \quad \Rightarrow \quad (J_n - J_{t2}) \dot{y} \doteq J_0 \dot{y} = 0 \quad \Leftrightarrow \quad \dot{y} \in Ker(J_0) = Span(I - J_0^{\#} J_0) \doteq S_0 \tag{3.4.5b}$$

Where the space of the admissible system velocity vectors, within firm grasp constraints, results established as indicate by the last membership condition

Thus, in order to account for the above restrictions within an action ultimately devoted to bi-manually transporting a shared firmly grasped object, the following modification to the previously seen object-grasping action, can consequently be introduced

Action: Transporting Shared Objects

M-Obj. 0: tool-frames commanded to comply grasp-constraints

M-Obj. 1: system with stable posture

M-Obj. 2: vehicle distance from obstacle above threshold

$$M\text{-}obj.3: \begin{cases} arms\,with\,joints\,inside\,limits \\ arms\,manipulability\,above\,threshold \end{cases}$$

M-Obj. 4: tool-frames toward the goal

M-Obj. 5: arms with preferred postures

Where the additional M-obj. 0 has been inserted at the *highest* priority level; just representing the need, in presence of physically imposed constraints, of also having the *commanded* motions satisfying these lasts; in such a way to always guarantee the avoidance of unfeasible motion requests, to which the *physically imposed* grasp constraints could consequently react via uncontrolled internal stresses imposed to the object (and to the end-effectors as well), that in turn might even damage such involved parts. As matter of fact, by adopting such philosophy of approach, whatever tool-frames motion will be commanded (thus among the admissible ones) will be reproduced without inducing any internal stress to the object; thus allowing decoupling the problem of object motion control, from its complementary one regarding object-stress control requirements. As already stated, interaction control will however not be considered here. Nevertheless it can be at least argued how, obeying to the above motion-command specifications, however constitutes the basis on which interaction-control can be then complementary grafted, as an homogeneous extension of the same task-priority structure used for motion control only. Details about this aspect can be found within the pioneering work [7]; and much more recently, with advancements, within [8, 9]

Thus, with the above specification in mind; and resorting to the formalism (3.1.1) for formally representing objectives; for the M-obj. 0 we should therefore require it to be the following

$$t \to \infty \;\Rightarrow\; x_0 \doteq col\,(\rho_{n/t2}, r_{n/t2}) = 0 \tag{3.4.6a}$$

That however, since "achieved and maintained since the very beginning" (in force of the physically realized firm grasp condition) it actually induces the associated M-feedback task to permanently provide a *null output*; that is, since the beginning physically having that

$$\dot{\bar{x}}_0 \doteq -\gamma x_0 \equiv 0 \quad ; \quad \gamma > 0 \quad ; \quad \forall t \tag{3.4.6b}$$

Then, since it is here also well known that *desiring* the above (even in case it were non-zero) just corresponds *desiring* the *difference* of the absolute velocities of tool-frames $<t_1>,<t_2>$ to be equal to the above (2.4.6b) (actually regardless from being it identically zero) it then follows that the Jacobian relationship to be accounted for M-reactive-task 0, necessarily results into the following

$$\dot{x}_0 \doteq J_0 \dot{y} \quad ; \quad J_0 \doteq J_{t_1} - J_{t_2} \tag{3.4.6.c}$$

With J_0 just defined as in (3.4.5b); as indeed could not have been otherwise.

To which the following minimization consequently corresponds (to be included at the highest priority level within the task-priority inverse kinematic part of the corresponding control procedure).

$$S_0 \doteq \left\{ \dot{y} = \underset{\dot{y} \in R^n}{arg\,R\text{-}min} \left\| \dot{\bar{x}}_0 - J_0 \dot{y} \right\|^2 \right\} \tag{3.4.6d}$$

That however, just in force of the identity condition (3.4.6b) induced by the physical firm grasp constraint, consequently leads to the set S_0 of the admissible commandable system velocities; just in the *same form* already established by (3.4.5);); thus imposing any commanded system-velocity vector always complying with the firm grasp constraint. Finally, for what concerns the M-obj.4, it also inserted within the above action high-level description, it actually remains equal to the one of the previously seen action (see (3.4.4)); apart however the fact of now requiring the velocities of tool-frames $<t_1>,<t_2>$, each one best approximating the, common to the two, reactive velocity desired for the object frame $<b>$ for asymptotically achieving the goal $<g>$; therefore represented by the following (assuming a component-by-component *desired* convergence)

$$\dot{\bar{x}}_b \doteq -\gamma x_b \quad ; \quad \gamma > 0 \quad ; \quad x_b \doteq col(\rho_b, r_b) \tag{3.4.7}$$

Where ρ_b, r_b now are the misalignment/distance vectors of goal frame $<g>$ with respect to the object one $<b>$ (with component on the same output frame of J_4 in (3.4.4)). Thus, in summary, in such a way to now have, within the same M-obj. 4 as in (3.4.4)

$$\dot{\bar{x}}_4 \doteq -\gamma\, col(\dot{\bar{x}}_b, \dot{\bar{x}}_b) \tag{3.4.8}$$

By the way, also note how the best approximation that will be then provided for the above velocity request, will be in any case with *identical* half-upper and half-lower parts; accordingly with the fact of having preliminary forced (at the highest priority level, via (3.4.6)) the system-velocity space to be compliant with the grasp constraint.

Before concluding this example, it is however deemed important remarking how, still in force of the firm grasp assumption, in lieu of considering M-obj. 0 in its complete form (3.4.6a-d) (i.e. by strictly obeying to earlier given definition for objectives) we could also have considered its *sole part* (3.4.6d) with $\dot{\bar{x}}_0 \equiv 0$; to be then inserted in the inverse kinematic part of the procedure as a velocity tracking task directly defined in the velocity space (thus *not supported* by any error feedback coming from the generalized position space, where instead objectives are defined). From now onward, tasks of this kind will be therefore termed as *M-tasks*, just for distinguishing them from any M-reactive-task instead included in a corresponding M-objective. From now onward, tasks of this kind will be therefore termed as *M-tasks*, just for distinguishing them from any M-reactive-task instead included in a corresponding M-objective. However, despite their conceptual difference (closed-loop tasks versus open-loop tasks) for the example in hand, since supported by the firm grasp assumption, the two choices obviously reveal as equivalent

The difference between the above choices instead reveals crucial whenever the firm grasp assumption may for some reason be looser (because for instance a not-sufficient grasp-pressure has been accounted to be exerted by part of the grippers). Thus meaning that in these cases, with adoption of the second choice (open-loop) the tool-frames could actually drift away without noticing it; while possibly uncontrollably dragging also the grasped object. While with the first choice (closed-loop), the system can instead reactively accommodate for such drifts (since measured as errors with respect to the required coincidence of the tool frames; while also imposing the necessary grasp-pressure increase for reactively immediately stopping the object drag. Thus meaning that the first choice should nbe generally considered the preferrable one. However, finally note that even in the example in hand, the need of explicitly introducing M-tasks, instead of M-objectives, might still arise: as for instance whenever we would like to have the vehicle minimally moving when not stricly required (because generally heavier and slower than the arms, more energy consuming, much less precise, etc.). Thus representing an additional requirement that can however be easily kept into account, by simply inserting, at the lowest priority level, the followiing final *M-task (N+1)*

$$S_{N+1} \doteq \left\{ \dot{y} = \underset{\dot{y} \in S_n}{arg\, R\text{–}min} \, \|v\|^2 \right\} \tag{3.4.9}$$

With v the vehicle velocity (i.e. the final three components of the overall system velocity vector $\dot{y}$)

3.5. Actions and missions

On the basis of the general terms description formerly given about the task-priority control procedure; then followed by few relevant worked examples, the following main aspects can however be now highlighted

a) Although actions can in-principle be organized by prioritizing very large varieties of objectives, in practice they

can always be reduced to a reasonable number of them, each one including all the safety/operational-enabling objectives reasonably expectable within a given work-field; so that they (the actions) will differ only for their end-effectors (or tool frames) ultimate goals; and consequently much more simply classifiable in terms of these lasts only (e.g. as it has been done in the previous examples), obviously plus the relevant ultimate-goal parameters.

b) Actions can be temporally sequenced accordingly with a *mission plan* (*mission* or *plan* for short) represented by a *decisional action graph*, having the actions as nodes and the arcs as logic decision alternatives. At each node, a logic decision should be the response to range-values assumed by the components of a suitably defined *action-state vector*, measuring the state of completion of an action in an aggregate form; preferably in terms of its ultimate end-effectors/tool-frames goal parameters only. Moreover, transitions from an action to another one located at the end of a selected arc, should however be smoothly activated.

c) The above possibility, offered by the task priority approach, of classifying the actions only in terms of their end-effector/tool-frames goals; thus without any concern regarding so many aspects and details not directly related with them (because reactively managed by the underlying agent controller) can in turn greatly simplify the mission-planning process (i.e. constructing the decisional action-graph), that consequently can be much more effectively let ranging from the simplest forms of pre-programmed plans, till arriving to include the most advanced techniques for the *automatic action-planning and re-planning* as in [10, 11, 12]; that in force of the allowed simplifications, may even be employed in real-time, within the whole operative scenario

3.6. Scaling/expanding robotic structures

It can be now instrumentally observed how, in case there might be some interest in having a vehicle/dual arm system behaving like a simpler robotic structure, for instance like a vehicle/single-arm, or a fixed-base/dual arm, or a fixed-base/single arm; or even solely as a vehicle; it would be then sufficient to simply insert, at a priority level immediately lower than the highest one, relevant to the posture-stability objective, the objectives relevant to the rest positions to be achieved by the not-to-be used sub-structures. So, in case such sub-structures were actually non-existing (simply because the considered structure is really a physically simpler one) but however provided the basic units of such reduced structure are the same *standard* ones composing the most complex vehicle/dual-arm system; then the agent controller of this last can be used also for the reduced structure; by simply expressing the non-existence of some parts in it as corresponding geometric constraints (i.e. satisfied at any time since the very beginning) to be imposed in correspondence of their "virtual" rest-positions. Moreover, in case of *standardized basic units*, the same idea of using the agent controller of the most complex structure also for the simpler ones, also applies in the reverse way, whenever a reduced structure (with standard units) is instead upgraded to a more complex one. And this by simply relaxing its virtual constraints in correspondence of the added parts. Note however that such very simple way of upgrading a reduced structure would not be applicable (even within the case of standardized basic units) whenever the agent controller of the starting reduced structure were not the constrained one of the most complex robotic structure within the class. Such considerations might therefore encourage toward the development of standardized basic units within reconfigurable scenarios; to the aim of practically avoiding any SW context switching in correspondence of any reconfiguration within the class; and consequently allowing even for on-fly reconfigurations.

4. Multi Agent Coordination Policies

4.1. Introductory Remarks

The case of two vehicle/single-arm agents occasionally required to cooperate in manipulating/transporting a common shared object, is now considered, as sketched in Fig. 3. Note that here also we assume that, upon the occurred firm grasp of the object, by part of both the end-effectors, the tool-frame of each system is conveniently transferred to the object-attached frame.

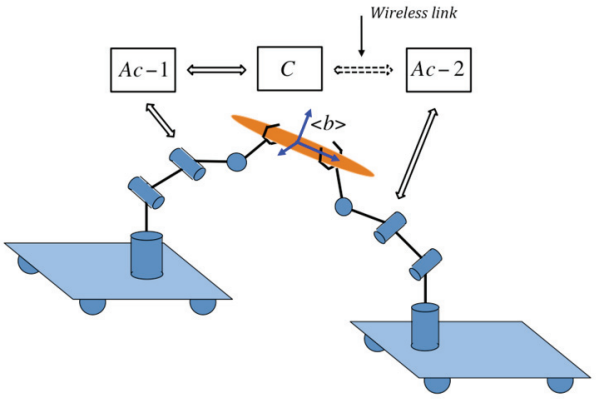

Fig. 3. Multi-agent cooperative Manipulation and Transportation

As already evidenced, in this case the development of a centralized agent-controller for the resulting global robotic structure, even if in-principle possible, from a practical point of view it results unthinkable; because (apart the occasionality of the situation) establishing a centralized agent-controller would require an unsustainable high-rate and huge amount of information to be real-time wireless exchanged between the two systems; thus making here unavoidable resorting to some coordination policy, operating on the basis of a reduced set of information exchanges.

Accordingly with what has also been already evidenced in section 2, provided possible, such coordination policy should preferably work on the basis of the sole information provided by variables and quantities strictly necessary to qualify the evolution of the cooperation objectives; that for the case in hand corresponds to primarily complying with the grasp condition (possibly within suitably controlled object-stress conditions); and secondarily to transfer the object to the desired location. By however, possibly, also letting the agents still with as much as possible liberty for continuing to best-care for their individual safety/operational-enabling objectives. To the achievement of the above aims, the following additional considerations, still regarding the firm grasp assumption, are however preliminary required. Obviously, here also, under the firm grasp assumption, the following differential constraints are imposed as consequence of the geometric ones (assumed written with components on a frame common to both systems; i.e. the environment frame or frame $<b>$; and actually no others).

$$\dot{x}_b = J_{t_1}\dot{y}_1 = J_{t_2}\dot{y}_2 \tag{4.1.1}$$

Which correspond to the same ones that were used in section 3, for formerly assessing the physically admissible system velocities for an individual vehicle/two-arm system, when similarly manipulating a shared grasped object. However the difference with the mentioned case of section 3, is here represented by the fact that the definition of such constrained space of overall system velocities, actually would result useful only in case we were searching for a centralized solution for the current problem; that however has been already judged as unthinkable. As consequence of this, and accordingly with the requirement (if possible) of resorting to coordination policies based on information exchanges restricted to variables and quantities characterizing the evolution of the sole cooperation objectives; it then follows that, in order to be fruitfully used, the above (4.1.1) actually needs to be preliminary transformed (if possible) in a form evidencing such variables and quantities, only. Fortunately enough, this can be done via the use of the hereafter deduced expressions for the involved Jacobian relationships, and relevant successive developments

$$\dot{x}_b = J_{t_1}J_{t_1}^{\#}\dot{x}_b = J_{t_2}J_{t_2}^{\#}\dot{x}_b \Rightarrow (J_{t_1}J_{t_1}^{\#} - J_{t_2}J_{t_2}^{\#})\dot{x}_b \doteq C\dot{x}_b = 0 \Leftrightarrow \dot{x}_b \in Ker(C) = Span(I - C^{\#}C) \tag{4.1.2}$$

Where, within each one of the Jacobian relationships, we have formerly eliminated the internal motion components

of the corresponding system velocity vector; thus transferring the original constraint form, from being expressed within the overall system velocity space, into the related one instead expressed in the Cartesian space. Then, upon the obvious definition of square matrix C, to be naturally termed as the *Cartesian Constraint Matrix*, the membership of $\dot{x}_b$ to the finally indicated spaces immediately follows; representing the space of the physically achievable object velocities at the current, overall-system constrained, configuration. Meanwhile this is also the space of the tool-frames velocities (and then object velocities) that can be *feasibly commanded* to *both* agents (because complying with the grasp constraint); with the warranty of having them executed by both these lasts; obviously provided that the corresponding common M-task is, here also, located at the *highest priority level* within the individual task-priority list of *both* the Agents. At this point, accordingly with the above conclusion and the expressed general requirements, the following coordination policy development can be therefore proposed.

4.2. Coordination Policy

For its development the following assumption are assumed satisfied:

a) The coordinator Sw is indifferently installed aside agent-controller 1 or 2 (let us say 1)

b) The transformation matrix ${}^{b}_{g}T$, between goal frame $< g >$ and the object-attached frame $< b >$, is at each sampling interval known to both the Agent controllers.

c) Each Agent-controller can therefore evaluate the current reference feedback velocity $\dot{x}_b^*$ that should be applied to the object for being asymptotically transferred to the desired goal location $< g >$

Then, during each sampling interval, the following actions sequentially take place, whose rationale is meanwhile explained

1) Each agent runs its individual task-priority control procedure as it were the sole one holding the object; while having its safety/operational-enabling objectives located at the higher priority levels.

   Thus resulting each one ending-out with the following, individually computed, respective tool-frame Cartesian velocities and related matrix quantities

$$(\dot{\hat{x}}_1 \; ; J_{t1} J_{t1}^{\#}) \quad ; \quad (\dot{\hat{x}}_2 \; ; J_{t2} J_{t2}^{\#}) \tag{4.2.1}$$

   Where the contained velocities $\dot{\hat{x}}_1, \dot{\hat{x}}_2$ will however be generally different; because the two Agents might be differently engaged in dealing with their individual, higher priorities, safety/operational-enabling objectives; and/or because endowing arms with different characteristics (different number of degrees of freedom, different topology and/or dimensions, etc.). Att this stage each agent is however ending-out by having evaluated what should be for it the best to do, for primarily caring about its own individual prioritized objectives.

3) Agent 2 wireless transfers its computed quantities to the Coordinator; that also locally acquires the homologous quantities from Agent-controller 1.

4) The Coordinator perform the following sub-steps

   4.1) Evaluates the barycentric velocity vector

$$\dot{\hat{x}} = \frac{1}{\mu}(\mu_1 \dot{\hat{x}}_1 + \mu_2 \dot{\hat{x}}_2) \quad ; \quad \mu = \mu_1 + \mu_2 \quad ; \quad \mu_1, \mu_2 > 0 \tag{4.2.2}$$

   Corresponding to suitably compromising between the two, generally differing, output velocities $\dot{\hat{x}}_1, \dot{\hat{x}}_2$, accordingly with a weighting policy that will be explained shortly.

   Note however how, in case the Agents have all safety/operational-enabling inequality objectives achieved; and also have their current Jacobian non-defective (thus $J_1 J_1^{\#} = J_2 J_2^{\#} = I$); then $\hat{x} = \hat{x}_1 = \hat{x}_2$ independently from the weights; and also feasible, because directly satisfying the grasp conditions.

   However, in the general case, when the individually so-computed end-effector Cartesian velocities are different, no-one of them generally satisfy the grasp condition; nor, most likely, their weighted sum $\dot{\hat{x}}$; that consequently needs to be projected on the space of the feasible object velocities, for just reducing it to be feasible. Thus explaining the reason for the following others steps

4.2) Evaluates the Cartesian constraint matrix (as above defined)

$$C = (J_1 J_1^{\#} - J_2 J_2^{\#}) \tag{4.2.3}$$

4.3) Evaluates the projection

$$\dot{\hat{x}} = (I - C^{\#} C)\,\dot{\hat{x}} \tag{4.2.4}$$

5) The Coordinator wireless transfers $\dot{\hat{x}}$ to Agent-controller 2, and locally to Agent-controller 1

6) Both Agents repeat their individual task-priority control procedure, by this time locating the common M-task of tracking $\dot{\hat{x}}$ at the *highest* priority level

7) Each Agent-controller then actuates its own resulting system velocities.

Finally, regarding the possible ways of assigning the weights μ_1, μ_2 within (4.2.2), the choice of setting them to be respectively of the form

$$\mu_1 = \mu_0 + \left| \dot{x}_b^* - \dot{\hat{x}}_1 \right| \quad ; \quad \mu_2 = \mu_0 + \left| \dot{x}_b^* - \dot{\hat{x}}_2 \right| \quad ; \quad \mu_0 > 0 \tag{4.2.5}$$

Actually seems quite reasonable; because, by interpreting the modulus terms in each one of them, as the aggregate measure of the ”difficulties” that the corresponding agent has in tracking the body referenced velocity $\dot{x}_b^*$ (because still engaged in primarily achieving its own safety/operational objectives; and/or because mechanically defective with respect to the other), it therefore appears advisable implementing (4.2.2) with weights (4.2.4), just for always acting in favour of the system currently resulting in a “weaker” condition between the two.

As it can be noted, the proposed coordination policy has been defined also in such a way to allow each agent exploiting the maximum residual motion freedom (compatible with the grasp constraint) for still continuing in primarily caring for their individual safety/operation-enabling objectives; and possibly also caring for additional safety objectives between them (e.g. avoiding the vehicles to collide, provided their mutual distance can be evaluated by each one of them, possibly via the same sensing-system individually used for detecting environmental obstacles).

Accordingly with the above remark, we can further note how, on the contrary to what is generally suggested in the literature, the proposed coordination policy does not a-priori impose any leader between the cooperating agents. In fact, the policy operates in such a way to naturally induce one of the agents to smoothly tend becoming a leader, only when it reveals “weaker” then the other agent in accomplishing the object transportation task; and similarly for the other agent. Naturally enough, no tendency to any leadership instead emerges, when they currently are in the same operative condition (same weakness or same strength) with respect to the common transportation task.

At this point, by assuming tht however it has not gone unnoticed the fact that the policy actually works via the insertion of a common M-task, instead than an M-objective; here again we might consequently have, in case of a loose the firm-grasp condition, the tool-frames drifting one-from the other.

To overcome such possible drawback, a solution that could actually reintroduce the needed M-objective in a decentralized way, could be for instance that of real-time evaluating the vectors $\rho_{t_1/t_2}, r_{t_1/t_2}$, relevant to the actual mutual location of the tool-frames (in turn evaluated from vectors $\rho_{e_1/e_2}, r_{e_1/e_2}$, relevant to the actual mutual position of the end-effectors, provided by a vision system endowed on on the agent hosting the coordinator) to be separately close-loop compensated by each agent. Such refinements, since still being the subject of investigations, will however be no further discussed here.

5. Simulations and Experiments

Some simulations and experiments have been executed, exhibiting satisfactory performances. Due to space limitations, we will however limit ourselves in presenting only few snapshot relevant to the case of two cooperating vehicle/single-arm systems; each one working accordingly with its own agent-control architecture of section 3; while being globally coordinated by the coordination policy of section 4. Accordingly with the targets of this work, the purpose of this section is therefore not the one of providing detailed quantitative descriptions of the achieved

simulative and experimental results; but instead that of in some way certifying how the proposed unified control framework allows managing complex cooperative activities The interested reader can however find in [8] many quantitative details, relevant to realistic simulations, even if regarding underwater robotic applications.

The first series of snapshot (Fig. 4) refers to a simulation where the vehicles are both of non-holonomic type; thus meaning that, starting from the agent-controller architecture developed for holonomic vehicle/dual-arm systems (till now implicitly assumed) the following two *differential constraints*

$$(I - i_1 i_1^T) v_1 = 0 \;\; ; \;\; (I - i_2 i_2^T) v_2 = 0 \qquad \text{equivalent to} \qquad [i_1 \wedge] v_1 = 0 \;\; ; \;\; [i_2 \wedge] v_2 = 0$$

With $i_1 ; i_2$ the unit vector along the longitudinal direction of the corresponding vehicle (each one with components on the corresponding vehicle-frame), had to be therefore inserted as the M-tasks with the highest priority, within the corresponding Agent-control procedure.

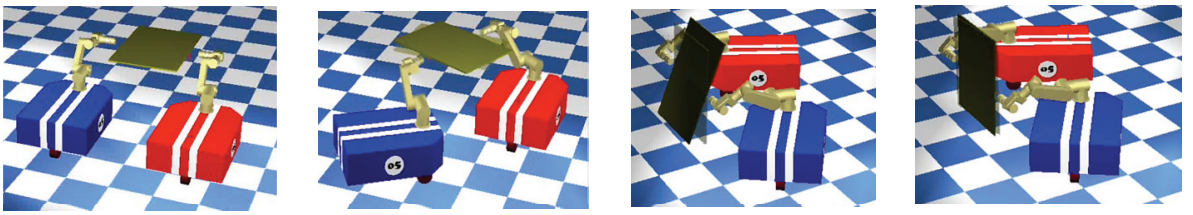

Fig. 4. Simulation of cooperating non-holonomic Mobile manipulators

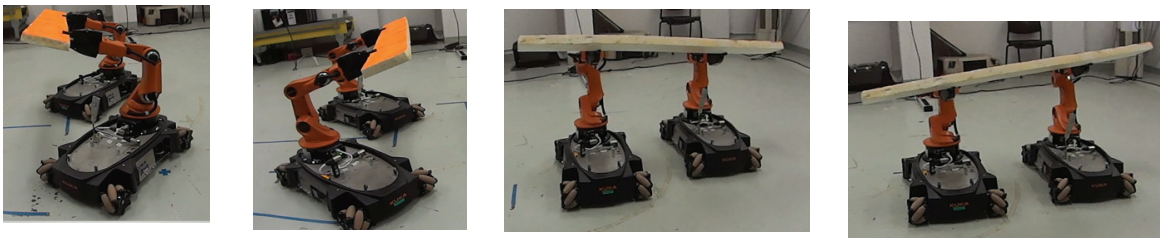

Fig. 5. Experimental trials with holonomic cooperating mobile manipulators

The second snapshot series instead regards experimental trials performed by using a couple of small KUKA You-Bots; which are instead with holonomic vehicles, in force of their adopted special omnidirectional wheels (Fig. 5). In the above experimental trials, communication among the agents was wireless occuring, not directly between them, but instead via the Wi-Fi net of the lab. Moreover, the absolute localization of each vehicle (when needed by the agents, or even by the sole coordinator, for real-time evaluating the object reference feedback $\dot{x}_b^*$ toward the goal frame) was assured by a motion-capture sytem installed on the Lab ceiling; real-time providing the information to the corresosponding agent, still via the same Wi-Fi net of the Lab. The achieved communication badwidth actually revealed very compatible with the used control sampling times.

Conclusions

The paper ha presented the development of a unified functional, algorithmic and Sw architecture, adoptable as a standard KCL for controlling, at action level only, any robotic structure belonging to a given wide class of them, even of reconfigurable type, within the class; and consequently deemed very suitable for operating within factory and/or logistic, possibly reconfigurable, scenarios. Moreover, for the few cases of cooperative activities between agents not allowed to be cable connected, an effective coordination policy, based on the exchange of a reduced information set, only regarding the cooperation goal, has been developed and experimented. Moreover, the advantage of having, in whatever cooperative condition, the possibility of commanding the involved structures only in terms of the ultimate goals of each action, seems to be the right basis for also having non-negligible improvements within their integration with automated action planning and learning techniques.

References

[1] Y. Nakamura: “Advanced Robotics: Redundancy and Optimization”. Addison Wesley, Reading, MA, 1991
[2]. B. Siciliano, J.J. E. Slotine: “A General Framework for Managing Multiple Tasks in Highly Redundant Robotic Systems”. In Proc. Fifth Int Advanced Robotics ’Robots in Unstructured Environments’, 91 ICAR. Conf. Pisa, Italy: IEEE, 1991.
[3] E. Simetti, G. Casalino, S. Torelli, A Sperinde, A. Turetta: “Floating Underwater Manipulation: Developed Control Methodology and Experimental Validation within the TRIDENT Project”. Journal of Field Robotics, vol. 31, no. 3, pp. 364–385, May 2014.
[4] E. Simetti , G. Casalino, “Whole Body Control of a Dual-arm Underwater Vehicle Manipulator System”. Annual Reviews in Control, vol. 40, pp. 191–200, 2015.
[5] E. Simetti, G. Casalino, “A Novel Practical Technique to Integrate Inequality Control Objectives and Task Transitions in Priority Based Control”. Journal of Intelligent & Robotic Systems, vol. 84, no. 1, pp. 877–902, 2016.
[6] G. Casalino, E. Simetti: “A Study on Coordination and Control of Floating Manipulators”. Internal Report GRAAL (Genoa Robotics and Automation Lab), University of Genoa, January 2015.
[7] G. Casalino, D. Angeletti, T. Bozzo, G. Cannata: “Strategies for Control and Coordination within Multi-arm Systems”. In S. Nicosia, B. Siciliano, A. Bicchi, P. Vaigi (Eds.): “RAMSETE-Articulated and mobile robotics for services and technologies”, Lecture Notes in Control and Information Science, Springer, 2001
[8] E. Simetti, G. Casalino: (2016). “Manipulation and Transportation With Cooperative Underwater Vehicle Manipulator Systems” IEEE Journal of Oceanic Engineering, p. 1-18, 2016 ISSN: 0364-9059.
[9] E. Simetti, G, Casalino, F. Wanderling:”Robotized Underwater Intervention”. Lecture Notes in Control and Information Science, Springer, 2017
[10] L.S.H. De Mello, A.C. Sanderson: “ AND/OR Graph representation of Assembly Plans”. IEEE Transactio on Robotics and Automation, Vol. 6, No 2,1990.
[11] K.Darvish, B. Bruno, E. Simetti, F. Mastrogiovanni, G Casalino: “An adaptive human-robot cooperation framework for assembly-like tasks Proceedings of the Third Italian Workshop on Artificial Intelligence and Robotics (AIRO 2016), Genova, Italy, December 2016.
[12] L. Johannsmeier, S. Haddadin: “A hierarchical human-robot interaction-planning framework for task allocation in collaborative industrial assembly processes”. IEEE Robotics and Automation Letters, Vol. 2, No 1, pp 41–48, 2017.