Taylor & Francis
Taylor & Francis Group

Journal of Experimental & Theoretical Artificial Intelligence

ISSN: 0952-813X (Print) 1362-3079 (Online) Journal homepage: www.tandfonline.com/journals/teta20

Model-free learning on robot kinematic chains using a nested multi-agent topology

John N. Karigiannis & Costas S. Tzafestas

To cite this article: John N. Karigiannis & Costas S. Tzafestas (2016) Model-free learning on robot kinematic chains using a nested multi-agent topology, Journal of Experimental & Theoretical Artificial Intelligence, 28:6, 913-954, DOI: 10.1080/0952813X.2015.1042923

To link to this article: https://doi.org/10.1080/0952813X.2015.1042923

Journal of Experimental & Theoretical Artificial Intelligence, 2016
Vol. 28, No. 6, 913–954, http://dx.doi.org/10.1080/0952813X.2015.1042923

Taylor & Francis
Taylor & Francis Group

Model-free learning on robot kinematic chains using a nested multi-agent topology

John N. Karigiannis and Costas S. Tzafestas*

National Technical University of Athens (NTUA), School of Electrical and Computer Engineering, Zographou Campus, Athens 15780, Greece

(*Received 21 May 2013; accepted 16 November 2014*)

This paper proposes a model-free learning scheme for the developmental acquisition of robot kinematic control and dexterous manipulation skills. The approach is based on a nested-hierarchical multi-agent architecture that intuitively encapsulates the topology of robot kinematic chains, where the activity of each independent degree-of-freedom (DOF) is finally mapped onto a distinct agent. Each one of those agents progressively evolves a local kinematic control strategy in a game-theoretic sense, that is, based on a partial (local) view of the whole system topology, which is incrementally updated through a recursive communication process according to the nested-hierarchical topology. Learning is thus approached not through demonstration and training but through an autonomous self-exploration process. A fuzzy reinforcement learning scheme is employed within each agent to enable efficient exploration in a continuous state–action domain. This paper constitutes in fact a proof of concept, demonstrating that global dexterous manipulation skills can indeed evolve through such a distributed iterative learning of local agent sensorimotor mappings. The main motivation behind the development of such an incremental multi-agent topology is to enhance system modularity, to facilitate extensibility to more complex problem domains and to improve robustness with respect to structural variations including unpredictable internal failures. These attributes of the proposed system are assessed in this paper through numerical experiments in different robot manipulation task scenarios, involving both single and multi-robot kinematic chains. The generalisation capacity of the learning scheme is experimentally assessed and robustness properties of the multi-agent system are also evaluated with respect to unpredictable variations in the kinematic topology. Furthermore, these numerical experiments demonstrate the scalability properties of the proposed nested-hierarchical architecture, where new agents can be recursively added in the hierarchy to encapsulate individual active DOFs. The results presented in this paper demonstrate the feasibility of such a distributed multi-agent control framework, showing that the solutions which emerge are plausible and near-optimal. Numerical efficiency and computational cost issues are also discussed.

Keywords: multi-agent architecture; fuzzy reinforcement learning; developmental robotics; dexterous robotic manipulation

1. Introduction

Understanding aspects of coordinated behaviours in the frames of multi-agent systems and elaborating distributed learning schemes for application on collaborating robots constitutes an important research topic in robotics. In particular, a lot of work has already been done mainly in

*Corresponding author. Email: ktzaf@cs.ntua.gr

the field of cooperating mobile robots, where several approaches have already been proposed and experimentally tested. For instance, a team of mobile robots has been demonstrated in Donald et al. (1997) to perform a collaborative task, which consisted of multi-robot cooperation to efficiently push large boxes based on a set of specific manipulation protocols. Similar tasks have been reported in Rus (1997), where a specific algorithmic structure has been used to coordinate the reorientation of objects on a plane by independent robot-agents, and in Ahmadabadi and Nakano (2001), where distributed cooperation strategies have been employed by a group of behaviour-based mobile robots for jointly handling an object. A common ground in all these approaches is that the motion of the object under cooperative pushing and manipulation is quasi-static, and that all the agents involved have predefined behaviour models that they combine by employing some pre-specified control architecture.

Cooperative behaviour is one of the aspects studied within a multi-agent framework. Human behaviours also demonstrate evolutionary characteristics and self-organising abilities. These unique attributes of human behaviours are extensively studied in the process of designing intelligent robots that need to operate and collaborate autonomously adapting to their environment. In this context, the application of bio-inspired techniques, such as reinforcement learning (RL), evolutionary computation and fuzzy systems, constitutes an emerging research field. In RL approaches (Sutton & Barto, 1998), new skills are acquired by means of a trial-and-error process (Bertsekas & Tsitsiklis, 1996; Dayan & Abbott, 2001), which is based on exploring the robot's own actions and obtaining consequent perceptual observations on its environment. State–action policies are thus progressively developed based on the definition of reward functions that act as positive reinforcement or negative punishment depending on the performance of the robot with respect to the desired goal.

RL is an active area of machine learning research that is also receiving attention from the fields of decision theory, control engineering and human–machine interaction (Lemon & Pietquin, 2012), and is progressively finding an increasing number of new applications in the field of robotics (Kormushev, Calinon, & Caldwell, 2013; Stulp, Buchli, Theodorou, & Schaal, 2010). Various RL methods have been employed on multi-agent architectures that target the control of mobile robots operating within a fully or partially observable environment (Kok & Vlassis, 2004, Murciano, Millan, & Zamora, 1997). Moreover, in Takahashi et al. (2001) and Doya (1996), we have seen cases where single agent architectures employ RL methods in a continuous three-dimensional (3D) space, implemented by neural networks. In Liu et al. (2007), a three-layered architecture is introduced (namely, motion patterns, behaviour models and planning component), which employs RL in order to control a robotic fish. In general, RL is an approach where building a policy is based on data acquired through exploration.

Another approach to acquire robot manipulation skills is through learning from demonstration (LfD) (Argall, Chernova, Veloso, & Browning, 2009), also referred to as imitation learning (Schaal, 1999) or programming by human demonstration (Billard, Calinon, Dillmann, & Schaal, 2008; Kaiser & Dillman, 1996). By LfD, instead of learning by exploration, a policy is learned from examples or demonstrations provided by a teacher (that is, a human acting as the tutor of the robot). These examples are defined as sequences of state–action pairs that are recorded during the teacher's demonstration of the desired robot behaviour. Such a method, employing RL to achieve skill learning by demonstration, has been proposed in Pastor et al. (2011). This approach is based on a centralised (single-agent) framework, which enables the generalisation of knowledge acquired during demonstration, while providing policy improvement with path integrals. A relevant methodology is presented in Lopes and Santos-Victor (2007), proposing a hierarchy of developmental stages and skills that have to be acquired at each stage. The initial steps of skill acquisition in this hierarchy allow the robot to establish

sensorimotor coordination. The next step in the hierarchy allows building skills by interacting with the environment, while the last step introduces skill acquisition by imitation of teacher's demonstrating examples.

Other relevant research efforts have been conducted in the field of humanoids and human–robot interaction, also aiming to endow robots with a capacity to learn new motor skills. Work in this field is often based on probabilistic approaches, like for instance in Calinon et al. (2010), where a combination of hidden Markov models and Gaussian mixture regression (GMR) has been applied for the learning and reproduction of gestures by imitation. Relevant work has also been recently reported in Rozo et al. (2013), regarding the transfer of physical interaction (impedance-based) skills by direct kinesthetic teaching. Guenter, Hersch, Calinon, and Billard (2007) have also explored the use of Gaussian mixture models and GMR to encode motor skills that can be learned by demonstration through expectation–maximisation techniques, applied to constrained reaching movements. Other approaches for skill representation include (1) dynamic motor primitives (Kober & Peters, 2009; Schaal, Mohajerian, & Ijspeert, 2007), where RL is used to adapt the position of specific movement attractors, and (2) motor coordination schemes, such as the one proposed in Pardo et al. (2009) that has been applied to the learning of simple rest-to-rest movements. The challenges that all these approaches are still facing relate to the task requirements that are inherent in human–robot systems, involving large multi-dimensional and continuous state–action spaces, as well as operation in constrained, dynamic and partially unknown environments.

Inspired by all the above research directions and the challenges that they face, this paper proposes a model-free learning scheme for developmental acquisition of robot manipulation kinematic control skills. The approach is based on a multi-agent architecture, which resides on an original nested-hierarchical structure that encapsulates naturally the topology of robot kinematic chains. Within the proposed multi-agent architecture, the activity of each joint (link or, more generally, degree-of-freedom, DOF) of a kinematic chain is represented by a distinct agent at the lower level of the hierarchy. Each one of those agents maintains a local (i.e. partial) view of the whole system topology and of the task progress, which is incrementally updated through a recursive (top-down, bottom-up) communication process. Learning is then approached not through demonstration and training, but through an autonomous exploration and self-learning (unsupervised) process, where each agent (joint) evolves a local sensorimotor behaviour by receiving information (reward signals) related to observations of task performance.

The main motivation behind the development of such an incremental multi-agent topology is to enhance system modularity, to facilitate extensibility to more complex problem domains and to improve robustness with respect to structural variations including unpredictable internal failures. These attributes of the proposed system are assessed in this paper through numerical experiments in different manipulation contexts, involving single- or multi-robot kinematic chains. In particular, two series of numerical experiments are performed to evaluate the performance of the proposed multi-agent learning scheme. The first experimental series addresses the problem of kinematic control for redundant multi-DOF robotic chains. Generalisation capacity of the learning mechanism is experimentally assessed and robustness properties of the multi-agent system are evaluated with respect to unpredictable variations in the kinematic topology. Furthermore, extensibility of the system to more complex task scenarios is demonstrated by considering a direct extension of the above multi-agent learning system into a constraint manipulation task, involving redundant manipulation under additional obstacle avoidance conditions. Scalability of the system is further evaluated in the second experimental series presented in this paper, which addresses a more complex robotic task scenario that

consists of a simulated dexterous manipulation set-up involving multiple cooperating robot chains in a quasi-static multi-finger grasping task.

Besides inherent topological scalability, an important issue also concerns numerical efficiency and convergence of the learning mechanism, especially when this multi-agent system addresses a robot manipulation task in a continuous state–action domain. This paper constitutes in fact a proof of concept demonstrating that such global manipulation skills can indeed evolve through distributed iterative learning of local agent sensorimotor mappings. Indeed, in the proposed architecture, each agent is selecting an action independently of the rest, based on local state information and by observing and generating an estimate of what the rest will do in a game-theoretic sense (that is, always as viewed in the local perspective of each agent, with limited access only to those agents that are *visible* according to the nested-hierarchical topology). Efficient exploration in a continuous state–action domain is then achieved by employing a fuzzy RL scheme. The resulting cumulative action of the system is a joint effort (joint actions) of all the nested entities comprising the system. Although autonomous, the agents are closely coupled with each other due to the physical connectivity, making the effective cooperation and coordination among them extremely important to achieve a convergent behaviour for the whole system. This paper demonstrates the feasibility of such a distributed multi-agent control framework in the domain of dexterous manipulation, a domain where very few related experimental results, regarding developmental control strategies, are available in the literature. This paper also demonstrates that the solutions which emerge (at least, in the single redundant kinematic chain) are plausible and near-optimal (in a least-squares (LS) sense). Finally, it is shown that this multi-agent architecture provides significant computational cost benefits as compared to a centralised single-agent approach.

The paper is organised as follows. The next section discusses related work and highlights the contributions of this paper. Section 3 outlines the proposed multi-agent framework, describing the topology of the nested-hierarchical architecture and the basic algorithmic structure employed for multi-agent control. Section 4 describes more in detail the learning approach employed in this work, an RL-based methodology focusing both on state-space continuity and localised action selection. Section 5 presents the numerical experiments performed and discusses the results obtained particularly in two case studies: (i) a simulated single kinematic chain and (ii) a three-finger dexterous manipulation task (with four DOFs per finger). Section 6 discusses issues related to the computational cost of the proposed multi-agent framework. Concluding remarks and future work directions are presented in Section 7.

2. Related work and contribution

In the previous introductory section we have discussed in general some approaches that employ LfD in solving relevant problems. Although these methodologies advance the state-of-the art in using human demonstration as a bootstrap step in robot learning, they differ inherently from the approach that is proposed in this paper, particularly in the sense that they are basically centralised (single-agent) approaches. The main contribution and originality of our approach, with respect to existing LfD-based methodologies, resides principally in the fact that we employ a nested-hierarchical multi-agent learning architecture and demonstrate how such a distributed system can evolve coordinated complex behaviours, with each agent participating at its own level in a cooperative game with the rest of the agents. In this paper we investigate how robotic skills, such as goal reaching, constrained motion coordination and cooperative grasping, can be acquired autonomously and efficiently, with high degree of generalisation and robustness, without any initial bootstrap learning step performed by human demonstration. Representing a

dexterous robot manipulation skill acquisition and learning problem as a cooperative multi-agent game, as proposed in this paper, is a feature that, to the best of our knowledge, has not been addressed in previous work.

The research described in this paper extends our previous work presented in Karigiannis et al. (2010, 2011). In Karigiannis et al. (2011), the main concept of the nested-hierarchical multi-agent architecture has been presented and general descriptions about how this framework could be embedded on robotic systems have been provided together with some preliminary results. In Karigiannis et al. (2010), the learning algorithm has been extended to address problems without natural proximity (i.e. no physical relation) between the DOFs handled by the different agents. Some early results have been reported regarding the case of two mobile robots performing a box-pushing task without any prior built-in model of the task. The work that is presented in this paper covers a full theoretic description of the proposed approach, including the presentation of the fuzzy Q-learning mechanism and the joint action selection algorithm employed. Moreover, extended numerical experiments have been conducted and the results are presented and analysed in this paper, addressing new performance aspects of the system, namely knowledge generalisation, robustness to unexpected failures of individual agents and scalability of the proposed framework to more complex tasks (such as hyper-redundant robot kinematic chains performing constrained manipulation task). The work presented in this paper is inspired by and builds upon related work conducted in the fields of multi-agent RL and control of complex kinematic chains. In the sequel, we briefly review prior work from these two areas and discuss relevance to the approach proposed in this paper, aiming to further highlight its originality and contribution.

2.1 Multi-agent reinforcement learning

Claus and Boutilier (1998) have formulated how Q-learning can be applied in multi-agent systems by using the notion of a Nash equilibrium for describing the optimal joint action. Our hierarchical architecture is based on the same formulation and manages to reduce the complexity of the learning problem by assuming that the multi-agent system is composed of agents that have the same state and action space. A different approach has been proposed by Guestrin, Lagoudakis, and Parr (2002), where a structured representation of the multi-agent RL process is described. The coordination requirements of the system are supposed to be captured by a connectivity graph, representing agents as nodes and direct coordination requirements between agents as edges. The optimal joint action of the system can then be computed by using variable elimination on the coordination graph.

Using a coordination graph for multi-agent learning requires having precise knowledge of the agents' fixed connectivity and topology. Instead, the hierarchical architecture that will be presented in this paper does not need a-priori knowledge on the exact topology of the agents but provides an alternative way of coordinating the agents by assuming a hierarchical chain among them, which may in fact be arbitrary without necessarily making any assumption on the precise physical connectivity of the agents.

A relevant multi-agent RL approach has been proposed in França et al. (2014), where by using a hierarchical partitioning of the input space, a group of agents (predators) are learning how to coordinate their activities in order to capture another agent (prey). Without a global coordination mechanism, each agent defines its objective individually based on its perception of the environment. The system generalises the knowledge obtained and hence the agents-predators manage to capture the agent-prey despite the fact that it is located in random locations. The main difference with our approach is that the method proposed in França et al. (2014) operates

in a discrete state space, and robustness issues concerning potential failures in the agents composing the multi-agent system are not addressed. For instance, there are no results demonstrating how the system may react in the case that agents-predators experience unexpected failures and hence the number of operational agents-predators is suddenly reduced.

Another multi-agent RL approach is proposed in Zhang and Lesser (2013). The learning scheme in this approach is similar to what we propose in this paper, in the sense that the agents ignore the actions and rewards of the other agents, and concurrently learn their own action-value functions solely based on their local observations and rewards. Every agent communicates with only a reduced set of agents comprising the system in order to minimise the cost of communication. This is done by introducing for each agent an interaction measure which identifies the maximum expected utility that an agent can potentially receive when it coordinates with a subgroup of agents. Based on this expected utility, each agent dynamically decides to increase or decrease the number of agents with which it establishes communication. In our approach, however, the communication and goal formation scheme is essentially different. As will be presented in the subsequent sections, the communication between agents is defined based on the connectivity specified by a nested-hierarchical topology, which adapts intuitively to the requirements of cooperative robot manipulation tasks, also meaning that each agent effectively communicates only with a specific set of neighbouring agents without having to maintain any utility measure.

2.2 Motion planning for robot kinematic chains

Much of the work on manipulation planning using open kinematic chains is covered by approaches employing randomised path planning algorithms, such as probabilistic roadmap planners (Kavraki, Svestka, Latombe, & Overmars, 1996) and rapidly exploring random trees (LaValle & Kuffner, 2001). In Yakey et al. (2003), for instance, initial random configurations of the kinematic chains are generated and, by rejecting those resulting in a collision, neighbour configurations are identified through random steps in a tangent space. By sampling in the tangent space, a search is actually performed to identify possible neighbour configurations that the kinematic chain can move to. Hence, by performing this process recursively, the system manages either to reach a goal or to perform a grasp without using any predefined model.

All these approaches constitute the state-of-the art in robot path planning. These models assume, however, exact knowledge of the overall kinematic chain topology (open or closed) as well as centralised control of the system, and the problem is thus approached in a single-agent framework. Moreover, all the obstacles are generally considered to be static and the world fully observable, while unexpected failures of joints during executions constitute an open problem. A relevant very interesting work has been presented by Boularias, Krömer, and Peters (2012), where by employing a graph-based algorithm for apprenticeship learning, solution to a grasping problem is obtained on a discrete grid-world. In this case, a structured representation (i.e. a model) is required in the form of a graph that has to specify all states and relate them to actions. Moreover, this graph should indicate which pairs of states are supposed to have similar optimal actions.

As compared to the above methodologies, our approach does not require any graph model to associate states with optimal actions, since this mapping is actually derived by the agents autonomously. In our case, a collaborative multi-agent exploration process is introduced, for the autonomous development of skills related to tasks such as reaching a global goal or performing a cooperative quasi-static grasp. This approach has the benefit of handling unexpected failures through error self-recovery, where the system is able to adapt guided by the reward received

from the environment, without requiring hard assumptions about static obstacles and fully observable task space. It should be clarified at this point, though, that the main limitation of our approach, as for every model-free approach, is mainly related to the accuracy of the solutions it provides. Therefore, one could definitely argue that safety critical systems cannot be handled by such pure model-free architectures. In such cases, what is actually being proposed by our work is to encapsulate such a model-free approach on a higher control level (which could handle uncertainty, error self-recovery capacity and knowledge generalisation), driving lower control layers that could be model based and could provide the accuracy and speed required by a safety-critical system.

3. The proposed nested-hierarchical multi-agent framework

The multi-agent control framework proposed in this paper fits in the context of a continuous research effort aiming to explore architectures that would enable a complex robotic system to autonomously develop and progressively acquire control skills in a modular, scalable and robust manner, without the need for tedious task modelling and restrictive pre-programming. Therefore, the basic motivation underlying this research work derives from the quest for robot control systems that could exhibit some form of capacity for autonomous, *developmental skill acquisition* through self-experimentation and unsupervised self-learning, particularly without the application of any pre-programmed (model based) planning scheme. Another important requirement is *modularity* and direct *scalability* to increasingly complex kinematic topologies and task scenarios, without resorting to any significant modification of the control structure and of the algorithmic implementation involved. *Robustness* to potential internal structure variations, as well as *decentralisation and parallelism* are also important control requirements.

Decentralised multi-agent architectures present inherent advantages with respect to the above system requirements, as compared to centralised control approaches. In this research context, this paper proposes a multi-agent learning architecture based on a nested-hierarchical structure that intuitively encapsulates the topology of robot kinematic chains. Developmental skill acquisition properties are then explored by employing a fuzzy RL scheme in a continuous state–action domain (as will be described in Section 4). Figure 1 presents the basic structure of

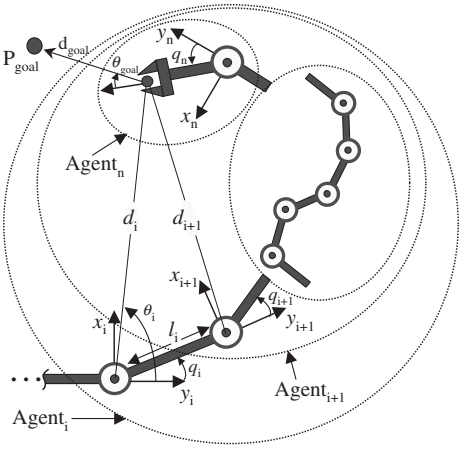

Figure 1. The proposed nested-hierarchical multi-agent topology encapsulating an *n*-DOFs robot kinematic chain; in this case, moving the end effector at a given goal position is the considered manipulation task.

the *multi-agent nested-hierarchical topology* proposed in this paper to encapsulate robot kinetic chains and formulate a distributed learning scheme for robot manipulation kinematic control skills. The basic features and requirements of this multi-agent control framework are described in the sequel.

3.1 Mapping agents to DOFs

Each robot joint is assigned to an agent having as a function to govern local control at that joint level. The challenge here is to evolve global dexterity through progressive acquisition of local skills at each local agent level. It should be noted here, however, that although every joint is assigned to an agent, the reverse definition is different. An agent can indeed represent more than a single DOF, since it can also encompass other agent(s), situated at a lower level of the nested-hierarchical structure. So, we could have an agent representing in fact a compound kinematic linkage and internally comprising two or more individual DOFs.

3.2 Local agent neighbourhood in the nested-hierarchical topology

Figure 2 presents a schematic representation depicting an example of the nested-hierarchical multi-agent topology. In this topology, every agent can communicate only with those agents that are situated one level below in the hierarchy. These 'visible' agents, in the nested-hierarchical sense, are considered to form the local neighbourhood of the agent. For instance, in the case of agent a_{i+2} in Figure 2, the visible agents are only a_{i+3} and a_{i+4}. This approach provides the multi-agent system the ability to perform a recursive and incremental state definition, as will be described in the following section.

Each agent functions locally and selects an action independently of the rest, by forming an estimate of the actions that the rest of the agents could potentially perform. This is of course always viewed in the local neighbourhood of each agent, that is, with limited visibility only to those agents that are located below in the nested-hierarchical sense; in other words, each agent maintains a partial (local) view of the system topology and does not know how many agents constitute the whole system above or below and what is the exact underlying topology.

However, a measure of global task performance is assumed to exist, provided by the top-level (root) agent in the hierarchy, and distributed to lower agents in the nested architecture, guiding in that manner the RL process through the computation of a reward function. This reward function must be computed on a continuous scale (instead of waiting for a discrete event of type success or failure to occur), leading to a continuous adaptive dynamic behaviour for the system. Although autonomous, the agents are closely coupled with each other due to the

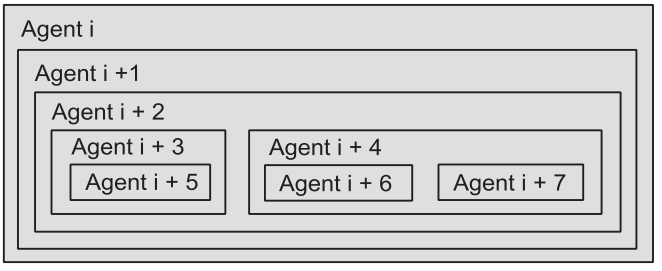

Figure 2. Example of a nested-hierarchical multi-agent topology.

physical connectivity, making the effective cooperation between them extremely important to achieve a convergent behaviour for the whole system.

3.3 Continuous problem setting and state definition

The learning process must function in a continuous state-space for the system to establish manipulation skills. For the methodology proposed in this paper, a fuzzification step is applied to the readings forming the system state. Learning is then accomplished in a discrete state–action mapping sense; a defuzzification step can subsequently be employed to perform action selection in a continuous domain.

Let us consider a kinematic chain that comprises n DOFs (agents $a_i : i = 1, \ldots, n$), nested in the manner presented in Figure 1. We define the state of every agent a_i as $S_i = \langle q_i, \theta_i, d_i, \vec{g}_i \rangle$, where q_i refers to the generalised displacement for the ith DOF (in the case of a simple revolute joint, it is the relative angular displacement between the two consecutive links in the kinematic chain), d_i and θ_i are the polar coordinates (radial distance and angle, respectively) of the robot's end effector with respect to the local reference frame of the ith agent, and $\vec{g}_i$ is a vector that describes the current position of the goal at the task space with respect to the end effector ($\vec{g}_i$ comprises in total of three variables: two for position and one for orientation). That means the state of each agent comprises six variables. We will be referring to the root agent as simply being the top agent in this hierarchy, sharing the same architectural structure with all other agents that are located below in the hierarchy. More about the proposed multi-agent architecture will be discussed in the following section.

Figure 3 depicts the flow of operations at an agent level, constituting the basic algorithmic structure of the proposed multi-agent control framework. Each agent in the system contains links only to those agents that are located at the level immediately below in the hierarchy and are considered 'embedded' in the nested hierarchical sense (as well as a link to its parent agent in the hierarchy). Every agent a_i identifies certain variables of its state (constituting a partial-local-view of the full system topology) and forwards that information in a nested manner to the agents of the next level in the hierarchy, in order to facilitate the process of identifying their own state variables. This whole process is composed of two sub-processes that evolve in a recursive manner. The first one is evolving in a top-down manner, while the second one is evolving in a bottom-up manner. The top-down sub-process starts from the root agent of the hierarchy and travels to the lower ones. During this phase, every agent observes its joint variables as well as its physical characteristics (i.e $\langle q_i, l_i \rangle$, where q_i is the agent's generalised coordinate, usually referring to the angular displacement of the corresponding joint, and l_i is the length of the corresponding link). It can be seen that q_i defines partially the state of agent a_i. In order to fully define its state, agent a_i requires additional computation for the following variables: $\langle \theta_i, d_i, \vec{g}_i \rangle$. These variables for agent a_i cannot be computed at this stage, since their computation requires information from the other agents that comprise the agent community; and this information is not available at the moment. So, since agent a_i cannot fully solve the problem of defining its state at the moment, it forwards the partially computed solution to agent a_{i+1} residing at the next layer in the hierarchy. Similarly, agent a_{i+1} calculates those variables that can be computed and forwards the partial solution to the agents below in the hierarchy. The sub-process iterates until we reach an agent a_n that succeeds in calculating all those variables that uniquely allow it to fully define its current state.

At the next stage, the recursive process continues by traversing back from agent $a_{i+1} \rightarrow a_i$ providing the agents at the higher layer in the hierarchy with the information that they were missing. Starting from agent a_n (i.e. the end effector) and moving up the hierarchical structure all

```
Fuzzification of the State Space
Fuzzification of the Action Space

State Evaluation
      Agent = Agent(i), Tries to fully evaluate its state.
      If success
            Traverse back the hierarchy of agents and provide them
            with all the available information in order for them to
            successfully define their state

            Go to Action Selection
      Else
            Pass Information gathered to Agent(i+1) and  i = i+1

            Go to State Evaluation

      Loop until all agents have evaluated their states.

Action Selection
      Agent = Agent(i),
      If NO experience exists in Agent
            Stochastically select Action = a(i)
            Stochastically estimate all other Agents actions
      Else if experience exist
            If Agent wants to explore
                  Select Action = not necessary the best action
                  Stochastically estimate all other Agents actions
            If Agent do not want to explore
                  Select Action = action that will generate the greater reward
                  Stochastically estimate all other Agents actions
            End
      End
      Agent = Agent(i+1)
      Loop until all agents have selected Action

Joint Action Execution
Reward Assigned to all Agents
Go Back to State Evaluation
```

Figure 3. Basic algorithmic structure of the proposed multi-agent control framework.

the way to the root agent, one agent after the other gets enabled to calculate the parameters needed to define its state. So when this phase concludes, every agent in the system will have fully solved its state definition problem, resulting in a multi-agent system with a fully defined state. The iterative/recursive process is then repeated in order to define the fuzzified state of every agent in the multi-agent environment.

We note here that throughout this 'top-down→bottom-up' process, the system self-defines its state in a decentralised way, autonomously and incrementally, and thus not directly. All agents in the system (including the root agent) communicate their partial (local) knowledge in a structured way (according to the nested-hierarchical topology), without obtaining at any moment a full observation of the whole system state (they do not know where all the other agents are, what joint configuration they have, how many agents are available, etc). The state definition of each agent (including the state of the root agent that shares the same modular structure as all other agents) contains some variables with information characterised as local with respect to the agent (i.e. local joint angular displacement), and certain state variables containing global

information with respect to the full system, but as viewed in the local agent's reference frame (i.e distance of the joint from the end effector etc). The root agent is simply the top agent in this hierarchy, sharing the same architectural structure with all other agents; no matter how many recursive iterations are performed, its state definition will at all times contain only partial information about the full system topology. The iterative top-down/bottom-up process described above is performed exactly for the reason of incrementally generating this partial state information within each agent, so that after all actions are performed and a reward is obtained and recursively communicated to all agents, each agent will be able to advance its learning by building up its (partial) state-to-(local) action mapping.

This choice of a distributed multi-agent framework is made to better satisfy properties such as modularity, scalability and computational efficiency. It is evident that if it were to have a root agent maintaining global (full), instead of local (partial), information of the system in its state representation, this would require an increasing number of state variables when the system complexity (number of DOFs) increases, and would have two major negative consequences: (a) first, it would have a significant negative impact on the modularity and extensibility of the approach, since a new state vector and learning algorithm would have to be reprogrammed in relation to the dimensionality of the system; as opposed to the proposed nested-hierarchical structure, which is inherently modular, with every agent always requiring an exact number of state variables to describe its local view of the environment (internal-state and task-goal) and (b) second, the increased dimension of the state space would also have a significant negative impact on the learning performance, since the computational load increases exponentially with the dimension of the state vector.

In the following section, the fuzzy Q-learning scheme and the action selection mechanism used in this work are described in detail.

4. Continuous reinforcement learning

This section describes in detail the learning process that is employed in this work. RL methods have been previously applied in significant number of cases, mostly in the field of mobile robotics (Iida, Sugisaka, & Shibata, 2003; Kondo & Ito, 2004; Shibata & Okabe, 1995, 1994; Shibata, Sugisaka, & Ito, 2001; Takahashi et al., 2001). In our work, an RL method is employed in a quite different domain, addressing skill learning and behaviour-based multi-agent control of (dexterous) robotic manipulation. Back in 1992, a multi-agent architecture for controlling a multi-fingered hand was presented, but without incorporating any learning method for skill acquisition at an agent level (Matsui, Omata, & Kaniyoshi, 1992). In Shibata and Ito (2002, 2003), some cases are presented where RL methods are employed in dexterous manipulations, but on a single-agent system architecture. This paper employs an RL method in the frames of the proposed nested-hierarchical multi-agent architecture, aiming to create a developmental process that will enable the robot control system to acquire, though self-experimentation, skills and knowledge on how to perform agile manipulation.

More formally, let us assume a collection of n (homogeneous) agents, each agent a_i ($i \in [1, \ldots, n]$) having a finite set A_i of individual actions $\alpha_j^i \in A_i$ available to it ($j = 1, \ldots, \text{size}(A_i)$). Agents repeatedly operate within the framework of the environment posed, in which they each independently select an individual action to perform. In Kaelbling et al. (1996), RL is defined as the problem faced by an agent that must learn a behaviour through trial-and-error interactions with a dynamic environment. In terms of mathematical description, RL has been formalised as a Markov decision process (MDP). An MDP has four components: states, actions, transitions and reward distributions. More precisely, an MDP is a four-tuple

(S, A, T, r), where S denotes a finite set of states, A denotes the action space, T is a probabilistic transition function $T : S \times A \times S \rightarrow [0, 1]$ that denotes the probability for transition from a state s to a new state s' when a certain action α is applied, and $r : S \times A \rightarrow \Re$ is a reward function that denotes the reward for applying a certain action α to a certain state s.

At this stage, we need to provide a formal definition for the state of our system. Given the agent architecture that has been formulated so far, the state of every individual agent a_i can be expressed as $\langle q_i, \theta_i, d_i, \vec{g}_i \rangle$, and the whole state of the entire multi-agent system can be defined as $s_t = \{\langle q_i, \theta_i, d_i, \vec{g}_i \rangle, \ \ i = 1 \ldots n\}$, at a specific time instance t. For a four-DOF manipulator, the corresponding state definition of each individual agent and subsequently the state definition of the entire system can then be written as

$$S_t = \{\langle q_1, \theta_1, d_1, \vec{g}_1 \rangle; \langle q_2, \theta_2, d_2, \vec{g}_2 \rangle; \langle q_3, \theta_3, d_3, \vec{g}_3 \rangle; \langle q_4, \theta_4, d_4, \vec{g}_4 \rangle\}.$$

Before proceeding to the detailed description of the RL and action selection mechanisms employed in this work, let us adopt some standard game theory terminology in order to facilitate the discussion below (Myerson, 1997). A randomised policy for an agent a_i is a distribution $\pi \in \Delta(A_i)$ (where $\Delta(A_i)$ is a set of distributions over the agent's action set A_i). Intuitively, $\pi(\alpha^i)$ denotes the probability of agent a_i selecting an individual action α^i. A policy π is deterministic if $\pi(\alpha^i_j) = 1$ for some $\alpha^i_j \in A_i$. A collection of policies for each agent a_i is called policy profile, $\Pi = \{\pi_i : i \in [1, \ldots, n]\}$, where n is the number of agents. The expected value of acting according to a fixed profile can easily be determined. If each $\pi \in \Pi$ is deterministic, we can think of Π as a joint action. A reduced profile for agent a_i, is a policy profile for all agents but a_i (denoted Π_{-i}). Given a profile Π_{-i}, a policy π_i is a best response for agent a_i if the expected value of the policy profile $\Pi_{-i} \cup \{\pi_i\}$ is maximal for agent i; that is, agent a_i could not do better using any other policy $\pi_i^{`}$. In the following section we refer to the requirement of continuous state-space and address the important issue of the infinite number of states (Sutton & Barto, 1998), which has to be resolved as discussed hereafter.

4.1 Fuzzy Q-learning

Q-learning algorithm was developed by Watkins (1989) and its goal is to find an optimal policy by maximising rewards received over time. In discrete state–action spaces, Q-learning defines a $Q(s_t, \alpha_t)$ value for each pair of state s_t and action α_t at time instant t.

$$Q(s_t, \alpha_t) \leftarrow Q(s_t, \alpha_t) + \lambda(r_t + \gamma \max_{\alpha} Q(s_{t+1}, \alpha) - Q(s_t, \alpha_t)), \tag{1}$$

where γ is the discount factor ($0 \leq \gamma \leq 1$), λ is the learning rate and r_t is the reward received at time t.

Due to the continuous character of the application considered in this paper, we have chosen to employ a variant of what is known as *fuzzy Q-learning* (Glorennec & Jouffe, 1997), in order to approximate the Q-table in a continuous state–action space. Many similar generalisation techniques have been used to approximate the Q-table, such as neural networks (Rummery, 1994), CMAC (Sutton, 1996), locally weighted regression (Smart, 2002) and fuzzy logic (Grefenstette & Schultz, 1994). In our approach, a typical fuzzy Q-learning mechanism is employed, which has been modified at the action selection process in order to enable learning of 'Joint Actions' (described in the following section) for the agents comprising the system. The mechanism that we employ is a fuzzy rule base, which can be seen as a function that maps the state variables vector to a scalar output. What is then required is a set of fuzzy rules.

The following expression depicts such a rule in the fuzzy rule base mapping:

$$Rule - i : \text{IF}\left(s_1 \in F_1^i\right) \text{ AND } \left(s_2 \in F_2^i\right) \text{ AND } \ldots \left(s_n \in F_n^i\right) \text{ THEN } \left(\text{output} = \phi_t^i\right),$$

where $\{s_1, \ldots, s_n\}$ are the n variables of the state vector, $\{F_1^i, \ldots, F_n^i\}$ are the respective fuzzy membership functions used in the i^{th} rule and ϕ_t^i is the output $\phi-$value produced by Rule-i at time instant t. Following Tagaki–Sugeno fuzzy inference (Takagi & Sugeno, 1985), a Q-value is calculated as follows:

$$Q(s_t, \alpha_t) = \left\{ \frac{\sum_{i=1}^{K} O^i(s_t) \cdot \phi_t^i}{\sum_{i=1}^{K} O^i(s_t)} \right\}, \tag{2}$$

where K is the number of rules and O^i is the rule activation degree for rule i, in state s_t.

In order to facilitate understanding of how the values for O^i are obtained, let us consider an example where a state vector s_t with three state variables (s_1, s_2, s_3) is considered, each variable having two sigmoid membership functions (with the two values being H for high and L for low). In this case of three state variables with two localised signal values for each variable, eight rules need to be included in the fuzzy rule-base. All rules, from 1 to 8, will provide, respectively, eight rule activation weights O^1 to O^8. These weights are then estimated based on the following set of probabilities: $\text{Prob}_H^{s_1}, \text{Prob}_L^{s_1}, \text{Prob}_H^{s_2}, \text{Prob}_L^{s_2}, \text{Prob}_H^{s_3}, \text{Prob}_L^{s_3}$.

For each state variable s_i, the probability $\text{Prob}_L^{s_i}$ can be defined as

$$\text{Prob}_L^{s_i} = \frac{1}{1 + \rho \cdot \text{e}^{-\sigma \cdot s_i}}, \tag{3}$$

where ρ and σ are constants selected manually and $\text{Prob}_H^{s_i} = 1 - \text{Prob}_L^{s_i}$.

The value of $O^i(s_t)$ can now be estimated as

$$O^i(s_t) = \prod_{j=1}^{n} \text{Prob}_{F_j^i}^{s_j}, \tag{4}$$

where n is the number of state variables contained in state vector s_t and i depicts the rule number. So, in our example where $n = 3$ state variables and $K = 8$ rules, we have

$$\begin{pmatrix} O^1(s_t) = \text{Prob}_H^{s_1} \cdot \text{Prob}_H^{s_2} \cdot \text{Prob}_H^{s_3} \\ O^2(s_t) = \text{Prob}_H^{s_1} \cdot \text{Prob}_H^{s_2} \cdot \text{Prob}_L^{s_3} \\ \vdots \\ O^8(s_t) = \text{Prob}_L^{s_1} \cdot \text{Prob}_L^{s_2} \cdot \text{Prob}_L^{s_3} \end{pmatrix}.$$

What is then needed is to find the changes in ϕ values in order to update them. This is achieved through finding, first, the changes δ of Q-values in Equation (1), and then calculating them based on Equation (2). From Equation (1) we have

$$\delta = \Delta Q = \lambda \left(r_t + \gamma \max_{\alpha} Q(s_{t+1}, \alpha) - Q(s_t, \alpha_t) \right). \tag{5}$$

Next, the changes of ϕ values can be found by computing the gradient in Equation (2):

$$\Delta\phi_t^i = \Delta Q \cdot \frac{\partial Q(s_t, \alpha_t)}{\partial \phi_t^i} = \delta \cdot \frac{O^i(s_t)}{\sum_{i=1}^{K} O^i(s_t)}. \tag{6}$$

The overall algorithmic structure of this fuzzy Q-learning mechanism is summarised below in Algorithm 1 (in this algorithm, JASM stands for the joint action selection mechanism, which is described in detail, along with the reward function, in the following section).

Algorithm 1 Fuzzy Q-learning algorithm

Require: Initialise ϕ^i values for all $i = 1 \ldots K$ rules
  while: *Epoch* $\neq$ *Final Epoch* **do**
    Current state $s_t = s_0$
    Current joint action $\alpha_t = \alpha_0$
    while: $t \neq$ *Final Trial step* t **do**
      Execute joint action α_t
      Get next state s_{t+1}
      Get reward r_{t+1}
      Select joint action α_{t+1} by JASM (section 4.2)
      Calculate $Q(s_t, \alpha_t)$ by using Eq. (2)
      Calculate ΔQ by using Eq. (5)
      Calculate $\Delta\phi_t^i$ by using Eq. (6)
      Update ϕ_t^i by using $\Delta\phi_t^i$
      $s_t = s_{t+1}$
      $\alpha_t = \alpha_{t+1}$
  end while
end while

4.2 Action selection mechanism and reward function

Let us first formulate the general principles of the JASM employed in this work, before proceeding to the detailed description of the algorithms and functions that have been implemented. Generally speaking, in a developmental RL process, all agents wish to select actions that progressively maximise the reward received. Each agent contributes its own action component to the joint action that is eventually applied to the environment and determines the transition. The goal is to find a policy that maximises the expected reward (Bertsekas & Tsitsiklis, 1996). It must be pointed out here, though, that every agent in the proposed system acts initially without having any prior knowledge (i.e. there is no previously defined state to joint action mapping), thus acting in a sense stochastically. Each agent a_i decides to perform a random action α_i and at the same time computes only an estimate of what the other (visible) agents in the system might choose as their potential actions. That is, each agent, independently of the rest, selects an action while at the same time estimating how the other agents are likely to act; each agent thus learns joint actions. In the nested-hierarchical topology employed in this work, this process is recursive downwards starting from the root agent and travelling consecutively towards the levels below in the hierarchy. So, each agent a_i selects action α_i and estimates what the actions of the rest of the agents (which are visible below in the hierarchy) will be (for instance, $\alpha'_{i+1}, \alpha'_{i+2}, \ldots$). After the completion of this recursive process, a specific joint action is formulated which the multi-agent system then executes. The system subsequently provides reward or punishment to the agents for their contribution to the joint effort that they have performed.

Let us now proceed to the detailed description of the JASM, as it was used in this work. In the implementation of the multi-agent system proposed in this paper, a specific discretisation of the action-space for each agent was employed. In particular, each agent is able (independently from the others) to select its action among three discrete types of actions: *increase*, *decrease* or *maintain* (with respect to its local joint displacement). However, if there exist multiple optimal joint actions (as in the case of redundant and dexterous manipulation tasks explored in this work), the complexity of the action selection problem increases significantly. If the joint actions are chosen randomly, or in some way reflecting personal biases, then there is a risk of selecting a suboptimal or uncoordinated joint action. This general problem of equilibrium selection (or joint action selection) can be addressed in several ways. One way is the communication among the agents (Shoham & Tennenholtz, 1992); another is to introduce conventions or rules that restrict behaviours and so to ensure coordination. In this paper, we apply a mechanism that results in a coordination among the agents' actions through repeated execution of the specific task by the same agents. In this action selection mechanism, each agent a_i keeps a count of the number of times a specific action has been performed in the past by the same agent (as well as by its collaborative agents). Although simple, this concept is sometimes quite effective and is known as *fictitious play* (Brown, 1951; Fudenberg & Kreps, 1992). More precisely, each agent a_i keeps a count $C^i(\alpha_k^j)$, for every agent a_j that is visible by a_i, indicating the number of times agent a_j has selected action $\alpha_k^j \in A_j$ in the past. When a task is assigned to our multi-agent system, each agent a_i treats the relative frequencies of the moves of all other agents a_j as indicative of their current policy. That is, agent a_i assumes that agent a_j performs action $\alpha_k^j \in A_j$ with probability:

$$P^i\left(\alpha_k^j\right) = \frac{C^i\left(\alpha_k^j\right)}{\sum_{b^j \in A_j} C^i(b^j)}. \tag{7}$$

We note, at this point, that most models in game theory assume that each agent can observe the actions executed by its counterparts with certainty. What we actually employ in this work is somehow more general, allowing each agent to obtain an observation that is related stochastically to the actual joint action selected. Action selection is more difficult when agents are not aware of the rewards associated with various joint actions; hence the expected reward associated with individual and joint actions has to be estimated based on previous experience.

The next issue that needs to be addressed at this stage concerns the definition of the function that specifies the action selection policy of each agent, which is particularly important since effective learning requires sufficient exploration. The action selection mechanism employed in this work is a variant of ε-greedy, called ε-decreasing, where the probability of an exploration action decreases as trials progress. This action selection mechanism starts with an exploration probability: $\varepsilon \cdot (T(t) - 1)/(T_{\max} - 1)$. To compute the probability of choosing an action, we employ a Boltzmann distribution as explained below. On each trial, with a probability equal to $1 - \varepsilon \cdot (T(t) - 1)/(T_{\max} - 1)$, each agent a_i chooses the action α^i with the greatest estimated $\pi(\alpha^i)$:

$$\pi(\alpha^i) = \frac{e^{E(\alpha^i)/T}}{\sum\limits_{\alpha_j^i \in A_i} e^{E(\alpha_j^i)/T}}, \tag{8}$$

where $E(\alpha^i)$ denotes the expected value of an action α^i and T is the temperature parameter that is controlled to diminish over time so that the exploitation probability is increased. Temperature determines the likelihood for an agent to explore other actions: when T is high, even when the $E(\alpha^i)$ of an action is high, an agent may still choose an action that appears to be less desirable.

This exploration strategy is especially important in stochastic environments like the one we are examining, where payoffs received for the same action combination may vary. For effective exploration, high temperature is used at the early stages of a task. The temperature is then decreased over time to favour exploitation, as the agent is more likely to have discovered the true values of different actions. The temperature T as a function of iterations is given by $T(t) = 1 + T_{\max} \cdot e^{-st}$, where t denotes here the iteration number, s is the rate of decay and $T_{\max}$ is the initial temperature.

Now, let us elaborate on the definition of the expected value $E(\alpha^i)$. The presence of multiple agents, each one learning simultaneously with others, is a potential impediment to the successful employment of Q-learning (and RL in general) in multi-agent settings like the one considered in this paper. When an agent a_i is learning the value of its actions in the presence of other agents, it is learning in a non-stationary environment. Thus, convergence of the Q-values is not guaranteed. What we need is each agent's policy to settle. This is a key issue and is discussed hereafter. In general, there are two distinct ways in which Q-learning could be applied in a multi-agent system; the *Independent* and the *Joint-Action Learner* algorithm (Claus & Boutilier, 1998; Glohon & Sen, 2004; Lauera & Riedmiller, 2004) In an Independent Learner algorithm, each agent learns its Q-values regardless of what the other agents are performing. This method is appropriate to be employed when an agent has no reason to believe that other agents are acting strategically. Joint action learner algorithm is the one where the agents do not learn Q-values of their individual actions but learn the Q-values of their joint actions. This implies that each agent can observe the actions of other agents. Each agent in such a system maintains beliefs about the policies of other agents. So, an agent a_i assesses the expected value $E(\alpha^i)$ of selecting an individual action α^i at a current state s to be

$$E(\alpha^i) = \sum_{\alpha^{-i} \in A_{-i}} \left\{ Q(s, \alpha^{-i} \cup \{\alpha^i\}) \cdot \prod_{\alpha_k^j \in \alpha^{-i}} \left[P^i(\alpha_k^j) \right] \right\}. \tag{9}$$

In the above equation, A_{-i} denotes the set of all possible joint actions that can be performed by all other agents except agent a_i. In our case, this set refers to the group of agents that are considered 'visible' by agent a_i, which, in the sense of the nested-hierarchical topology proposed in this paper, refers to the agents that are located one level below in the hierarchy. Then, $\alpha^{-i} \in A_{-i}$ denotes one such joint action performed by this group of agents, and $\alpha_k^j \in \alpha^{-i}$ is an individual action performed by a single agent a_j in this group.

The final issue that we need to address at this stage concerns the definition of the *reward function* $R(t)$. This function computes the reward (or penalty) that an agent receives at each time instant t, after selecting a certain action and moving to a new state. In this work, when we refer to the kinematic control of open kinematic chains, the reward function has been formulated as follows:

$$\begin{cases} \text{if } (D_{\text{goal}}(t) \leq D_{\min}) \wedge (\Delta D_{\text{goal}}) \leq 0) \text{ then} \\ \quad R(t) = e^{-c \cdot (D_{\text{goal}}(t))} \\ \text{if } (D_{\text{goal}}(t) > D_{\min}) \text{ then} \\ \quad R(t) = -2 \\ \text{if } (D_{\text{goal}}(t) < D_{\min}) \wedge (\Delta D_{\text{goal}}) > 0) \text{ then} \\ \quad R(t) = -1 \end{cases}, \tag{10}$$

where $D_{\text{goal}}(t)$ denotes the distance from the goal at iteration time t, D_{min} is a threshold distance after which the agents start receiving reward, ΔD_{goal} is the rate of change of distance from the goal and $c \in (0, 1)$ is a constant parameter.

The overall algorithmic structure of the *JASM* employed in this work is summarised below in Algorithm 2.

Algorithm 2 JASM algorithm

Require: Initialise *Current_Agent* to *Root* of the hierarchy
  while *Current_Agent* $\neq$ NULL **do**
    Let a_i = *Current_Agent*
    $\forall$ agent a_j (visible from a_i),
      Calculate $P^i(\alpha_k^j)$ by Eq. (7), $\forall \alpha_k^j \in A_j$
    $\forall$ possible action α^i, Calculate $E(\alpha^i)$ by Eq. (9)
    $\forall$ possible action α^i, Calculate $\pi(\alpha^i)$ by Eq. (8)
    Select the action α^i (ε-decreasing scheme)
    Lock action α^i to the overall Joint Action: α_t
    Current_Agent $\leftarrow$ *Select_next_visible_Agent*
  end while
  Return the Joint Action: α_t for execution
  Receive Reward $R(t)$ by using Eq. (10)

4.3 RL-based robot control architecture

The overall robot control architecture employed in this work has three layers, as depicted in Figure 4. The first one is the error observation layer, the second one corresponds to the learning and action selection mechanisms, and the last one refers the (low-level) servo control. The first layer receives as input the desired goal at the task-level along with the feedback from the last layer (as shown in Figure 4). The error observation layer provides input to the next layer, which is the action selection layer coupled with the RL module. The RL module augments in fact the action selection mechanism, by receiving error observation data and providing appropriate input to the action selection mechanism, in order to guide the generation of appropriate action(s) (without any initial prior knowledge regarding optimal joint-level motion). Subsequently, the action selection mechanism stochastically generates the joint-level motion commands (i.e. increase, decrease or maintain joint displacements) to be sent to the robot servos, while providing information to the RL module regarding the probabilistic distribution assigned to the different action(s). The commanded joint-level motion propagates to the third layer, which corresponds to the robotic mechanism and generates the motor actuation for the selected actions resulting in the actual robot motion at the task level.

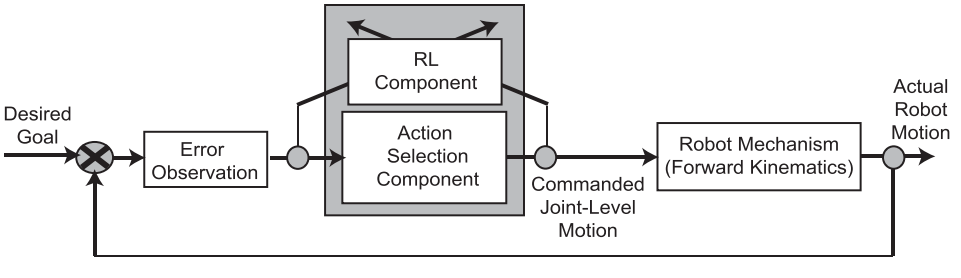

Figure 4. RL-based robot control architecture.

It is important to mention at this point that our proposed control architecture has an interesting relation to work in meta-level control. For instance, in the approach proposed by Schmill et al. (2010), a metacognitive loop (MCL) layered architecture is described, which introduces also a metacognitive layer (similar to our second layer, i.e. the action selection – RL layer) aiming to provide robustness to autonomous systems in the presence of unforeseen perturbations. The MCL basically encodes knowledge on how the system potentially fails in the form of a Bayesian network. By using this network the system may reason abstractly on the action(s) that can be performed to resolve an unexpected situation. In comparison to such approaches, it would be interesting in future work to study and selectively assess the computational cost and the efficiency of each of the three layers proposed in this paper, that is (a) error observation, (b) learning and action selection and (c) servo control.

5. Numerical experiments

To evaluate the performance of the proposed multi-agent learning architecture, we performed a set of numerical experiments grouped into two categories. This section describes and analyses the results obtained, and evaluates specific properties of the system, in order to assess the efficacy of the approach and demonstrate the feasibility of the proposed learning and kinematic control framework in different robot manipulation task scenarios. Initially, we consider a single (redundant) planar kinematic chain, like the one presented in Figure 1, consisting of four links with four independent DOFs, employing the proposed multi-agent topology. Generalisation and robustness properties of the system are assessed in Sections 5.2 and 5.3, respectively, and extensibility of the approach to more complex kinematic topologies, involving constrained manipulation tasks, is demonstrated in Section 5.4. In the second series of numerical experiments, described subsequently in Section 5.5, we consider a simulated planar three-finger grasp of a rectangular object. This second experimental case extends the manipulation skills developed in the previous experiment (for the single kinematic chain), focusing this time on quasi-static control of dexterous multi-finger grasping, where each finger of the simulated dexterous manipulator is again considered to consist of an independent kinematic chain with four links.

5.1 Single kinematic chain

The first task considered in this experimental analysis concerns controlling an open kinematic chain that possesses kinematic redundancies. We consider the case of a planar four DOFs (4R) kinematic chain, with the goal being to control the motion of its end effector towards a pre-specified target location at a Cartesian task space (thus, presenting kinematic redundancies defined by a null-space of rank 2). The six state variables of each agent (in this case, revolute joints) are fuzzified as described in Section 4.1 with several sigmoid functions. In particular, for each one the q_i and θ_i variables we use 20 membership functions, while d_i has 13 membership functions and $\vec{g}i$ has 7 signals for each of the three variables comprising it. The objectives of the multi-agent system in this case are twofold: (i) to respect the limitations imposed by their physical interconnectivity and (ii) to learn how to collaborate and communicate information among agents, in order to reach the goal position, without any a-priori experience or model-based task planning scheme pre-programmed in the system. The first phase of the process is the training of the multi-agent system. The system is trained over a number of epochs (usually, up to 150 epochs), each epoch lasting 500 time units. This means that we allow the group of four agents to come up with a solution to the problem assigned, within a time frame of 500 time units.

In any case (whether a solution is found or not), the knowledge obtained within the specific time interval is stored and the agents are expected to further improve their behaviour during the subsequent epochs.

The behaviour of the agents at the beginning of their collaborative activity must focus on exploration. To satisfy this requirement, the parameters in the equation $T(t) = 1 + T_{\max} * \mathrm{e}^{-st}$ are defined in such a way that for the first 30 time units of each epoch we obtain high exploration, a behaviour achieved with a decay factor $s = 0.35$ (meaning that during this first period the agents might select actions that do not have the highest Q-values), while in the remaining time the agents select those actions with the higher Q-values. The maximum temperature for $t \in [1 \ldots 500]$ is set to $T_{\max} = 100$. The learning rate, for the first epoch, is set at the value of $\lambda = 0.1$, and is decreased after every epoch by a factor of $1/1050$, which means that we promote a slow learning approach. Furthermore, the agents interacting with the environment must receive appropriate rewards. In this experiment, we set $D_{\min} = 3$, meaning that the agents do not receive any positive reward if the end effector is not closer than 3 distance units to the goal position.

Figure 5(a) depicts the learning results obtained (with a decay factor $s = 0.35$), showing the evolution of the mean task error over subsequent epochs (each epoch lasting 500 time units) and its gradual convergence towards zero as the learning epoch progresses (from epoch 1 to epoch 150). It is apparent from these results that initially, during epoch 1, the agents are exploring their workspace, a behaviour that results in a positioning error with respect to the goal position that they are trying to reach. This error is not reduced over the entire time duration of this epoch, leading to a large mean task error. A similar behaviour can be observed for a whole initial period comprising the first 12 epochs, with a slight improvement though observed progressively as the epoch evolves. After epoch 20, however, it becomes apparent that the evolution of the agents' collaborative behaviour manages to reduce the error, and the actions with highest Q-values are now dominating the joint actions that the agents are selecting. In Figure 5(b), we run again the same test with only one modification, the decay factor being now increased to $s = 0.75$. The results show that, already this time by epoch 3, the positioning error is now considerably reduced. In the subsequent epochs we can again observe that the agents' behaviour does settle to a sequence of actions having the highest Q-values, which provides indeed a distributed solution to the problem of kinematic control of the considered redundant open kinematic chain.

Besides the convergence properties of the proposed approach, another important issue concerns evaluating the quality of the solution(s) provided by this mechanism. In other words,

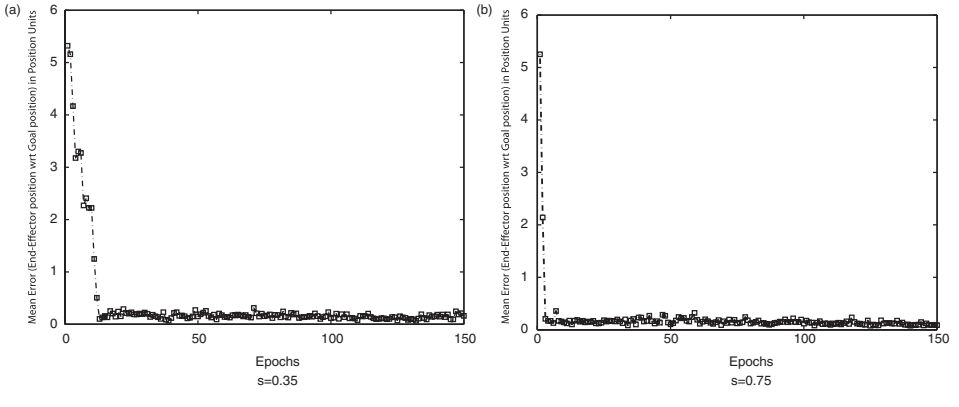

Figure 5. Mean task error convergence over Epochs for two different decay factors: (a) for $s = 0.35$ and (b) for $s = 0.75$.

given the distance-from-target type of reward signal used in this work, the issue here is to evaluate whether the exploration path stochastically adopted by the system will converge to an 'optimal' solution, thus properly resolving this form of kinematic redundancy. Although the proposed multi-agent framework proposed in this paper is not mathematically proven or guaranteed to provide such an optimal solution, it is important to evaluate the solutions that are generated within this framework. An optimal solution in the LS sense corresponds to a joint motion, for every DOF in the kinematic chain, that is minimal in this sense (i.e. 'minimum effort' solution, shortest path to target chosen with a-priori no internal null-space joint motion invoked (Ben-Israel & Greville, 2003; Whitney, 1972). To obtain this theoretical LS-optimal configuration (given the initial joint configuration of the system), one can apply a resolved motion rate control scheme, solving the inverse kinematics problem of the planar (redundant) kinematic chain, based on the computation of the Moore–Penrose pseudoinverse J^+ of the Jacobian matrix J (see Hollerbach & Suh, 1987; Siciliano, Sciavicco, Villani, & Oriolo, 2009) for a detailed analytic solution), which in our case is a 2×4 matrix consisting of four column vectors: $\vec{J}_1, \vec{J}_2, \vec{J}_3, \vec{J}_4$. Each $\vec{J}_i$ vector ($i = 1 \ldots 4$) can be computed as the cross product of the unit vector representing the axis of rotation of the ith joint against the vector expressing the distance between the corresponding ith joint and the end effector. We can then write $\Delta q = J^+ \cdot \Delta p$, where Δq is the increment of the joint angles causing the end effector of the chain to move by Δp, where the pseudoinverse J^+ can be computed (if J is full row rank) as $J^T \cdot (J \cdot J^T)^{-1}$.

Employing this iterative redundancy resolution (analytic) method, we obtain a theoretical optimum set of angular displacements Q_i (for the given initial configuration and final target position: $Q_1 = 0^\circ$, $Q_2 = 24.1^\circ$, $Q_3 = 53.7^\circ$ and $Q_4 = 59.8^\circ$), which are depicted graphically at the left side of Figure 6. An indicative set of corresponding solutions obtained with our proposed

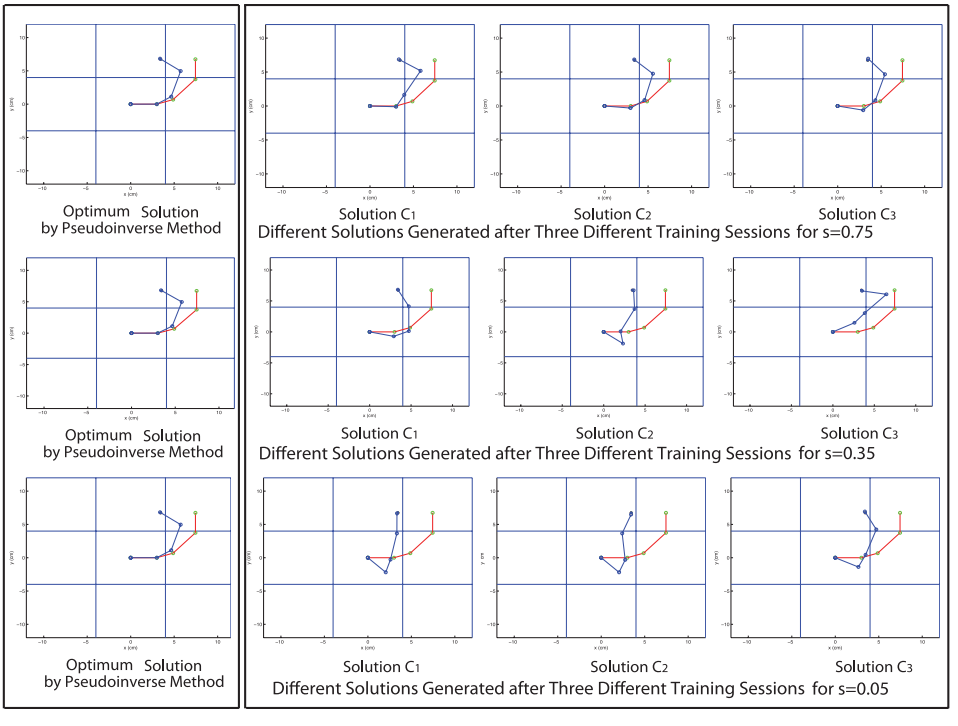

Figure 6. Sample-generated solutions for three different decay factors ($s = 0.75$, 0.35 and 0.05).

multi-agent architecture is also depicted graphically in the same figure, varying according to the decay factor s selected. These results show that for almost all values of s, the multi-agent system generates quite natural solutions, while in certain cases close to the optimal one, as can be seen in the corresponding schematics (stick diagrams). Each solution provided in the corresponding figure has been generated after the completion of three consecutive training sessions (each training session lasting 200 epochs), for three different values of the decay factor s, leading to a total of nine different indicative solutions.

To assess further the 'optimality' of the emerging solutions, we performed a statistical analysis of the results obtained (error with respect to the theoretical LS optimal configuration) for a decay factor s varying in the interval $[0.005, \ldots, 0.9995]$. The mean values and standard deviations (STD) of the error obtained are depicted in Figure 7, at two different instances of the learning process (after 10 and 100 learning epochs). These results demonstrate that, for a wide range of s values, the solutions stochastically generated by the system are near-optimal in the LS sense. Therefore, though it is not argued in any way that the proposed multi-agent learning system is theoretically guaranteed to provide such an optimal solution to this specific problem (nor is it necessarily, in any case, the goal sought by our system), it is shown that the system stochastically converges to a range of solutions that not only effectively resolve the kinematic redundancy but are also experimentally found (depending on the exploration-vs-exploitation strategy employed) to be indeed close to the theoretically optimal solutions in this sense, for a wide range of values of the exploration decay factor.

A similar conclusion can be drawn by observing the results depicted in Figure 8, showing the mean values and STD of the overall 'effort' in the joint space (computed here as the cumulative sum of the absolute joint displacements performed at every time step by all agents in order to reach the goal) after 100 learning epochs, for varying decay factor s. These results compare favourably to the respective cumulative effort computed in the case of a minimum distance (pseudo-inverse) solution, where a value of approximately 4.48 (rads) is obtained for the same initial and target positions. This finding is also validated by observing the paths depicted in Figure 9. This figure shows a sample goal-reaching path generated by the proposed model-free multi-agent approach (after 100 learning epochs, with a decay factor $s = 0.5$), which yields a quite natural result as compared to the minimum distance (theoretically LS-optimal) solution, obtained by applying resolved motion-rate control using an analytic computation of the Jacobian matrix pseudo-inverse. One could then argue that the reinforcement approach adopted in this

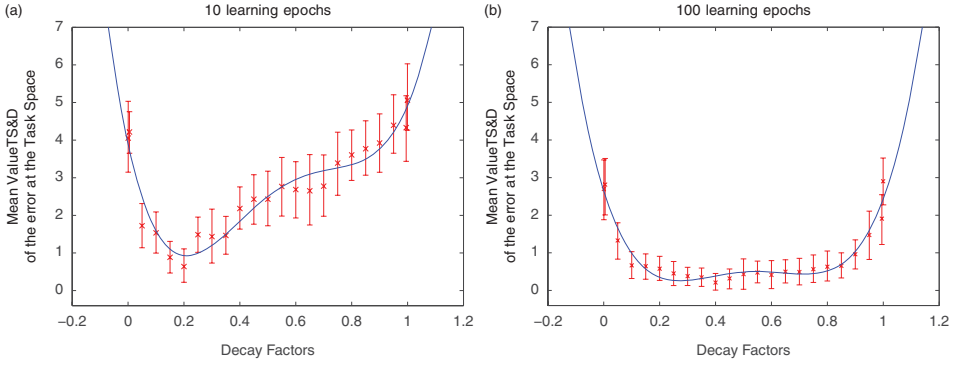

Figure 7. Mean value and STD of the task error (w.r.t to the LS optimal configuration), for a varying decay factor s, after (a) 10 and (b) 100 learning epochs.

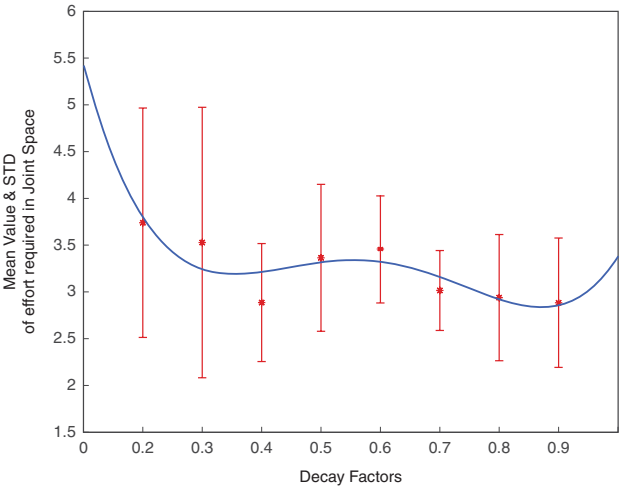

Figure 8. Mean value and STD of the cumulative 'Effort' required in joint space (the effort is here measured as the sum of the absolute joint displacements at every time step t for the entire time period T, $\sum_{t=0}^{T}|\Delta q_i(t)|$, for every agent i comprising the system), for a varying decay factor s, after 100 learning epochs.

paper statistically and progressively rewards actions and converges towards behaviours that lead to faster (and more direct) distance error minimisation, thus creating state-to-action maps that efficiently achieve the desired target and solve the redundancy by rewarding minimum-effort type of motions. This is an important experimental finding that supports the argument about the applicability of such a multi-agent distributed control methodology in a dexterous manipulation context.

Thus, regarding the question on how the multi-agent system can resolve the redundancy with only partial (localised) information about the goal used within each agent, and how it is expected to be able to resolve it in an 'optimal way', experimental findings in this paper support the argument that (1) the state representation, as distributed among agents, suffices for the system to search and find solutions in a distributed manner and (2) the solutions that emerge are indeed expected to be 'close to' the theoretical LS optimal solutions, since the system, during

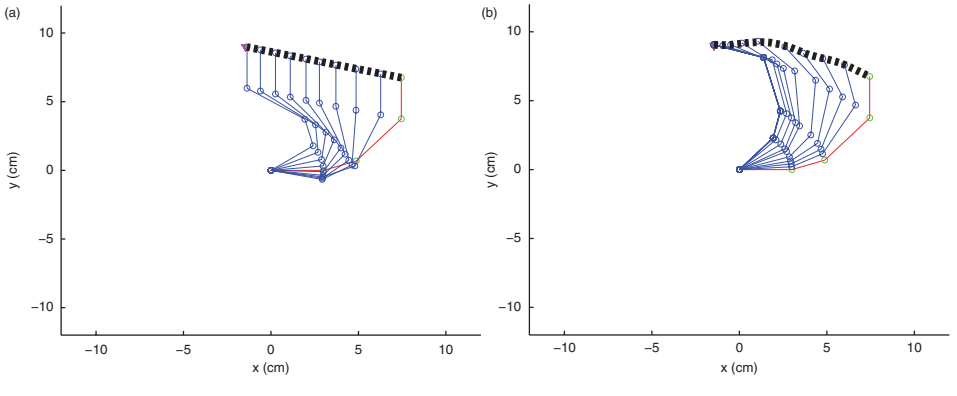

Figure 9. Path generated for goal-reaching task. (a) LS-optimal (pseudo-inverse) minimum distance solution. (b) Multi-agent (model-free) sample solution generated after 100 learning epochs (with $s = 0.5$).

exploration, attempts to find 'fast' (i.e. LS near-optimal) target-converging solutions, which represent minimum cost in the RL reward domain. It would, however, be interesting in future work to explore other reward functions that would probably involve a hierarchical task decomposition strategy (defining lower-priority subtasks), that is, attempting to resolve redundancy while exploiting dexterity in a targeted task-oriented manner. It would then be interesting to evaluate the solutions that emerge and how these compare to theoretical model-based analytic solutions that exploit null-spaces of full model Jacobians. Such an analysis is, however, beyond the scope of this paper, which constitutes a first proof-of-concept of the feasibility of the proposed multi-agent control methodology in a dexterous manipulation developmental learning context.

When the learning process is completed and the agents have acquired knowledge about how to collaboratively reach their goal, the next step is to evaluate the generalisation properties of the system, that is, to see how the agents use the knowledge acquired and how, without any additional training, they can explore and reach new targets different from those trained.

5.2 Multi-resolution goal training and generalisation

In this section, we evaluate the skill generalisation properties of the proposed multi-agent learning system. For this purpose, multiple layers of training data are created, each layer representing a different resolution at the target space. Thus, we aim to explore how well the system manages to extend the knowledge acquired during training, in order to subsequently reach new (untrained) goals. Initially, the (planar, in this case) task space is recursively subdivided into levels of increasing resolution, in a quad-tree structure. Starting from *Level*-1, containing four nodes (quadrants), we progressively move down to *Level*-4 that contains 256 nodes. Each node corresponds to a single target position, for which the multi-agent system receives training. Therefore, for each resolution level, the multi-agent system is trained on a different set of target positions, progressively increasing in size according to the resolution *Level* (in other words, at *Level*-1, the four nodes correspond to four goal positions on which the system is trained, for *Level*-2, 16 target positions are trained, etc).

For the purpose of the experiment presented in this section, we thus investigate a range of target-space resolutions starting from 4 (Level 1) up to 4^4 target training points (Level 4), as shown in Figure 10. For each resolution level, after completion of the training period for the corresponding target positions that constitute the training set at this level, a test-set is generated consisting of 100 new goals randomly distributed in the entire task space (by employing Halton sequences (Niederreiter, 1992) to obtain a uniform distribution of target points in the task space). Our aim is to explore the accuracy by which, at each resolution level, the multi-agent system manages to reach all the new, untrained and randomly generated goals, without performing any additional training.

The experimental process is the following: (i) select a resolution level, (ii) perform off-line training on the target points of the corresponding resolution level (training set), (iii) after the completion of the training phase, issue successive on-line requests to reach all 100 randomly generated goal positions of the test set. Indicative target-reaching results, obtained for two different resolution levels, are depicted in Figure 11. By observing these plots, it is clear that even at the lowest resolution level (comprising only four training points in the entire task space), the proposed multi-agent topology manages to successfully reach almost all 100 randomly generated new goals assigned. It is also clear that the situation improves as the resolution of trained goal positions increases up to the level comprising 256 points.

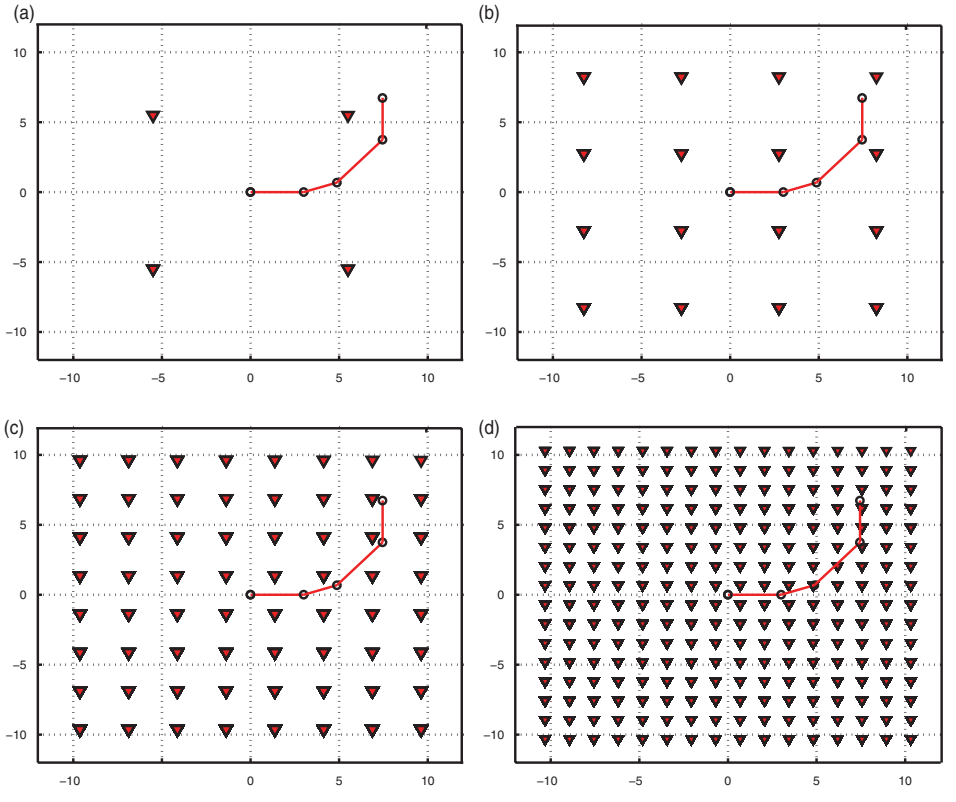

Figure 10. Four resolution levels and corresponding training targets, from (a) Level-1, with 4 trained targets, up to (d) Level-4, with 256 trained targets.

Figure 12 depicts the statistics of the target-reaching generalisation results. The results demonstrate that even at the lowest resolution level (Level-1, comprising only four training points), the mean value of the error is quite low (mean = 0.3814), but with significant STD value (STD = 0.5596), showing again that most of the randomly generated sample target points at the test set are successfully reached, even when using such a small number of training points. The mean value of the error decays smoothly and reaches, at Level-3 with 64 points resolution, a value of mean = 0.2366, while the corresponding STD drops to the value of STD = 0.2036.

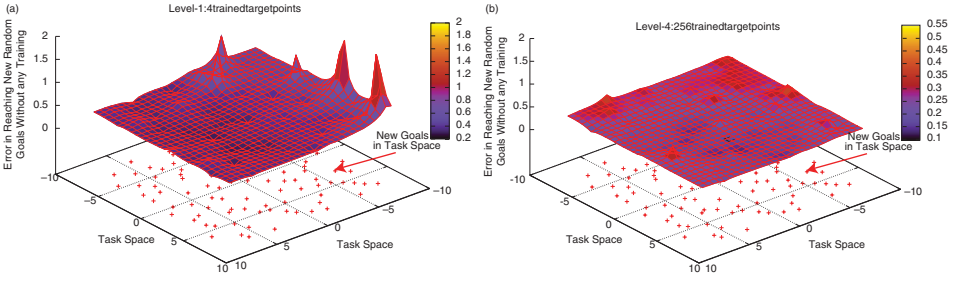

Figure 11. Target-reaching error in the entire task space for two different training resolution levels: (a) Level-1, with 4 trained target positions (b) Level-4, with 256 trained target positions.

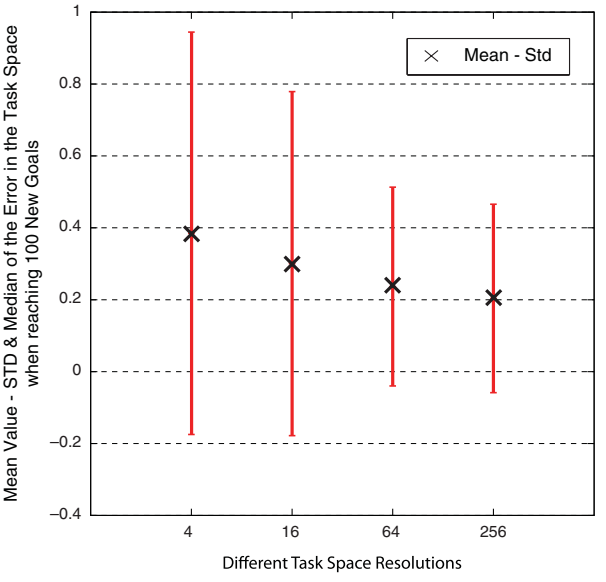

Figure 12. Overall generalisation performance: mean and STD values of the target-reaching error at the entire task space for different training resolution levels.

We also observe that, in this experiment, increasing the resolution from Level-3 to Level-4 (i.e. from 64 to 256 training points) does not significantly improve the accuracy of the system. These findings demonstrate the generalisation properties of the proposed multi-agent architecture.

5.3 Robustness to changes in kinematic topology

A key advantage of a distributed multi-agent control strategy (with respect to a centralised single-agent model-based approach) is related to the presumably better inherent robustness properties. In this paragraph, we evaluate the capacity of the proposed multi-agent architecture to compensate for partial failures that may occur to some of the agents (DOFs) comprising the system and, therefore, to adapt to unpredictable sudden changes in the kinematic topology. In other words, when one or more agents fail during the operation of the system, we want to explore to what extent the rest of the active agents manage to collaborate in order to provide a new feasible solution to the task (without retraining, and of course, without any a-priori knowledge or modelling of the event).

To evaluate such robustness properties of the multi-agent architecture, a series of numerical simulations has been performed. We start by training the multi-agent system to reach a goal position, as described in the previous section. After completion of the training period, the fully operational multi-agent system is now able to find a solution to the problem and the tracking error (end effector w.r.t. goal position) converges to zero, as depicted in Figure 13. Figure 14 shows the actions (joint displacements) of individual agents. The cooperation of these agents' actions successfully completes the task (that is, moves the end effector of the kinematic chain towards the specified target position). The system has the same learning parameters as before, and the number of time units per trial is 1500.

In the sequel, we simulate a number of failure situations in the agents, in order to evaluate the robustness properties of the system. First, we simulate an initial partial failure of the system,

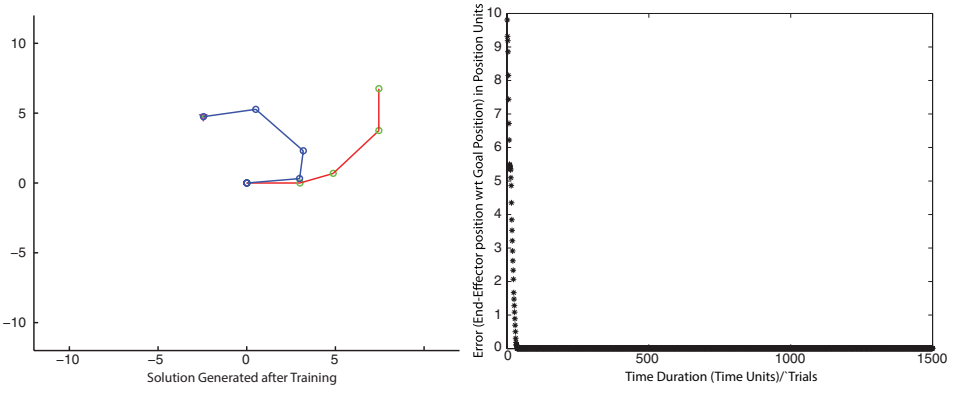

Figure 13. Kinematic solution generated by the multi-agent system and position error over time (for the fully operational system, with all agents active).

which is assumed to be progressively recovered. We consider that, initially (at $T = 0$), three of the agents (agents 1, 2 and 3) are non-operational and only agent 4 is responding. This situation corresponds to a failure of receiving and executing any action by these agents, which means that agents 1 to 3 stay blocked to their initial angular position. Subsequently, at $T_1 = 100$, agent 3 starts responding and later on, at $T_2 = 500$ and $T_3 = 1000$, agents 1 and 2, respectively, are also recovering from their failure. Our goal is to explore how, without any external assistance, the multi-agent system manages to compensate for the imposed failure situation. The results are quite interesting, demonstrating that the multi-agent system manages, at run-time, to find a new solution adapting to the unpredictable changes in the kinematic topology caused by the considered failure situation with some of the agents not responding during operation. Figure 15 depicts the new kinematic solution generated by the multi-agent system, and the evolution of the tracking error. Figure 16 shows the actions of the individual agents, and how they adapt to cope for the disturbance caused by the failure events.

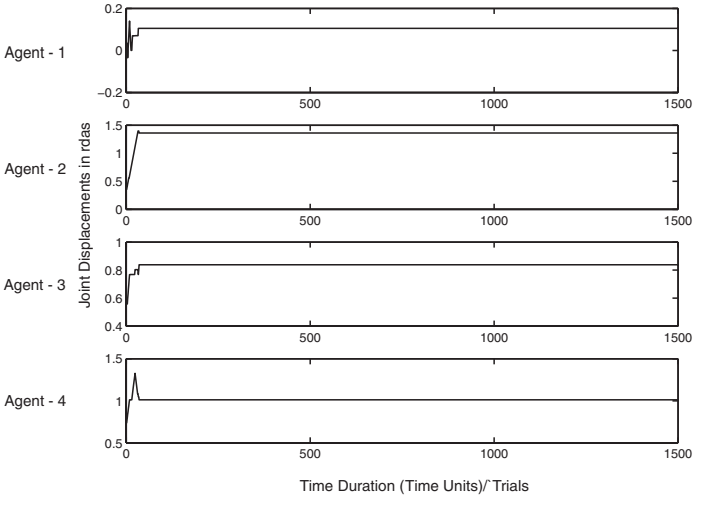

Figure 14. Actions (joint displacements) of the agents. All four agents are active and cooperate to reach the goal position.

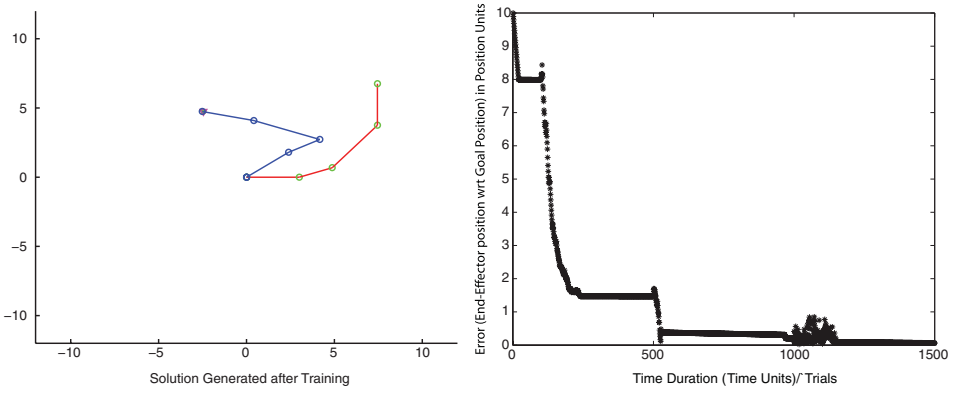

Figure 15. Kinematic solution generated and tracking error after failure and progressive recovery of agents 1 to 3.

Subsequently, we simulate a more complex failure situation. This time, two of the agents (2 and 3) are blocked (to their initial joint position) from the beginning of the task execution period, leaving only two agents (1 and 4) operating normally at the beginning of the task. Then, at $T_1 = 700$, agents 2 and 3 start responding introducing random joint displacements, but only for a small period of 300 time steps. At $T_2 = 1000$, these agents go again out of operation for the rest of the experiment, remaining locked to their new current angular positions at that time. From the plots in Figure 17, we observe that the multi-agent system manages again to generate a new kinematic solution and reach the goal position successfully. Figure 18 shows the actions of the agents (joint displacements), where we see how the remaining active agents adapt their behaviour at run-time, to compensate for the assumed complex 'failure-disturbance-failure' situation.

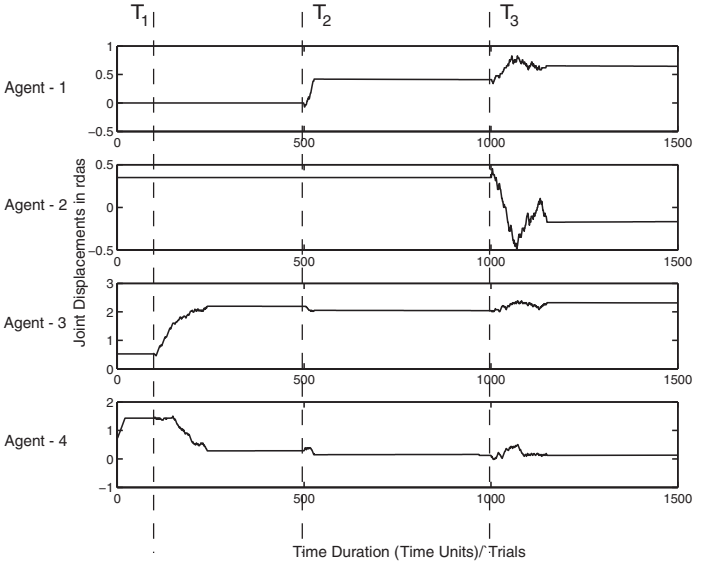

Figure 16. Agents are adapting their actions dynamically to compensate for unpredictable failure-recovery events of agents 1 to 3.

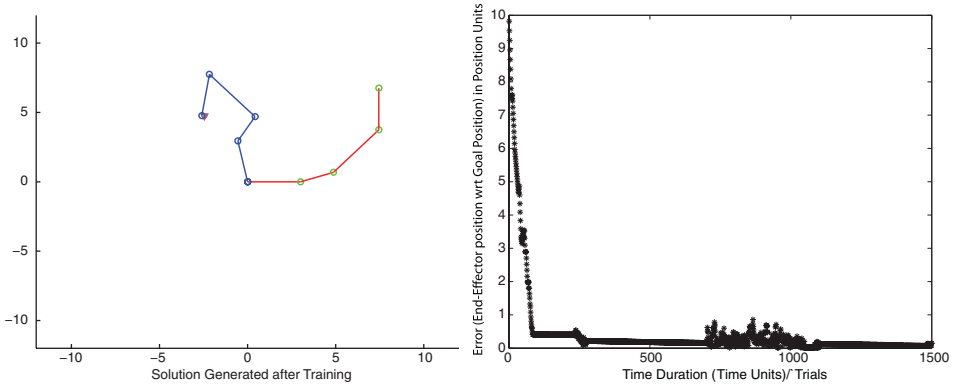

Figure 17. New kinematic solution generated and time evolution of the tracking error, when considering a complex 'fail-disturb-fail' situation for agents 2 and 3.

For the last experiment, we repeat the failure process already described above for agents 2 and 3, but this time these agents are completely blocked from the beginning of the experiment until its completion. Figures 19 and 20 demonstrate, once again, that the remaining active agents (1 and 4) manage to cooperate at run time and reach the goal position, by dynamically generating a new solution and compensating for the unpredictable failure situation assumed to occur for agents 2 and 3 (blocked throughout the task). These results demonstrate the robustness properties exhibited by the proposed multi-agent system and show its capacity to cope for unpredictable and complex failure situations, equivalent to sudden changes in the kinematic topology, which a model-based control strategy would a-priori not be able to handle.

To better highlight the superior robustness properties exhibited by the proposed multi-agent (model-free) system, as compared to a centralised (model-based) control scheme, we assume again the same failure situation as in the last experiment before (joints 2 and 3 completely

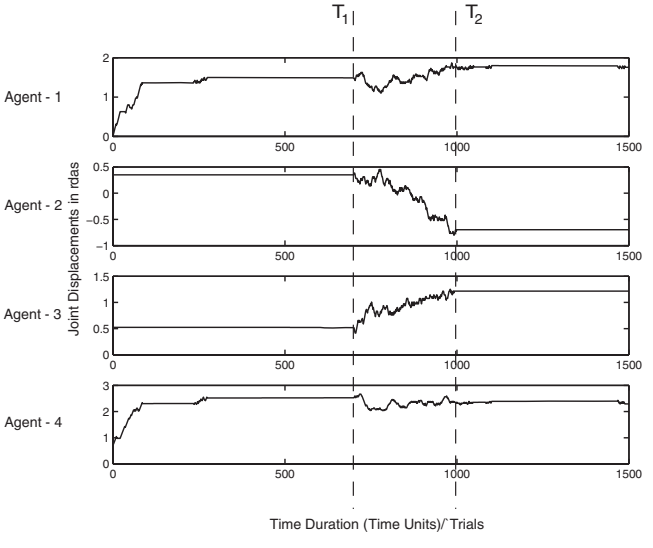

Figure 18. Agents 1 and 4 are adjusting their actions dynamically to compensate for a complex 'fail-disturb-fail' situation of agents 2 and 3.

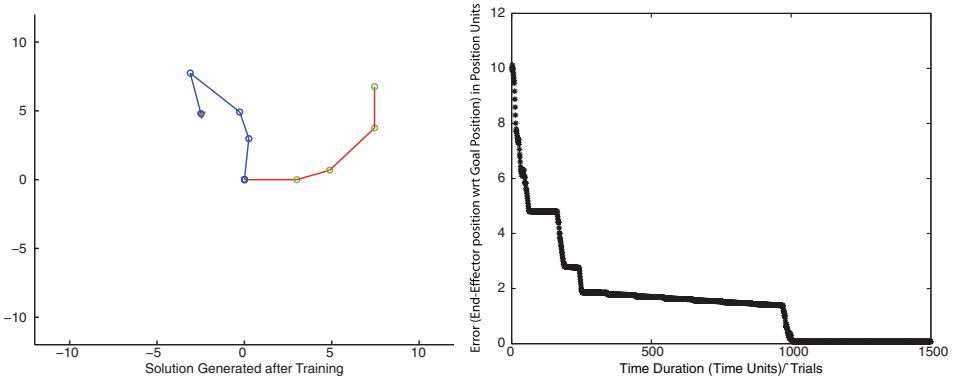

Figure 19. Kinematic solution generated and tracking error over time, when considering a 'complete fail' (fully blocked) situation for agents 2 and 3.

blocked), and perform a similar experiment but this time applying a model-based (resolved motion rate) kinematic control at the joints (using the Jacobian matrix pseudo-inverse as explained in the previous section). The tracking error results obtained are depicted in Figure 21, comparatively for the model-based and the multi-agent approach. The target-tracking response of the kinematic chain is also illustrated in Figure 22. From these plots, it is evident that the kinematic chain under model-based kinematic control is unable to reach the designated target position, generating oscillatory motion patterns between specific configurations. This is obviously due to the fact that the kinematic model is in this case a-priori invalid and a typical centralised model-based system does not have the mechanism (without any remodelling and replanning) to cope with such unpredictable errors, therefore potentially generating motions inconsistent with the target position at the task space. On the contrary, the proposed multi-agent system is still able to find a feasible solution (which seems, in fact, consistent with a minimum-energy configuration), as shown in Figure 22(b). It should be emphasised here that, although we are not claiming in any way that model-free approaches can be as accurate as model-based

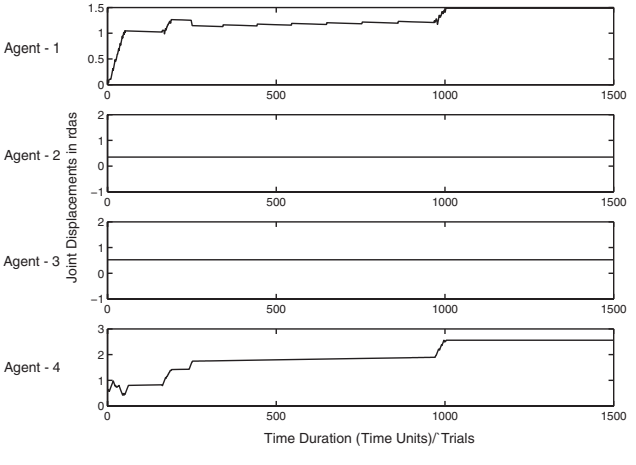

Figure 20. Agents 1 and 4 are adjusting their actions (joint displacements) dynamically to compensate for the unpredicted (complete failure) behaviour of agents 2 and 3.

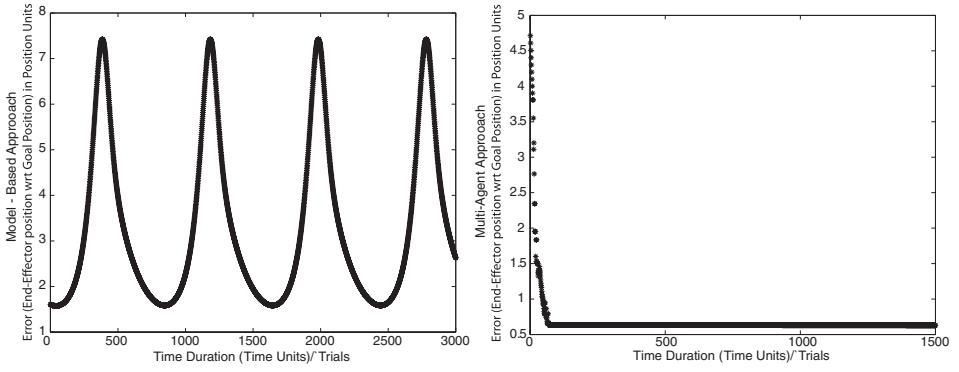

Figure 21. Comparative tracking error results (model-based vs. multi-agent approach) in a failure scenario with agents 2 and 3 fully blocked.

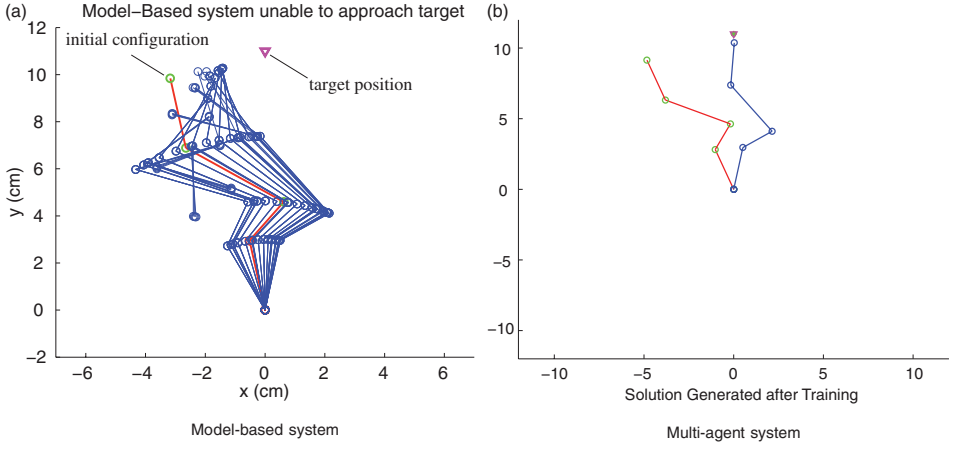

Figure 22. Comparative target-tracking response for the model-based and the multi-agent systems in a failure scenario with agents 2 and 3 fully blocked.

methodologies, the above results demonstrate the performance of the proposed multi-agent approach in terms of its adaptability to changes in kinematic topology and its capacity to cope with unknown and unpredictable failure situations, where model-based approaches would indeed require re-modelling and re-computation of analytic solutions.

5.4 Scalability to hyper-redundant constrained manipulation tasks

To evaluate the inherent scalability properties of the proposed multi-agent framework and its extensibility to more complex kinematic topologies, we assume the following constrained manipulation task. A hyper-redundant (7 DOFs planar) kinematic chain is considered and the corresponding multi-agent system must learn how to reach a target position that is located at the inside end of a narrow corridor. The goal is to generate a collision-free motion for the kinematic chain, enabling the end effector to reach the target position without using any a-priori model of the environment. In such an experimental scenario, a typical model-based (single-agent) kinematic approach would have to introduce additional distance optimisation criteria, which

would make the analytic redundancy resolution problem more complex and more sensitive to modelling errors. The proposed distributed multi-agent architecture is naturally extended to cope with such a topology, considering only an additional term on the reward function that signals collision events during training.

The reward $R(t)$ that an agent receives at each time instant t, after selecting certain action and moving to a new state, is now formulated as follows:

$$\begin{cases} \text{if}(D_{\text{goal}}(t) > D_{\min}) \text{ then } R(t) = -2 \\ \text{else if } (D_{\text{goal}}(t) \leq D_{\min}) \wedge (\triangle D_{\text{goal}} > 0) \text{ then } R(t) = -1 \\ \text{else if}(D_{\text{goal}}(t) \leq D_{\min}) \wedge (\triangle D_{\text{goal}} \leq 0) \wedge (\forall a_i, C_{a_i} = 0) \text{ then} \\ R(t) = \mathrm{e}^{-c_1 \cdot (D_{\text{goal}}(t))} \\ \text{else (i.e. if } (\exists a_i, C_{a_i} = 1)) \\ R(t) = -\mathrm{e}^{-c2 \cdot (\min_i D^{a_i}_{\text{Corridor}}(t))} \end{cases}, \tag{11}$$

where $D_{\text{goal}}(t)$ is again the distance from goal at time t, $D_{\min}$ is a threshold distance below which the agents may start receiving positive reward, ΔD_{goal} is the rate of change of distance from the goal and c_1, c_2 are constants $\in (0, 1)$. C_{a_i} is a Boolean flag detecting when agent a_i is close to collide with the obstacle (corridor) and $D^{a_i}_{\text{Corridor}}$ is the distance from the corridor that is locally perceived by agent a_i. In this reward function, if more that one agents detect to be close to collision, then in order to increase the global negative reward to the multi-agent system, we compute $R(t)$ based on the $\min_i D^{a_i}_{\text{Corridor}}$ among all agents a_i locally perceiving collision.

The results obtained using the proposed multi-agent system are presented in Figure 23. These results show that the multi-agent system manages to evolve behaviours that effectively guide this hyper-redundant kinematic chain to enter the opening of the narrow corridor and to successfully reach the target position, generating a natural collision free path.

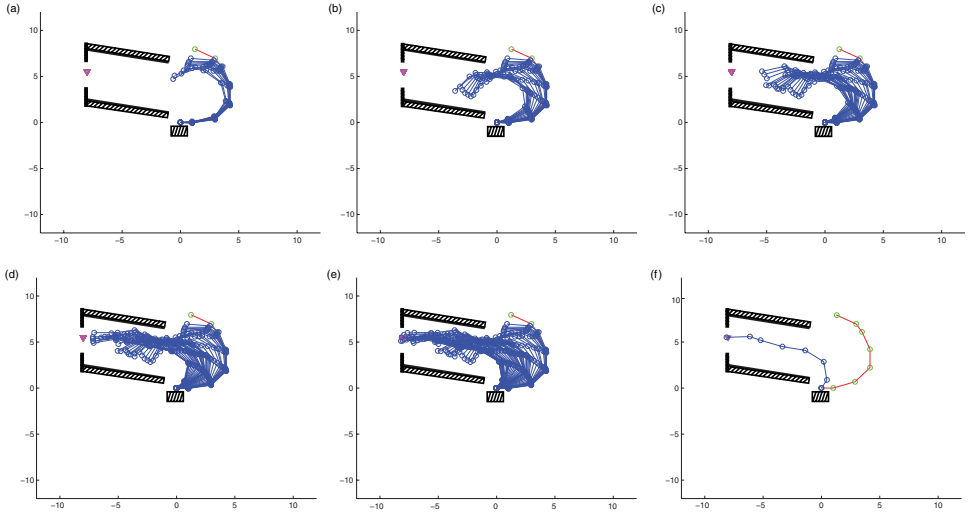

Figure 23. Extensibility of the proposed multi-agent framework to a hyper-redundant kinematic chain performing a constrained motion task, involving collision-free target reaching inside a narrow corridor.

5.5 *Multi-finger grasp*

The second series of numerical experiments presented in this section builds upon the results obtained in the previous sections regarding open kinematic chains and further extends the multi-agent topology on a more complex task, by now considering three independent kinematic chains cooperating together to achieve a stable three-finger grasp. The key assumption in this second series of experiments concerns 'quasi-static' manipulation, which means that we are still considering kinematic (in this case, the dual analogue, static) control of the system, neglecting from the control problem the linkage and object dynamics involved. Our main objective here is to explore the capacity of the system to learn how to generate a static force distribution, thus neglecting the object dynamics, which is equivalent to assuming an 'infinite mass' for the grasped object. This assumption leads to a simplified task-case scenario, where our aim is to focus only on the grasp distribution problem alone. Therefore, this second experiment aims to validate whether by using the proposed control structure, the system can autonomously acquire skills that consist in performing coordinated actions (of the individual kinematic linkages – acting fingers) to achieve a desired resultant grasp force/moment in a quasi-static sense. This experiment also constitutes a good example to demonstrate the scaling properties of the nested-hierarchical structure of the proposed multi-agent control architecture, where a new top-level 'grasp-agent' is now added in the hierarchy to encapsulate the individual 'fingers-agents' and their underlying topology.

The arrangement of the three-finger dexterous manipulator, attempting to perform a quasi-static grasp as considered in this experiment, is shown in Figure 24(a), where once again we assume that the motion of the manipulator is restricted on a plane. The multi-agent representation of this three-finger manipulator, as obtained using the proposed architecture, is shown in Figure 24(b). This figure visualises how the kinematic chains of the considered multi-finger dexterous manipulator model, shown in Figure 24(a), are mapped on agents using the proposed multi-agent topology. This multi-agent representation, as depicted in Figure 24(b), clearly illustrates the nested hierarchical properties of the proposed architecture.

Based on the above representation, the numerical results presented in the sequel assess the learning capacity of a multi-agent system comprising 13 hierarchically nested agents, with the goal being to perform a grasp with no slip on any of the fingertips and with resultant force and torque equal to zero. The initial joint configuration assumed for the kinematic chains is the one

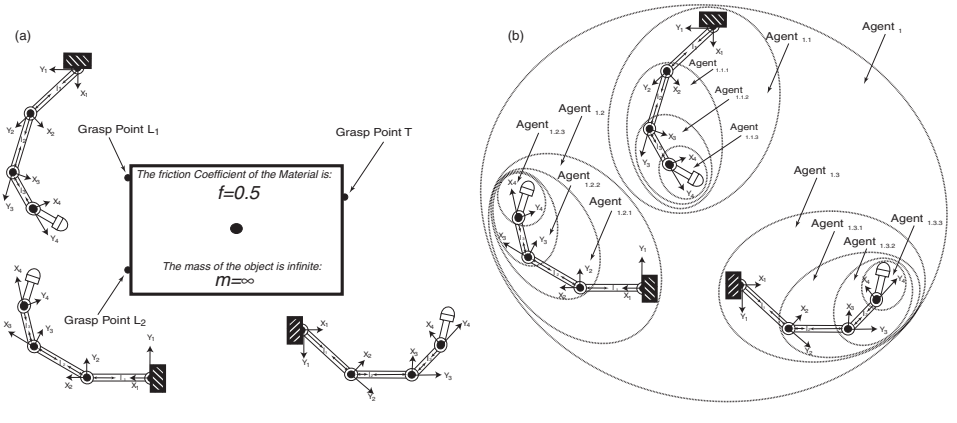

Figure 24. Considered set-up for the second series of numerical experiments: (a) Three-finger quasi-static grasp (b) Nested-hierarchical multi-agent representation of the three-finger manipulator.

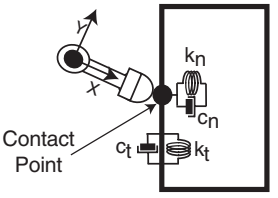

Figure 25. Simulated point contact model, with tangential and normal spring-damping coefficients.

shown in Figure 24(a), where the Coulomb friction coefficient, defining the friction cone limits (at each contact point) within which every multi-agent finger must learn to operate, is also depicted. We note that all contact points are simulated using a simple linear elasticity model, with specific spring and damping coefficients, as shown in Figure 25, where k_t is the tangential and k_n is the normal spring coefficient, and c_n, c_t are the normal and tangential damping coefficients, respectively. The rest of the simulation parameters are the same as those described in the previous sections (for the open kinematic chains). The fuzzification parameters as well as the learning parameters have been kept the same as before. The system is trained over 200 epochs, each epoch lasting 500 time units. Within the corresponding time interval, the agents will have to learn simultaneously how to solve two problems: (a) first, to reach the grasp points (defined as their goal positions, according to the training results obtained during the previous experiments) and (b) second, to successfully coordinate their actions in order to apply appropriate contact forces at the corresponding grasp points, always maintained within the indicated friction cone in order to avoid slipping effects, and summing up to a desired net force and moment (in the case of the considered experiment, summing up to zero).

According to these requirements, the goal of the multi-agent system for this experiment has been augmented, as compared to the previous experiments. Since the system is now attempting to learn, at a single step, two different goals, the reward function has to be modified appropriately in order to correctly guide the RL mechanism. The reward function employed in the previous sections, which supports the task of reaching a goal position, is still valid but is now augmented by a reward term R_i corresponding to the ith task constraint, namely R_τ being the reward for satisfying the net moment constraint, R_f being the reward for satisfying the net force constraint and R_s being the reward for satisfying the friction cone (i.e. no slip) constraint. Therefore, the reward function is now formulated as follows:

$$\left\{\begin{array}{l} \text{if } (D_{\text{goal}}(t) \le D_{\min}) \wedge (\triangle D_{\text{goal}}) \le 0) \text{ then } R(t) = \mathrm{e}^{-c\cdot(D_{\text{goal}}(t))} + \sum_i R_i \\ \text{else if } (D_{\text{goal}}(t) > D_{\min}) \text{ then } R(t) = -2 + \sum_i R_i \\ \text{else if } (D_{\text{goal}}(t) < D_{\min}) \wedge (\triangle D_{\text{goal}}) > 0) \text{ then } R(t) = -1 + \sum_i R_i \end{array}\right\}, \tag{12}$$

where the reward terms $R_i\,(i \in \{f, \tau, s\})$ are defined as follows:

$$R_f = \mathrm{e}^{-c^*\text{Net}_{\text{force}}(t)}, \text{ for the force constraint}$$
$$R_\tau = \mathrm{e}^{-c^*\text{Net}_{\text{torque}}(t)}, \text{ for the torque constraint and}$$
$$R_s = \mathrm{e}^{-c^*\text{Friction}_{\text{Cone}}(t)}, \text{ for the no-slip constraint, where } t \text{ indicates the trial.}$$

The numerical experiments described in the sequel are organised in two sets of trials, each one with a different value for the decay factor s ($s = 0.05$ and $s = 0.75$). The obtained results are

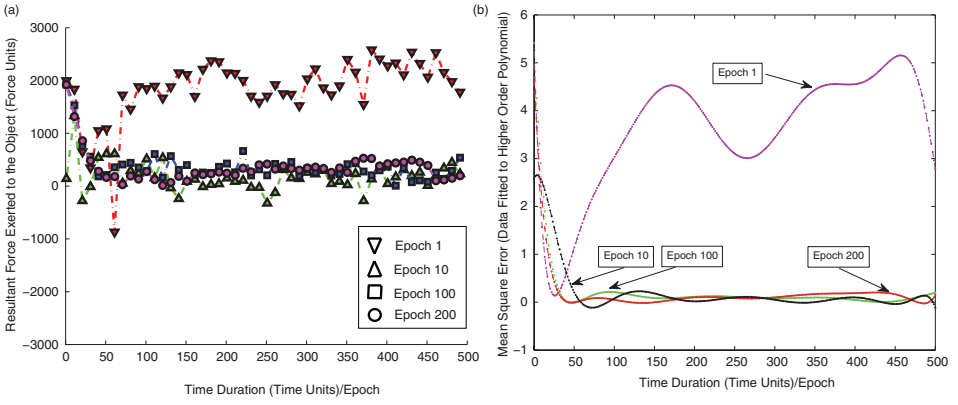

Figure 26. Evolution of (a) net force and (b) mean square force error over time, for four different epochs (with decay factor $s = 0.05$).

presented in Figures 26 and 27. In these figures, the results for four different epochs (1, 10, 100 and 200) are shown. Each pair of plots shows the evolution over time of the applied net force (with the desired net force equal to zero) and of the mean force error. These results clearly illustrate the learning performance of the multi-agent grasping system, and demonstrate the capacity of the proposed architecture to progressively achieve simultaneous collaborative adaptation of the dexterous manipulator's fingertips actions. Regarding the friction coefficient (which, we recall, has been set in this case at $\mu = 0.5$), this defines a friction cone limit within which all fingers must operate (in the sense that all fingertip contact forces must lie inside this limit). Figure 28 illustrates how the contact forces applied by each one of the three fingers (kinematic chains) progressively adapt (for epochs 1, 10, 100 and 200) to fit within these limits of the considered friction cone constraints. Finally, Figure 29 depicts an indicative set of solutions that the system generates for different simulation parameters. The proposed multi-agent system proceeds using the knowledge acquired, without any additional training, by exploring and reaching new contact points that are related to those trained, in a manner similar to that already described in the previous sections for the case of a single kinematic chain.

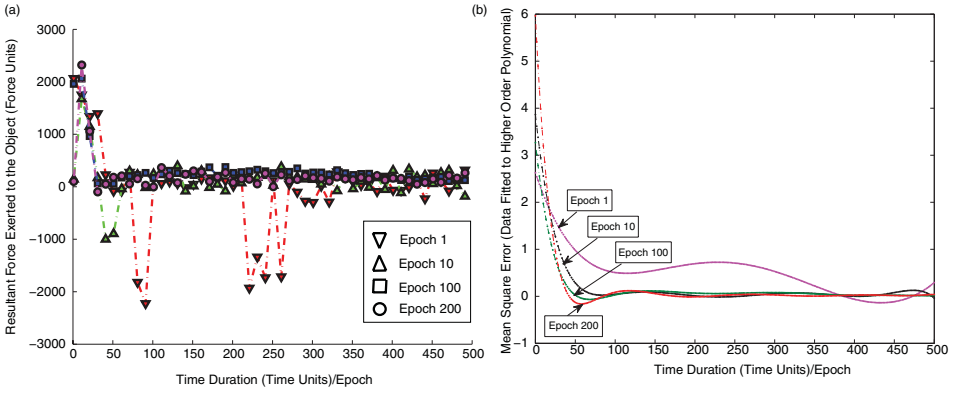

Figure 27. Evolution of (a) net force and (b) mean square force error over time, for four different epochs (with decay factor $s = 0.75$).

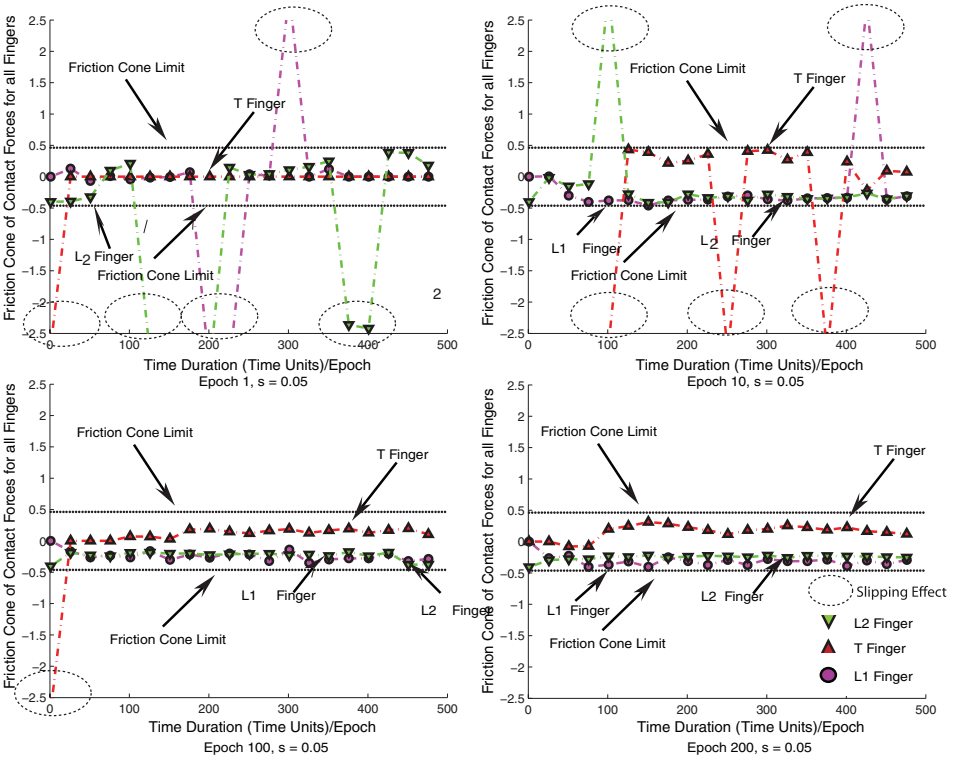

Figure 28. Adaptation of fingertip force within the friction cone limit ($s = 0.05$).

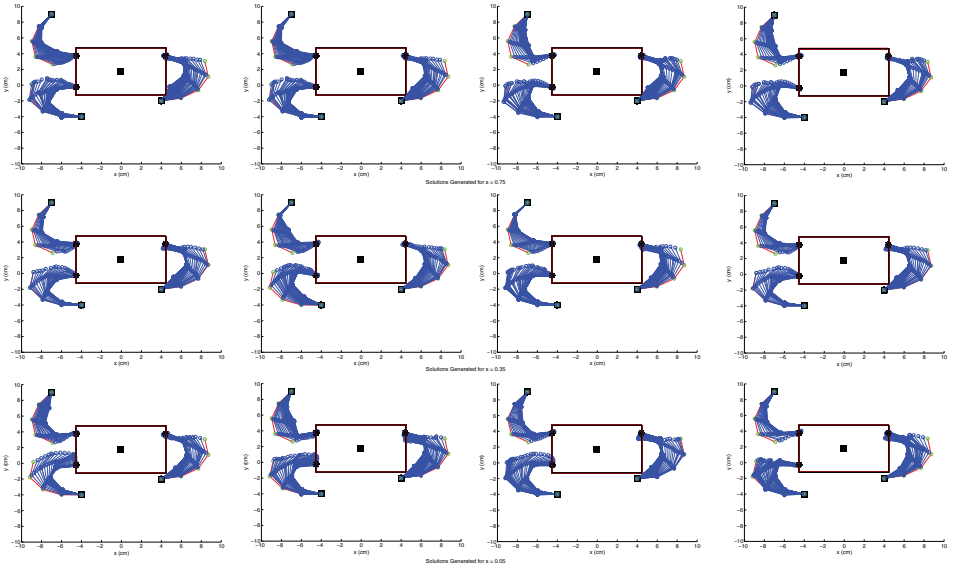

Figure 29. Example of generated grasps (for $s = 0.75$, $s = 0.35$, $s = 0.05$).

6. Computational cost considerations

The state space S of the multi-agent system is composed of the local state spaces $S_1, S_2, ...S_n$of the individual agents. Every local state-space comprises homogeneous state parameters. If we consider the system as a single agent, then $S = S_1 \times S_2 \times \ldots \times S_n$, meaning that the cardinality of the state-space in a centralised single-agent representation is $|S| = |S_n|^n$. By adopting the proposed multi-agent (nested-hierarchical) architecture along with a uniform state definition for every agent, the computational cost of the value iteration problem that we are solving is reduced, as compared to a single-agent approach.

The proposed multi-agent architecture is defined by homogeneous agents, meaning that all of them have the same number of state variables that uniquely define their state for all possible configurations. This implies that the cardinality $|\cdot|$ of every local state-space is the same: $|S_1| = |S_2| = |S_3| = \cdots = |S_i| = |S|$, for every agent i. According to the nested-hierarchical architecture, each agent is able to monitor only those agents that are below in the hierarchy, in order to formulate a joint action. Therefore, the corresponding action space of each agent is reduced, as we move from a higher to a lower level in the hierarchy. Eventually, the cardinality of the joint action space is $|A|^i$, where i is the number of agents that participate in the joint action at the specific level of the hierarchy, and $|A|$ is the number of distinct actions of each agent. Assuming that the state-space is finite, the number of state−action pairs to be updated at every iteration is$|S|\cdot|A|^i$. In order to update the value for a given state−action pair, the maximisation over the joint action space is solved by enumeration over $|A|^i$ elements. So, the cost per iteration is $|S|\cdot|A|^i\cdot|A|^i$ or $|S|\cdot(|A|^i)^2$. Assuming that our algorithm runs for L iterations and for n agents, the total computational cost can be computed as follows:

$$\begin{aligned} L\cdot\sum_{i-1}^{n}\left\{|S|\cdot(A^i)^2\right\} = L\cdot|S|\sum_{i-1}^{n}(|A|^2)^i = L\cdot|S|\cdot\frac{(|A|^2)^{n+1}-|A|^2}{|A|^2-1} = \cdots \\ \cdots = L\cdot|S|(|A|^2)^n\frac{|A^2|-1/(|A|^2)^{n-1}}{|A|^2-1} \simeq L\cdot|S|\cdot(|A|^2)^n\cdot K, \end{aligned} \tag{13}$$

where, for a large value of n, we can assume that

$$K \cong \frac{|A|^2}{|A|^2-1}. \tag{14}$$

Comparing now the above cost with the case of a centralised single-agent system is straightforward. In a single-agent representation, the state parameters, instead of being uniformly distributed among several agents, would be accumulated on a single agent, resulting in an exponential increase of the state-space cardinality. Thus, the computational cost for a single-agent architecture would be

$$L\cdot|S|^n\cdot(|A|^2)^n. \tag{15}$$

Comparing Equation (13) with Equation (15), it is clear that, when the number n of agents increases, the computational cost in the single-agent architecture increases exponentially, since the cardinality of the state space in (15) is raised to the power of n. In the case of the open kinematic chain considered in Section 5.1, the proposed multi-agent system comprises four nested agents, so $n = 4$. The state-space of each agent, as defined in Section 3.3, is composed of six state variables. Each state variable is fuzzified with eight membership functions, according to

the process described in Section 4.1. Therefore, the cardinality of the state-space of each agent is $|S| = 8^6$. Furthermore, as defined in Section 4.2 each agent has three distinct actions, so $|A| = 3$. Each training epoch, within which every agent is allowed to operate, is assumed to have a duration of 1500 iterations, so $L = 1500$. Based on these assumptions, the computational cost when employing the proposed multi-agent architecture equals $L = 1500.8^6 \cdot (3^2)^4 \cdot 3/3^2 - 1 = 2.9 \times 10^{12}$ operations. In the case of a single-agent approach, this cost would be 4.6×10^{28}. Therefore, as the number of agents increases, it is evident that the computational benefit of the proposed multi-agent architecture (as compared to a centralised single-agent approach) becomes significant.

7. Conclusion and future work

This paper proposes a multi-agent model-free learning architecture, particularly adapted in the context of dexterous robot manipulation and evaluated with respect to the developmental acquisition of related kinematic control skills. The main contribution of this work is twofold:

(i) The definition of an original nested-hierarchical multi-agent architecture that encapsulates naturally the topology of robot kinematic chains and supports a recursive, decentralised state-definition scheme. Within the proposed multi-agent architecture, each individual DOF is mapped onto a distinct agent that maintains a local view of the whole system topology and task progress, incrementally updated through a recursive inter-agent communication process.

(ii) The application of a game-theoretic JASM in the context of dexterous robot manipulation, enabling a computationally efficient development of distributed control policies through the application of a fuzzy RL scheme in a continuous domain. Learning is thus approached not through demonstration and training but through an autonomous exploration and self-learning process, where each agent evolves a local sensorimotor behaviour by receiving information (in the form of reward signals) related to observations of task performance. This paper constitutes in fact a proof of concept demonstrating that global dexterous manipulation skills can indeed evolve through such a distributed iterative learning of local agent sensorimotor mappings.

The main motivation of this research work thus was to explore the applicability of such distributed multi-agent approaches to robot manipulation tasks. As compared to typical centralised control schemes, the development of decentralised learning schemes is expected to enhance specific performance properties of the control system, in particular as related to the following: (a) the capacity for developmental self-learning of complex sensorimotor skills, (b) the modularity and intuitive scalability of the control architecture and the direct extensibility of the system into increasingly complex problem domains, (c) the adaptability and inherent robustness of the system with respect to internal structure variations, topological uncertainties and potential unpredictable failures and (d) the superior computational efficiency, achieved by employing distributed state definition and state–action mapping strategies.

These attributes of the proposed system are assessed in this paper through numerical experiments for different robot manipulation task scenarios, involving single or multi-robot kinematic chains. In particular, two main series of numerical experiments are presented in this paper, to evaluate the performance of the proposed multi-agent learning scheme. The first series concerns redundant multi-DOF open kinematic chains and the second series addresses multi-finger grasping. Results regarding generalisation and robustness are presented and analysed in this paper, while scalability with respect to more complex task scenarios is also demonstrated.

It is shown that, by using reward functions that amplify actions leading to faster distance (or force) error minimisation, the multi-agent system statistically and progressively converges towards behaviours (evolving local state-to-action maps) that efficiently resolve kinematic redundancies and achieve desired coordinated action. The numerical experiments described in this paper and the results obtained clearly demonstrate that the solutions which emerge are statistically expected to be feasible and near-optimal in a LS (or effort minimisation) sense, for a wide range of exploration factors and learning parameters. This constitutes an important experimental finding that supports our research hypothesis regarding the applicability of the proposed methodology in such a dexterous manipulation context.

It should be noted here, though, that, in all tested scenarios, the reward functions have been tuned manually, which constitutes indeed one limitation of the approach. Extending this approach by additionally employing methodologies such as apprenticeship learning (Abbeel & Ng, 2004) is currently being investigated, where the development of a reward function can be enhanced by means of human demonstrations performed during an initial training phase. It would then be interesting, in future work, to further explore other reward functions that would probably involve hierarchical task decomposition strategies, that is, attempting to resolve redundancy and evolve coordinated action schemes while exploiting additional DOFs in a target-oriented manner. A more enhanced mathematical analysis regarding the comparison of the emerging solutions with respect to analytic theoretically optimal configurations would also fit within the scope of future research work. In addition, although the motion of the manipulators considered in this work is restricted on a plane, the extension of the framework on a more general 3D manipulation case could be straightforward and would only affect the existing state definition, without altering the basic architectural and algorithmic structure.

The redundant kinematic chains and the quasi-static coordinated grasping task have complexity similar to simple legged locomotion tasks, such as bipedal walking (if we exclude the dynamics). Therefore, modelling the limbs of a legged robot as independent agents appears to be a natural extension of our research work. In this direction, future research activity will investigate the adaptation of our architecture on a quadruped robot trying to learn gait parameters, in order to achieve a desired velocity with certain orientation. Furthermore, the proposed multi-agent system, owing to its homogeneous characteristics (all agents obey the same structural/modular internal architecture), as well as to its hierarchical formation, facilitates scaling to more complex structures. Figure 30 depicts a potential application of the proposed framework, where a more complicated multi-agent topology could be envisaged. By employing the proposed framework in the domain of dexterous articulated robotic mechanisms, we believe that challenging problems in this domain can be tackled in a very elegant and powerful way (in the sense of modularity, robustness and scalability). Similar (in some ways equivalent) problem settings, such as grasp planning, locomotion control or

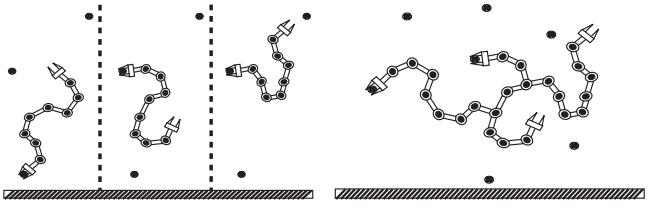

Figure 30. Dexterous robotic chains performing hybrid locomotion/manipulation tasks (such as climbing).

designing optimal climbing (and generally gaiting or locomotion) patterns, could also be approached within the same framework, leading to the notions of evolving cooperative learning and developmental robot control skills.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st international conference on machine learning*, Banff, Alberta, Canada, ICML'04, ACM (pp. 1–8).

Ahmadabadi, M. N., & Nakano, E. (2001). A "constrain and move" approach to distributed object manipulation. *IEEE Transactions on Robotics and Automation*, *17*, 157–172. doi:10.1109/70.928561

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*, 469–483. doi:10.1016/j.robot.2008.10.024

Ben-Israel, A., & Greville, T. N. E. (2003). *Generalized inverses: Theory and applications* (2nd ed). New York, NY: Springer.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.

Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Robot programming by demonstration. In B. Siciliano & O. Khatib (Eds.), *Handbook of robotics* (pp. 1371–1394). Secaucus, NJ: Springer.

Boularias, A., Krömer, O., & Peters, J. (2012). Structured apprenticeship learning. In *Proceedings of the European conference on machine learning – principles and practice of knowledge discovery in databases (ECML-PKDD'12)*, Bristol, UK (pp. 227–242).

Brown, G. W. (1951). Iterative solution of games by fictitious play. In T. C. Koopmans (Ed.), *Activity analysis of production and allocation*, chap. XXIV (pp. 374–376). New York: Wiley.

Calinon, S., D'Halluin, F., Sauser, E. L., Caldwell, D. G., & Billard, A. G. (2010). Learning and reproduction of gestures by imitation. *IEEE Robotics & Automation Magazine*, *17*, 44–54. doi:10.1109/MRA.2010.936947

Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the fifteenth national conference on artificial intelligence*, American Association for Artificial Intelligence, Madison, WI (pp. 746–752).

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience, computational and mathematical modeling of neural systems*. Cambridge, MA: MIT Press.

Donald, D. R., Jennings, J., & Rus, D. (1997). Information invariants for distributed manipulation. *International Journal of Robotics Research*, *16*, 673–702. doi:10.1177/027836499701600506

Doya, K. (1996). Temporal difference learning in continuous time and space. In *Advances in neural information processing systems* (Vol. 8, pp. 1073–1079). Cambridge, MA: MIT Press.

França, C. M., Marley, V., & Karla, F. (2014). Multi-agent systems with reinforcement hierarchical neuro-fuzzy model. *Autonomous Agents and Multi-Agent Systems*, *28*, 867–895.

Fudenberg, D., & Kreps, D. M. (1992). *Lectures on learning and equilibrium in strategic form games* CORE Lecture Series. Louvain-La-Neuve, Belgium: Core Foundation.

Glohon, M. M., & Sen, S. (2004). Learning to cooperate in multi-agent systems by combining Q-learning and evolutionary strategy. In *Proceedings of the world conference on lateral computing*, Dec.

Glorennec, P. Y., & Jouffe, L. (1997). Fuzzy Q-learning. In *Proceedings of the 6th IEEE international conference on fuzzy systems* (pp. 659–662).

Grefenstette, J., & Schultz, A. (1994). An evolutionary approach to learning in robots. In *Proceedings of the machine learning workshop on robot learning, eleventh international conference on machine learning*. Berlin: Springer.

Guenter, F., Hersch, M., Calinon, S., & Billard, A. (2007). Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, *21*, 1521–1544.

Guestrin, C., Lagoudakis, M., & Parr, R. (2002). Coordinated reinforcement learning. In *Proceedings of the 19th international conference on machine learning (ICML'2002)*.

Hollerbach, J., & Suh, K. C. (1987). Redundancy resolution of manipulators through torque optimization. *IEEE Journal on Robotics and Automation*, *3*, 308–316. doi:10.1109/JRA.1987.1087111

Iida, M., Sugisaka, M., & Shibata, K. (2003). Direct-vision-based reinforcement learning in a real mobile robot. *Artificial Life and Robotics*, *7*, 102–106. doi:10.1007/BF02481156

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kaiser, M., & Dillman, R. (1996). Building elementary skills from human demonstration. In *Proceedings of the 1996 IEEE international conference on robotics and automation*.

Karigiannis, J. N., Rekatsinas, T., & Tzafestas, C. (2010). Fuzzy rule based neurodynamic programming for mobile robot skill acquisition on the basis of a nested multi-agent architecture. In *Proceedings of the IEEE/RAS international conference on robotics and biomimetics (RO-BIO' 2010)*, Tianjin, China (pp. 312–319).

Karigiannis, J. N., Rekatsinas, T., & Tzafestas, C. S. (2011). Developmental learning of cooperative robot skills: A hierarchical multi-agent architecture. In V. Cutsuridis, A. Hussain, & J. G. Taylor (Eds.), *Perception-action cycle* Springer Series in Cognitive and Neural Systems (pp. 497–538). New York, NY: Springer, chap. 16.

Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*, 566–580. doi:10.1109/70.508439

Kober, J., & Peters, J. (2009). Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems* (Vol. *21*, pp. 849–856). Cambridge, MA: MIT Press.

Kok, J. R., & Vlassis, N. (2004). Sparse tabular multiagent Q-learning. In *Proceedings of the annual machine learning conference of Belgium and the Netherlands*, Brussels.

Kondo, T., & Ito, K. (2004). A reinforcement learning with evolutionary state recruitment strategy for autonomous mobile robots control. *Robotics and Autonomous Systems*, *46*, 111–124. doi:10.1016/j.robot.2003.11.006

Kormushev, P., Calinon, S., & Caldwell, D. G. (2013). Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, *2*, 122–148. doi:10.3390/robotics2030122

Lauera, M., & Riedmiller, M. (2004). Reinforcement learning for stochastic cooper-ative multi-agent systems. In *Third international joint conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, Vol. 3.

LaValle, S. M., & Kuffner, J. J. (2001). Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, & D. Rus (Eds.), *Algorithmic and computational robotics: New direction* (pp. 293–308). Wellesley, MA: A K Peters.

Lemon, O., & Pietquin, O. (2012). *Data-driven methods for adaptive spoken dialogue systems: Computational learning for conversational interfaces*. New York, NY: Springer.

Liu, J., Parker, L. E., & Madhavan, R. (2007). Reinforcement learning for autonomous robotic fish. In N. Nedjah, L. dos Santos Coelho, & L. de Macedo Mourelle (Eds.), *Mobile robots: Evolutionary approach* (Vol. 50, pp. 121–135). Berlin: Springer.

Lopes, M., & Santos-Victor, J. (2007). A developmental roadmap for learning by imitation in robots. *IEEE Transactions on Systems, Man, and Cybernetics (Part B)*, *37*, 308–321. doi:10.1109/TSMCB.2006.886949

Matsui, T., Omata, T., & Kaniyoshi, Y. (1992). Multi-agent architecture for controlling a multi-finger robot. In *Proceedings of the 1992 IEEE/RSJ international conference on intelligent robots and systems (IROS'92)*, Raleigh, NC.

Murciano, A., Millan, J. R., & Zamora, J. (1997). Specialization in multi-agent systems through learning. *Biological Cybernetics*, *76*, 375–382. doi:10.1007/s004220050351

Myerson, R. B. (1997). *Game theory: Analysis of conflict*. Cambridge, MA: Harvard University Press.

Niederreiter, H. (1992). Random number generation and Quasi-Monte Carlo methods. In *Proceedings of the SIAM CBMS-NSF regional conference series in applied mathematics* (Vol. 63, SIAM, Philadelphia, PA.

Pardo, D. E., Angulo, C., del Moral, S., & Català, A. (2009). Emerging motor behaviors: Learning joint coordination in articulated mobile robots. *Neurocomputing*, *72*, 3624–3630. doi:10.1016/j.neucom.2009.01.015

Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., & Schaal, S. (2011). Skill learning and task outcome prediction for manipulation. In *Proceedings of the IEEE international conference on robotics and automation (ICRA'2011)*, Shanghai, China (pp. 3828–3834).

Rozo, L., Calinon, S., Caldwell, D. G., Jimenez, P., & Torras, C. (2013). Learning collaborative impedance-based robot behaviors. In *Proceedings of the AAAI conference on artificial intelligence*.

Rummery, G. A. (1994). *On-line Q-learning using connectionist systems* (Ph.D. Thesis). Cambridge University, England.

Rus, D. (1997). Coordinated manipulation of objects in a plane. *Algorithmica*, *19*, 129–147. doi:10.1007/PL00014414

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, *3*, 233–242. doi:10.1016/S1364-6613(99)01327-3

Schaal, S., Mohajerian, P., & Ijspeert, A. J. (2007). Dynamics systems vs. optimal control a unifying view. *Progress in Brain Research*, *165*, 425–445.

Schmill, M., Anderson, M. L., Fults, S., Josyula, D., Oates, T., Perlis, D., … Wrights, D. (2010). The metacognitive loop and reasoning about anomalies. In M. Cox & A. Raja (Eds.), *The Metareasoning, Thinking about Thinking* (pp. 183–198). Cambridge, MA: MIT Press, chap. 12.

Shibata, K., & Ito, K. (2002). Effect of force load in hand reaching movement acquired by reinforcement learning. In *Proceedings of the 9th international conference on neural information processing (ICONIP'02)*, Computational Intelligence for the E-Age.

Shibata, K., & Ito, K. (2003). Hidden representation after reinforcement learning of hand reaching movement with variable link length. In *Proceedings of the international joint conference on neural networks (IJCNN'2003)* (Vol. 4, pp. 2619–2624).

Shibata, K., & Okabe, Y. (1994). A robot that learns an evaluation function for acquiring of appropriate motions. In *World congress on neural networks*, San Diego, June, International Neural Network Society Annual Meeting (Vol. 2, pp. 29–34).

Shibata, K., & Okabe, Y. (1995). *Smoothing-evaluation method in delayed reinforcement learning*, Technical report, University of Tokyo, Research Centerfor Advanced Science and Technology (RCAST), http://shws.cc.oita-u.ac.jp/shibata/RL/summary.html

Shibata, K., Sugisaka, M., & Ito, K. (2001). Fast and stable learning in direct-vision-based reinforcement learning. In *Proceedings of the 6th international symposium on artificial life and robotics (AROB)* (pp. 562–565).

Shoham, Y., & Tennenholtz, M. (1992). On the synthesis of useful social laws for artificial agent societies. In *Proceedings of the 1992 AAAI conference (AAAI'92)* (pp. 276–281).

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modeling, planning and control*. London: Springer.

Smart, W. D. (2002). *Making reinforcement learning work on real robots* (Ph.D. Thesis). Brown University.

Stulp, F., Buchli, J., Theodorou, E., & Schaal, S. (2010). Reinforcement learning of full-body humanoid motor skills. In *Proceedings of the IEEE international conference on humanoid robots (Humanoids)*, Nashville, TN, USA (pp. 405–410).

Sutton, R. S. (1996). Generalization in reinforcement learning: successful example using sparse coarse coding. In *Advances in neural information processing systems* (Vol. 8, pp. 1038–1044). Cambridge, MA: MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics (SMC)*, *15*, 116–132. doi:10.1109/TSMC.1985.6313399

Takahashi, T., Tanaka, T., Nishida, K., & Kurita, T. (2001). Self-organization of place cells and reward-based navigation for a mobile robot. In *Proceedings of the international conference on neural information processing (ICONIP'2001)*.

Watkins, C. (1989). *Learning from delayed rewards* (Ph.D. Thesis). University of Cambidge, England.

Whitney, D. (1972). The mathematics of coordinated control of prosthetic arms and manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, *94*, 303–309. doi:10.1115/1.3426611

Yakey, J. H., LaValle, S. M., & Kavraki, L. E. (2003). Randomized path planning for link ages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, *17*, 951–958.

Zhang, C., & Lesser, V. (2013). Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 international conference on autonomous agents and multiagent systems (AAMAS'13)*, Minnesota USA.