



An imitation learning approach for the control of a low-cost low-accuracy robotic arm for unstructured environments

Fabio Bonsignorio^{1,2} · Cristiano Cervellera³ · Danilo Macciò³ · Enrica Zereik³

Received: 6 December 2021 / Accepted: 4 September 2022 / Published online: 11 November 2022
© The Author(s) 2022

Abstract

We have developed an imitation learning approach for the image-based control of a low-cost low-accuracy robot arm. The image-based control of manipulation arms is still an unsolved problem, at least under challenging conditions such as those here addressed. Many attempts for solutions in the literature are based on machine learning, generally relying on deep neural network architectures. In typical imitation approaches, the deep network learns from a human expert. In our case the network is trained on state/action pairs obtained through a Belief Space Planning algorithm, a stochastic method that requires only a rough tuning, particularly suited to unstructured and dynamic environments. Our approach allows to obtain a lightweight manipulation system that demonstrated its efficiency, robustness and good performance in real-world tests, and that is reproducible in experiments and results, despite its inaccuracy and non-repeatable kinematics. The proposed system performs well on a simple reaching task, requiring limited training on our quite challenging platform. The main contribution of the proposed work lies in the definition and real-world testing of an efficient controller, based on the integration of Belief Space Planning with the imitation learning paradigm, that enables even inaccurate, very low-cost robotic manipulators to be actually controlled and employed in the field.

Keywords Inaccurate lightweight manipulator · Imitation learning · Belief space planning · Soft robotics

1 Introduction

Many different control strategies have been proposed in the literature for robotic manipulation, mainly dealing with a specific problem and environment. Still, the proposed solutions often fail to achieve the necessary robustness and dexterity, especially with lightweight arm structures operating

in open-ended environments and/or with inaccurate or soft bodies, noisy sensing and actuation, as in our case. Furthermore, many of the systems proposed in the literature are only simulated or very few tests in real-world environments have been performed. This is very limiting since the transition from simulation to reality always strongly reduces the system nominal performance (which are usually difficult to realistically evaluate in simulation). This appears very clearly in Bonsignorio and Zereik (2021), which explicitly demonstrated that, for lightweight, inaccurate robotic structures like H₂Arm, classic control algorithms are not robust enough whenever a substantial number of real-world tests are performed. In fact, the behaviour of a classic PID (Proportional Integral Derivative) control was there evaluated against a stochastic BSP (Belief Space Planning) methodology; conclusions drawn from such comparison highlighted a twofold important result: the BSP has a success rate remarkably higher than the PID (93.3 vs 40.0%), but in the meantime it is much slower (31.5 s vs 109.6 s as mean execution time throughout all the experiments).

Recent approaches to the manipulation problem in unstructured and uncertain environments rely on machine

✉ Enrica Zereik
enrica.zereik@cnr.it

Fabio Bonsignorio
fabio.bonsignorio@heronrobots.com;
fabio.bonsignorio@fer.hr

Cristiano Cervellera
cristiano.cervellera@cnr.it

Danilo Macciò
danilo.maccio@cnr.it

¹ Heron Robots, Via Malta 3/7, 16121 Genoa, Italy

² FER, University of Zagreb, Unska Ulica 3, 10000 Zagreb, Croatia

³ Institute of Marine Engineering, Italian National Research Council, Via de Marini 16, 16149 Genoa, Italy

learning techniques. Such methods typically exploit the reinforcement learning (RL) paradigm to derive controllers able to learn and adapt in order to successfully execute the required tasks. In most proposed approaches a deep neural network is employed, trained either on datasets featuring state variables or directly on images.

In our paper an imitation learning approach is proposed to train a low-accuracy low-cost arm to perform a simple reaching task, a critical and necessary step in every manipulation problem.

The adoption of the imitation learning strategy allows to cope with the effort required to perform experiments with a real robot. In fact, with respect to approaches based on reinforcement learning, exploiting the experience of an expert teacher avoids the need for a very large number of experiments before the controller starts learning something meaningful. Indeed, a very large number of experiments would be impossible in practice with a low-accuracy arm and uncertain setting such as those here considered. In particular, we implemented the behavioral cloning paradigm (Bain and Sammut 1999), which involves setting the problem of imitation as a supervised learning problem.

However, while in the typical imitation learning paradigm the controller learns from a human expert, we follow the less common approach of training from an algorithmically generated dataset, like for example in Seita et al. (2020) and train a deep neural network on datasets coming from a Belief Space Planning (BSP) algorithm, representing the teacher to be imitated by the network. The BSP belongs to the class of stochastic algorithms, which provide better performance with respect to deterministic ones in unstructured and dynamic environments. It is able to concurrently drive the manipulator to the goal and reduce uncertainty; hence, it is very suitable for manipulation in open-ended environments. The main drawback of the BSP approach is the computational burden to obtain the next action, which severely limits its real-time applicability. The introduction of a neural controller that learns to imitate its behavior is thus aimed at providing a controller that yields, in real-time, the same control action that the BSP would compute in the current conditions. In our case, the data obtained from the set of system trajectories corresponding to state-control action pairs computed through the BSP method are used to learn a mapping able to associate to each control the corresponding output.

In this context, the main objective of the work is not only to define an imitation learning technique for inaccurate structures, but also to support the analysis through real-world experimentation. To this purpose, a particularly difficult test architecture has been selected, consisting in a very inaccurate and non-repeatable structure, namely H₂Arm (Bonsignorio and Zereik 2021). Such system is an *ad hoc* designed, 3D-printed open-source tendon-driven

robotic arm, with a total cost of less than 200 € overall and a weight of less than 1 kg. Its intrinsic unknown dynamics and uncertainties make it very suitable to simulate errors and non-modelled effects, in order to stress and test the algorithmic approach. H₂Arm motors are not endowed with any sensor (e.g. encoders) providing current position. The only feedback signal is produced by a simple low-quality camera mounted on the arm wrist. Hence, the robotic platform turns out to be very challenging for controllers, in terms of both uncertainties and unknowns: it is not possible to compute inverse and forward kinematics from measurements, and a model-free controller should be applied.

As in the previous work Bonsignorio and Zereik (2021), also the work reported here aims at the assessment and validation of the proposed control strategy through the conducted experimental real-world test campaign on H₂Arm.

In our experiments performed under various conditions the neural BSP-based controller yielded a strong improvement in the execution time with respect to the original BSP: overall, the neural controller turned out to be about 10 times faster than the BSP one, making it actually exploitable for real-time robust manipulation. Moreover, a great advantage of this system is its high experimental reproducibility: different controllers may be applied and their performance compared.

We remark that we tested our system in a laboratory context but we target the underwater environment for H₂Arm final application; this is one of the strongest motivations that justify the employment of a stochastic control framework integrated with a neural controller. In fact, the unstructured environment poses great challenges in terms of control performance and decisional autonomy, and is one of the major reasons why robots are not really employed in everyday tasks in close contact with humans, yet.

Concerning the proposed controller, although more advanced imitation learning strategies could be applied (such as, for instance, inverse reinforcement learning), behavioral cloning was chosen for our preliminary tests due to its simplicity. In fact, this basic imitation scheme proved to be good enough to yield suitable performance in our experiments. The developed deep learning controllers have been able to execute the required task successfully in all the tests; they showed good extrapolation properties and robustness also in case of noise, injected to stress the system and draw conclusion about its robustness.

The paper is organized as follows: in Sect. 2 a comprehensive literature overview is provided, while the experimental robotic platform is described in Sect. 3. The proposed control strategy is detailed in Sect. 4 and the obtained results are presented in Sect. 5. Finally, Sect. 6 draws conclusions and depicts the foreseen future research activities.

2 Related works

H₂Arm is a specific example of compliant robot arm. It is a common opinion that soft structure robots could, in principle, make the manipulation task easier, therefore, a number of partially compliant robots has been proposed. In particular tendon-driven manipulators have been proposed to imitate human arms driven by muscles. Traditional approaches to robotic manipulation control rely on linearized models and accurate sensing and actuation. Reducing weights and increasing compliance, as with H₂Arm and typically in soft robotics applications, leads to a dramatic increase in non-linearities and uncertainties both in the dynamics and the measures, thus making most widely used methods unsuitable. The most promising approaches are based on stochastic planning and control methods such as BSP, or through Machine Learning (ML) and in particular Deep Learning (DL). These approaches will be discussed later in this section. The GummiArm is an example of compliant robotic manipulator, similar to ours, 3D-printed and driven by a combination of accurate (and expensive) digital servos with encoders and viscoelastic composite tendons. It has been used to test several advanced bio-mimetic control strategies (Stoelen et al. 2016). A non-linear robust control strategy for a tendon-driven manipulator is presented in Okur et al. (2015). The proposed approach considers, like in our work, uncertainties and external disturbances. However, such control strategy needs to know actuator positions and velocities, and also forces generated on tendons; moreover, it has been tested only in simulation. A visual-servoing control for a cable-driven soft manipulator is reported in Wang et al. (2016), where the authors discuss the effect of system linearization and approximation, reducing system complexity but also sacrificing accuracy. A planning algorithm for a soft planar arm is proposed in Marchese et al. (2014), where a series of constrained optimization problems is solved to find a locally optimal inverse kinematics, introducing errors and limitations. The environment is supposed to be known *a priori*, which is a very strict and limiting hypothesis. In Subudhi and Morris (2009), soft computing techniques are applied to the control of a multi-link flexible manipulator. Such methodology is suited to control systems that are particularly difficult to be modelled, and provides a greater tolerance to imprecision. An overview of soft robotics approaches is presented in Rus and Tolley (2015), whose conclusion is that an effective control of such structures needs new models and algorithms to iteratively learn the necessary manipulation skills.

While, for example, advanced solutions leveraging on the group regularities in the movement and local displacements of mechanical systems (Bonsignorio 2013), have been proposed for the control of systems like ours, we have

focused on a simpler and more widely used approach, i.e., BSP (Zereik et al. 2015; Agha-Mohammadi et al. 2014; Platt et al. 2010). Belief Space Planning belongs to the family of stochastic control approaches, aimed at increasing the effectiveness and robustness of control methods on ‘real-world robots’. In Bonsignorio and Zereik (2021) authors report on the performance comparison of a standard PID (Proportional Integral Derivative) and Belief Space Planning control implemented on H₂Arm. That work clearly demonstrates the capability of the BSP, with very rough tuning, to perform a simple reaching task. However, this method is too slow for real-time control. This motivated us, as told above, to find more effective solutions based on an imitation learning approach exploiting deep neural networks.

Machine Learning approaches have focused on the identification (and composition) of ‘atomic’ chunks of motion trajectories, ‘motion primitives’ and related planned tasks, ‘motor skills’ through Reinforcement Learning, for example based on policy gradients and various policy search techniques (more on this point later in the section). To cope with the real-time requirements of robots, other approaches have developed incremental online learning schemes suitable for high dimensional spaces. Similar methods have been applied to the learning of the inverse kinematics – the joint trajectories corresponding to the desired end-effector trajectories – of the robot itself. An adaptive neural network controller is proposed in Xie et al. (2010), but only simulation has been used to evaluate the approach, without any experimental validation. Other approaches that exploit neural networks are in Rolf et al. (2015), which is based on learning the inverse kinematics/statics of soft manipulators and Thuruthel et al. (2017), where the forward dynamics of the model is learned using a RNN (Recurrent Neural Network) in order to enable predictive control.

Reinforcement Learning and especially Deep Reinforcement Learning (DRL) methods have been adapted to various planning, grasping and manipulation problems. Efforts have been dedicated to make learning faster – implementing asynchronous and parallelization processes – and to develop realistic simulation – and appropriate simulation trial randomization procedures – to reduce the needed time-consuming learning runs on ‘real robots’, (Kober et al. 2013; Rusu et al. 2016). In Bonsignorio et al. (2020) some key theoretical issues are discussed, such as: how to exploit the manifold structure of data, how to characterise temporal features, how to implement unsupervised learning, and multifidelity reinforcement learning as well as the assurance and verification of autonomous operations, see Bonsignorio et al. (2020). Reference DL implementations of Reinforcement Learning approaches can be found in the DeepMind Control Suite (Tassa et al. 2018) and OpenAI Gym (OpenAI 2020). Both platforms implement a number of test cases in simulation in a partially reproducible way. New approaches based on

Contrastive Unsupervised Learning have shown some promise, see Srinivas et al. (2020). Deep reinforcement learning is exploited in more recent research, as in Gu et al. (2017), which proposes the utilization of many different robots to gather data to train the neural network; this reduces the training time, but increases complexity (for instance, it is difficult to integrate different experiences learnt by different robots and to obtain a generalizable policy) and costs (many robots needed). A similar approach is adopted in Ebert et al. (2018), where a Model-Predictive Control (MPC) is combined with a retrying option to learn manipulation skills; as in previous work, a really huge amount of training data is collected and employed, using two different robots. A lower amount of training demonstrations is needed in Zhu et al. (2018), where a model-free deep reinforcement learning algorithm is applied to visuomotor tasks. However, a quite low success rate is obtained, when simulation results are transferred to real-world experiments.

The imitation learning framework, that we merge in our work with BSP for H_2 Arm control, consists of a heterogeneous set of techniques able to emulate a control scheme through the imitation of the controlled system itself during its operating phase. The first techniques studied in this field are those referring to behavioral cloning (Bain and Sammut 1999). Some behavioral cloning methods make use of local models based on kernel functions and also provide notions of convergence to the reference controller (Macciò 2016; Macciò and Cervellera 2012); others use different approximators and are more application-oriented (Bojarski et al. 2016; Sammut et al. 1992). In order to overcome the issue of having a not enough representative dataset, some behavioral cloning methods have evolved further by introducing techniques to artificially increase the available dataset (Pomerleau 1989; Giusti et al. 2016). In particular, the method known as Dataset Aggregation (DAgger) (Ross 2011), which has recently shown promising performance in several applications (such as, e.g., Ross 2012), is noteworthy. These techniques, however, show limitations when applied in contexts where the specific target is complex and the control scheme to be imitated is implemented by one or more human operators. In this case, techniques that try to infer the latent reward that is at the base of the policy generated by the reference controller are preferred. Probably, the best known methods in this case are the Inverse Reinforcement Learning (Abbeel and Ng 2004) and its derivative, the Apprenticeship Learning (Ng and Russell 2000), which, as mentioned, propose to use the trajectories generated by the controlled system to infer the reward signal. This, in turn, allows to reconstruct the reference controller. Other methods belonging to this category are the (Deep) Maximum Entropy Inverse Reinforcement Learning (Ziebart et al. 2008; Wulfmeier et al. 2015), the Guided Cost Learning

(Finn et al. 2016) and the Generative Adversarial Imitation Learning (Ho and Ermon 2016).

The imitation learning paradigm, for tasks similar to the one we study, has been proposed by several authors. In Rahmatizadeh et al. (2018) a cheaper manipulator is considered, being endowed with rigid links and better sensors w.r.t. H_2 Arm. A behavioral cloning approach is applied to such platform in order to learn some manipulation tasks; again, a huge amount of data (acquired through demonstration by a human operator) is needed to train the system. A virtual reality-based teleoperation approach is employed for collection of (again, a lot of) training data in Zhang et al. (2018). In Odetti et al. (2020), a behavioral cloning approach based on reservoir computing, trained by inexperienced human teachers during a festival, was successfully implemented for the control of a marine Unmanned Surface Vessel (USV).

Almost all of the cited works exploiting machine learning techniques employ robots with quite complex and expensive structures (that can be easily modelled starting from the mechanics), with precise and accurate sensors and a reasonably repeatable behavior. We follow an approach similar to the one described in Seita et al. (2020). However, in comparison to that study, where the authors train the imitation learning systems on a synthetic dataset generated by a fabric simulator and an algorithmic supervisor that has access to complete state information, there are significant differences. In our case we use the data calculated by a stochastic planner.

3 H_2 Arm

In this section we present H_2 Arm, the reference soft robotic structure we consider for the proposed methodology. Both its hardware and software architectures are described, as well as its “standard” controller based on BSP.

3.1 Hardware and software architecture

H_2 Arm, depicted in Fig. 1, is a 3D-printed low-cost and inaccurate manipulator designed and developed within the Joint Lab between the Institute of Marine Engineering of the Italian National Research Council and Heron Robots. It has 4 degrees of freedom (DOFs), $J_1 - J_4$; the first three are rotational joints, while the last one is a prismatic joint. Each rotational joint is equipped with two counteracting motors winding up a tendon-like rope. The last prismatic joint is directly moved by one motor, without the need of tendons. The joint configuration allows to perform visual-servoing tasks consisting in reaching by tracking and centering objects with respect to the image plane. H_2 Arm is equipped with a commercial low-cost and inaccurate webcam, mounted on top the mobile part of J_4 , and hence

Fig. 1 3D-printed H₂Arm and related reference frames: $\langle c \rangle$ is the camera frame centered on the webcam, and $\langle w \rangle$ is the world frame, assumed to be placed in the center of the first joint (the big yellow horizontal disk). The main defined kinematic quantities for H₂Arm control are also depicted (color figure online)

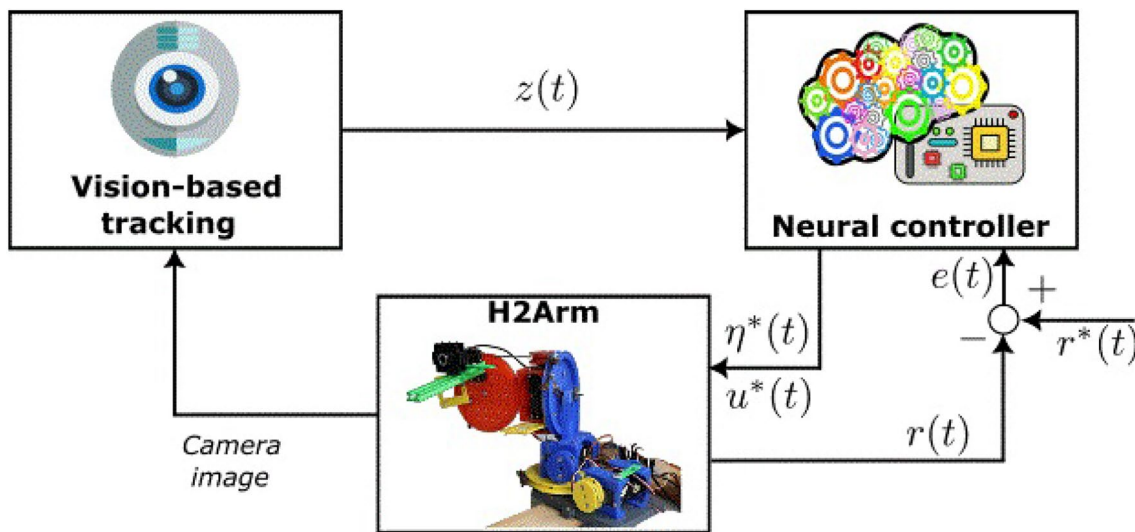
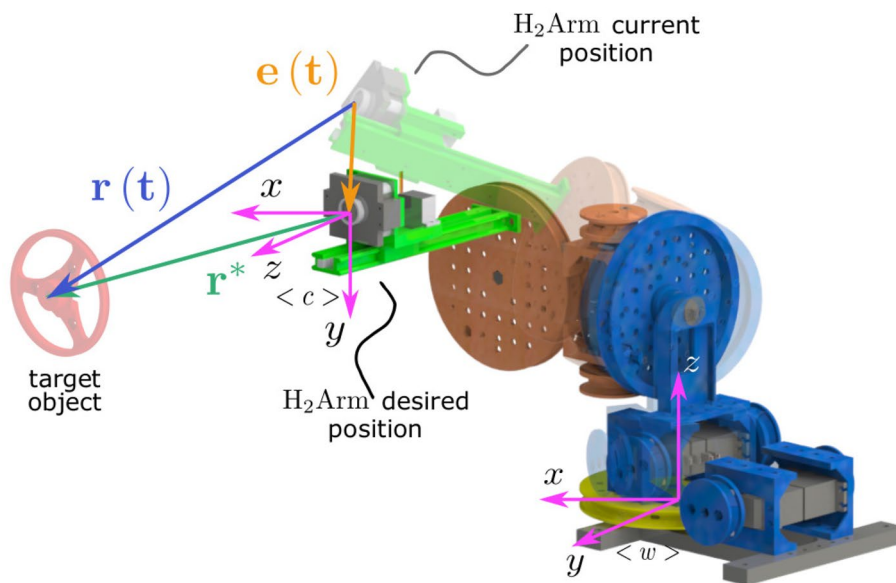


Fig. 2 H₂Arm architecture

moving integrally with the last prismatic joint. H₂Arm low-level is driven by a Pololu Mini Maestro 12-Channel USB Servo Controller (Pololu 2020), suitably generating PWM (Pulse Width Modulation) signals to drive the motors.

The software architecture is composed by two different modules: i) the vision-based recognition algorithm, in charge of measuring the red blob 3D position, and written in C++; ii) the neural controller (that will be introduced in Sect. 4), in charge of driving H₂Arm to track and center the blob in the image plane, and written in Python. This last module receives the measurements about the red blob 3D position with respect to frame $\langle c \rangle$ (refer to Fig. 1) from the recognition algorithm through a standard TCP socket.

The same simple vision-based recognition algorithm implemented and exploited in Bonsignorio and Zereik (2021) was employed to retrieve the red blob position in 3D space with respect to the arm camera. This allows to directly compare system performance with respect to different trials and controllers. H₂Arm introduces many uncertainties in the control: first of all, its tendon-based structure produces unwanted forces that affect motion; furthermore, the (very low-cost and inaccurate) motors do not provide reliable and repeatable motion, and the only available feedback comes from the camera, measuring the arm distance from the center of the object of interest. The camera itself introduces an error on

the object estimation. A block diagram summarizing the H_2 Arm architecture is depicted in Figure 2.

3.2 Reference BSP controller

The reference control scheme under which H_2 Arm operates is based on the Belief Space Planning algorithm. For details on theory and experiments of the BSP technique applied to H_2 Arm, the interested reader can refer to Bonsignorio and Zereik (2021) and related supplemental material (<https://ieeexplore.ieee.org/ielx7/100/9535319/9186201/supp1-3014279.pdf?arnumber=9186201>).

BSP is a stochastic control strategy able to concurrently: (i) optimize the goal-reaching probability, and (ii) reduce uncertainties acting on the system. The disturbance affecting the observations is represented by the algorithm with a linear Gaussian model and the assumption of maximum likelihood observations is made. As proved in Platt et al. (2010), this assumption allows to apply non-linear optimization techniques in the belief space (that is the juxtaposition of the mean and the covariance matrix columns of the state). The actual system state is, in fact, not directly measured and only its (Gaussian) probability density function (pdf) is assumed to be known (the pdf mean is close to the current arm position and the covariance matrix represents the confidence of this information).

In brief, the algorithm works as explained in the following:

- a trajectory moving the end-effector from the current point to the next one is planned;
- such trajectory is linearly approximated by Direct Transcription with a series of segments¹;
- the algorithm moves piecewise on the trajectory. The transition between segment extremes is accomplished through LQR that computes the needed control action, namely the Cartesian velocity $\eta^*(t)$ to be followed by the end-effector. This procedure is iterated until the final goal is reached;
- from the current reference point, the reference arm joint velocities $\dot{q}^*(t)$ are computed through inverse kinematics. Then, the reference motor velocities $u^*(t)$ are esti-

mated by the robot Low-Level Control (LLC), applying the motor model. The LLC is in charge of managing the physical interaction with the robot hardware;

- at each step, the algorithm checks how much the computed plan contributes to the final goal reaching. If the planned trajectory deviates too much from the desired one, re-planning occurs.

The BSP algorithm is graphically summarized by the flow-chart of Fig. 3.

Notice that the joint velocities $\dot{q}^*(t)$ are obtained exploiting a model of the robot equations. In our case, due to the inaccurate structure of H_2 Arm, the resulting values are expected to be very noisy. Yet, the BSP algorithm is able to cope with these uncertainties. In fact, since the BSP control actions balance the objective to reach the goal with the one to reduce the covariance matrix of the observations, they can also be considered as information-gathering actions. Note that the planning is performed in a higher dimensional space (with respect to the state space), even in case of a coarse-grained approximation of the belief space. Furthermore, the belief state dynamics is non-linear, stochastic and inherently under-actuated (the number of control inputs for the physical system is lower than the belief space dimension). To this aim, it is possible to simplify this very complex problem through the already mentioned assumption of maximum likelihood observations. This assumption means that the current system state is supposed to be the most likely state according to the past observations and the performed actions; this is equal to state that the control actions are able to achieve their intended purpose, thus leading the system to the desired state.

4 DL-BSP control

Despite the robustness and efficiency of BSP, obtaining the optimal action for the current state requires a computational effort that is, in general, unfeasible in real-time tasks. Here, we trained a deep neural network to imitate (clone) the BSP controller behavior, and be able to replicate it in real-time without the aforementioned computational burden. We refer to this controller as DL-BSP.

We assume that all the notation in the following is referred to frame $\langle c \rangle$ (see Fig. 1). Let us assume the time horizon during which the task is performed is divided in T discrete intervals at which actions are performed and measurements are collected. Denote by $r^* = [r_x^*, r_y^*, r_z^*]^T$ the desired final position of the end-effector with respect to the target object. Define, for $t = 0, \dots, T - 1$, $r(t) = [r_x(t), r_y(t), r_z(t)]^T$ as the current end-effector position with respect to the object, estimated by the camera at time t . Finally, define the error $e(t) = r(t) - r^* = [e_x(t), e_y(t), e_z(t)]^T$ at time t . Figure 1

¹ Direct transcription is a method of trajectory optimization (in our case in the belief space) used together with non linear optimization techniques, in our case Linear Quadratic Regulation (LQR). Direct Transcription methods translate a continuous optimization problem, like in our case that of optimizing a trajectory in the belief space, into a discrete one by evaluating the problem in a finite set of nodes obtained by time discretization. The set of differential equations governing the motion in the belief space is ‘transcribed’ into a finite set of equality constraints. This allows to solve the optimal trajectory problem with the degree of accuracy of the numerical optimizer used (the interested reader can refer to Betts (2010) for details).

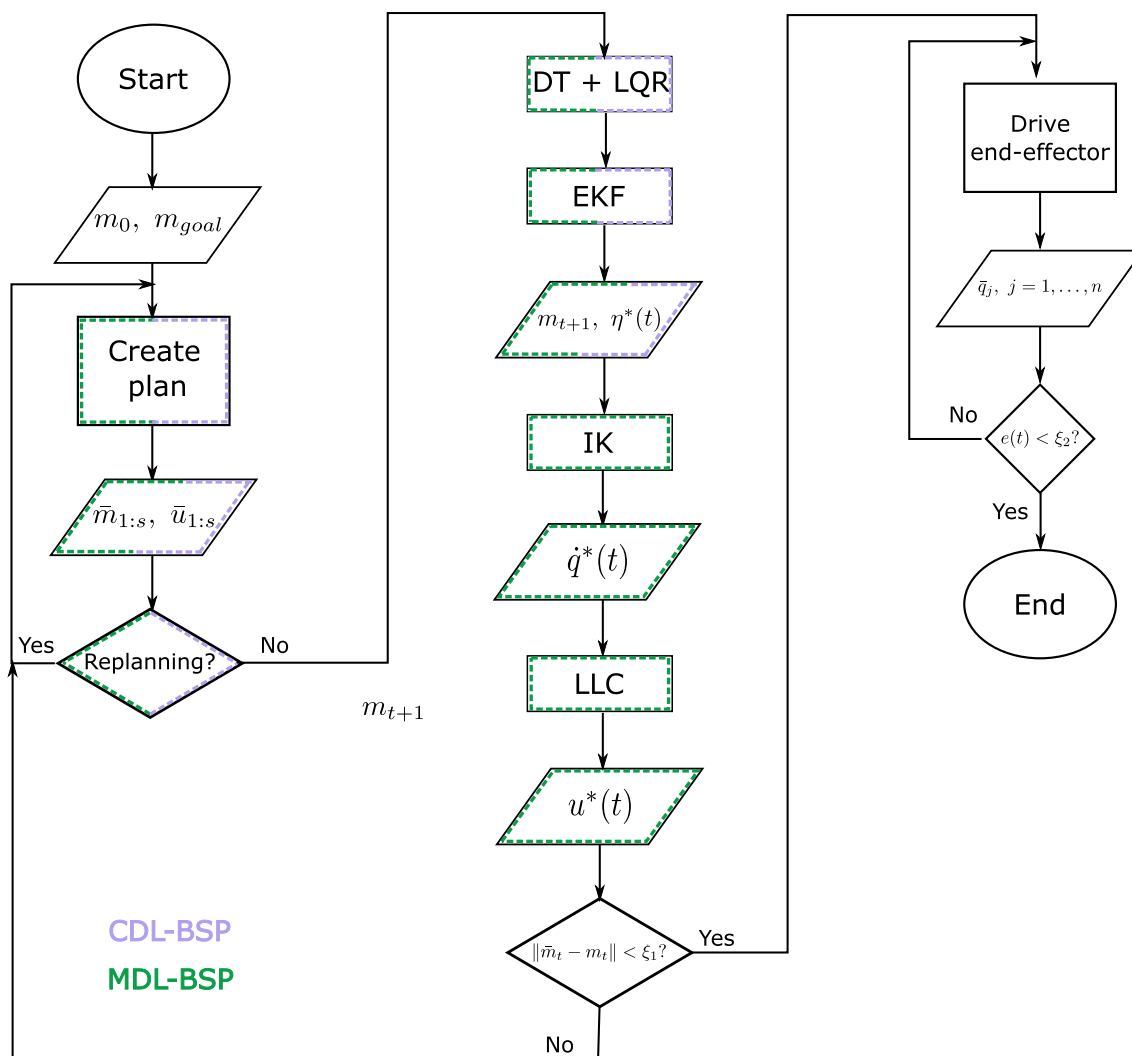


Fig. 3 Flowchart of the BSP algorithm and its relation with the neural controllers MDL and CDL. Note that m_0 and m_{goal} are the initial and final belief state mean, respectively; $\hat{m}_{1:s}$ and $\hat{u}_{1:s}$ are the planned trajectories for the belief state mean and the related control actions; $\eta^*(t)$ is the Cartesian reference velocity for the end-effector, $\dot{q}^*(t)$ and $u^*(t)$ are the reference joint trajectories and the motor velocity commands, respectively. Finally, \bar{q}_j are the joint positions and ξ_1, ξ_2 are

pre-defined thresholds related to the belief state and the Cartesian error, respectively. DT, LQR and EKF stand for Linear Quadratic Regulator, Direct Transcription and Extended Kalman Filter; IK and LLC denote Inverse Kinematics computation and Low-Level Control of the robot. The blocks whose behavior is incorporated in the CDL-BSP and MDL-BSP are highlighted in green and violet, respectively.

provides an overview of the defined kinematic vectors $r^*, r(t)$ and $e(t)$ for H_2 Arm in the execution of the required visual-servoing task.

In order to train the deep neural controller, the first step consists in defining the training set of pattern/targets for a supervised learning procedure. As said, the patterns are derived from the arm trajectories obtained by BSP, while the targets are derived from the corresponding actions.

To this purpose, recall that $u^*(t)$ is the vector of the velocities assigned by BSP to the various motors at stage t . In particular, we write $u^*(t) = [v_1^*(t), \dots, v_M^*(t)]^T$, where M

is the total number of motors. At each stage t , the optimal Cartesian velocity reference $\eta^*(t)$ given the error $e(t)$ is computed by the BSP algorithm, to which correspond the motor velocities $u^*(t)$ obtained as described in Sect. 3.2.

We consider two setups, corresponding to two different training sets for the behavioral cloning procedure:

- *Cartesian Deep Learning (CDL) control*: the deep learning model provides, at every temporal stage t , an approximation of the optimal Cartesian velocity $\eta^*(t)$; then, the corresponding motor velocities are computed through inverse kinematics and the LLC model.

- *Motor Deep Learning (MDL) control*: the deep learning model directly estimates the vector of motor velocities $u^*(t)$.

The Cartesian control involves the approximation of a generally smoother and lower-dimensional function with respect to the motor velocities. On the other hand, the motor control avoids the need to compute the pseudo-inverse of the Jacobian matrix that, in case of robots with many links and joints, can be very large and lead to numerical issues. Further, for some kinds of robots, the structure exact kinematic model could be unavailable.

Consider a set of N successful trials obtained with the BSP, i.e., tasks in which the robot successfully reached the target within a maximum number of steps. In the case of Cartesian control, for $j = 1, \dots, N$, define the j -th trajectory as

$$\tau_j = \{(e_j(0), \eta_j^*(0)), \dots, (e_j(T_j - 1), \eta_j^*(T_j - 1))\}$$

where the subscript j in e and e^* denotes the j -th trajectory and T_j is the final time before trajectory j converged to the target. Similarly, we define the j -th trajectory in the case of motor control as

$$\tau_j = \{(e_j(0), u_j^*(0)), \dots, (e_j(T_j - 1), u_j^*(T_j - 1))\}$$

According to the definitions above, a trajectory is the collection of the observed errors, paired with the corresponding feature of the BSP control scheme that we want to imitate with the learning model.

The aim of the imitation controller is to learn from these sets of trajectories a mapping from the observed errors and the desired BSP output feature, in order to estimate online the action that would be yielded by the BSP approach for the current state.

For the sake of introducing the imitation learning scheme, let us focus on the motor control setting and denote by $u(t)$ the vector of motor velocities produced by the imitation controller at stage t . Notice that in the case of Cartesian control it is sufficient to replace $u(t)$ with $\eta^*(t)$ in the rest of the section.

For the generation of the closed-loop control $u(t)$, due to the dynamic nature of the robotic system, we consider as approximating architecture a neural network of the recurrent kind (RNN), i.e. a structure endowed with memory (see Goodfellow et al. 2016 for a detailed description of these models). The generic mathematical form is given by

$$u(t) = g(h(t), \theta_1) \quad (1)$$

where θ_1 is a set of parameters to be optimized, g is the output function (e.g., a linear function), and $h(t)$ is an internal state vector (usually called "hidden state") that keeps

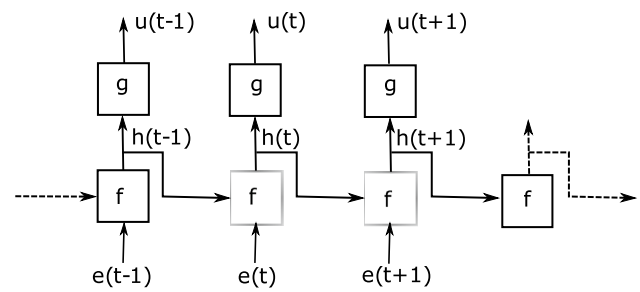


Fig. 4 Graphical representation of the RNN structure implementing the proposed controller

track of the evolution of the input, thus, implementing the memory mechanism. In particular, we have

$$h(t) = f(h(t-1), e(t), \theta_2) \quad (2)$$

where f is the state equation ruling the internal state and θ_2 is the set of parameters devoted to the internal state evolution. In general, a RNN typically receives as input a regressor that contains the last l states, i.e., $\delta_l(t) = [e(t), \dots, e(t-l+1)]^T$, and applies recursively the equations (1) and (2) l times determining the output $u(t)$. Figure 4 provides a graphical representation of a RNN as described by equations (1) and (2).

The optimization of the sets of parameters θ_1 and θ_2 corresponds to the training procedure of the approximating architecture. This operation relies on the minimization of an empirical cost defined on the basis of the observed (error/BSP control) pairs from the successful trajectories. More specifically, we define the following mean squared error

$$J(\theta_1, \theta_2) = \sum_{j=1}^N \sum_{t=0}^{T_j-1} \|u_j(t) - u_j^*(t)\|^2,$$

where $u_j(t)$ is the output of the RNN in correspondence to the regressor $\delta_l^{(j)}(t) = [e_j(t), \dots, e_j(t-l+1)]^T$. The optimal parameters θ_1^* and θ_2^* are determined by minimizing the cost J ; this can be done using standard training procedures relying on the well-known backpropagation algorithm applied to the RNN structure (Goodfellow et al. 2016).

For the imitation of the BSP scheme we consider deep RNNs, i.e., neural architectures obtained by a cascade of recurrent layers. However, we employ a more sophisticated version of the "naive" RNN presented in equations (1) and (2); it is in fact known that such neural networks suffer from some structural problems, such as the vanishing/exploding gradient (Goodfellow et al. 2016). Therefore, in order to implement the recurrent deep architecture, we make use of multiple layers of the popular Gated Recurrent Unit (GRU) (Cho et al. 2014) with a linear output layer, which

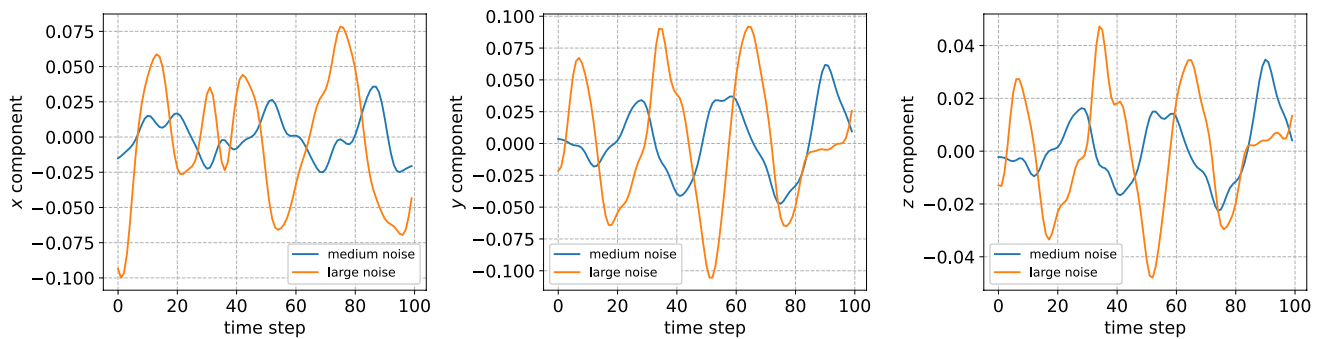


Fig. 5 Samples of generated noise for $w = 0.035$ and $w = 0.05$

is more robust in the training phase with respect to simple RNNs.

The parts of the standard BSP whose behavior is imitated by the neural controllers (CDL and MDL) are highlighted in the flowchart of Fig. 3.

5 Experiments

The imitation learning strategies described above were implemented and tested on H₂Arm. The tests involved the execution of a visual-servoing task consisting in centering a red blob in the image plane, acquiring 3D information on the error through a camera. This task was chosen to be the same as in Bonsignorio and Zereik (2021), in such a way to have a basis for comparison. For details about H₂Arm hardware and the software implementation of the task, refer to Sect. 3.1.

Our tests were designed to evaluate the performance of the deep neural controller (introduced in Sect. 4) when used in place of the BSP-based one. Both kinds of neural control methods described in Sect. 4 (i.e. Cartesian and motor) were considered. The tests also involved additional noise injected in the system in such a way to evaluate system robustness. In the remainder of the section we provide a detailed description of the setup for the testing campaign, and present the main results of the experiments, together with some discussion.

5.1 Experimental setup

The neural controllers were trained using data collected from $N = 97$ successful trajectories generated by the BSP approach. The training trajectories always started with the red blob in view (totally or partially) in two of the four corners of the image, namely the top right corner - denoted as ‘Ref1’ - and the bottom left corner - denoted as ‘Ref2’ (the top left and bottom right corners are denoted as ‘Ref3’ and

‘Ref4’, respectively). The relative position between the red blob and the arm wrist, estimated by the robot camera, was used (as the only available information about the arm end-effector) to train the neural controllers, together with the corresponding commands sent to motors. As said, a noise was injected into the system, more specifically added to the end-effector position $r(t)$ estimated by the camera at time t . In particular, since our long-term objective is underwater manipulation, the generated noise is distributed according to the Pierson-Moscowitz spectrum (Pierson and Moskowitz 1964). According to this formulation, the sea elevation is expressed as

$$\zeta = \sum_{j=1}^W A_j \sin(\omega_j t - k_j x + \epsilon_j) \quad (3)$$

where for each wave $j = 1, \dots, W$, A_j is the amplitude, ω_j is the angular frequency, k_j is the wave number and ϵ_j is a random phase angle; x is the wave location from the origin. Angles ϵ_j are constant and uniformly distributed in the range $[0, 2\pi]$; for deep water, the relation $k_j = \frac{\omega_j^2}{g}$ holds. In these experiments, the additional noise generated at each step accordingly to the Pierson-Moscowitz spectrum was of the form $n = w\zeta$ along each considered axis, where w is a scaling factor. We tested our system versus two different noise magnitudes, namely $w = 0.035$ and $w = 0.05$; an example of the temporal behavior of such noise for each scale factor is shown in Fig. 5.

The scale factor values were chosen in light of the standard BSP experimental campaigns previously conducted and detailed in Bonsignorio and Zereik (2021). The BSP training trajectories have been executed both with or without additional noise injected into the system. In particular, 34 trajectories were executed without noise, and the remaining 63 were executed with added noise as described above. Note that in the training trajectories different scale factors

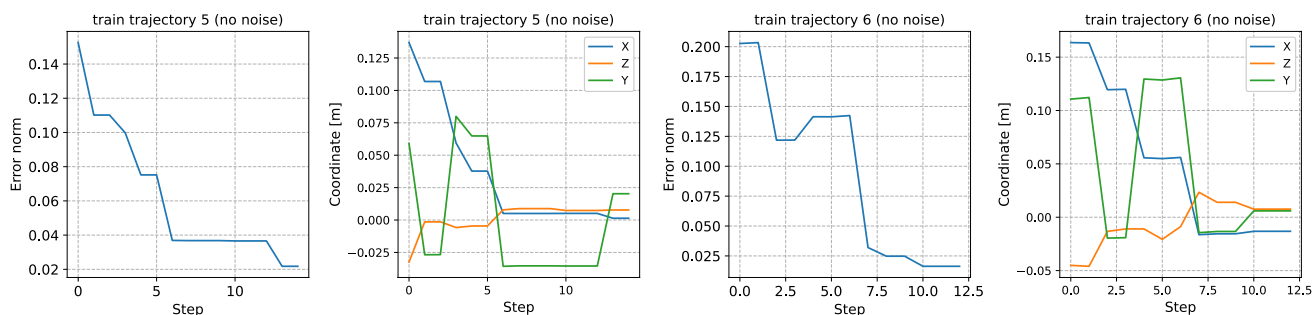


Fig. 6 Examples of BSP training trajectories (without injected noise)

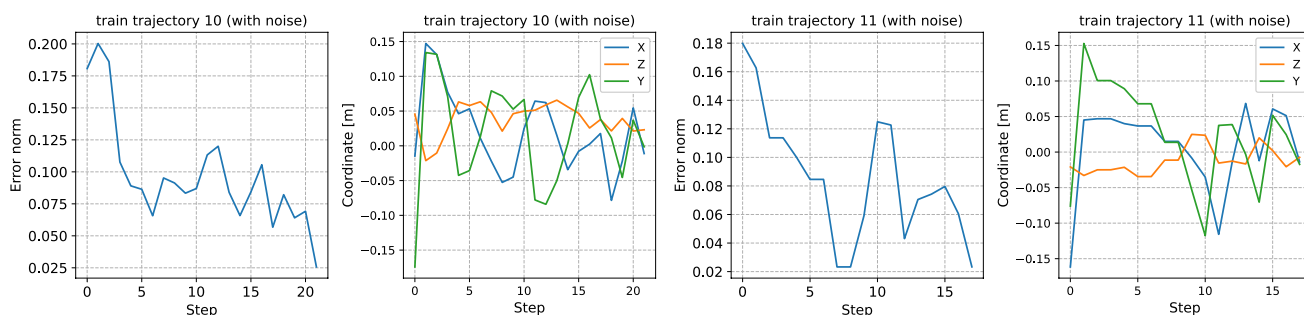


Fig. 7 Examples of BSP training trajectories (with injected noise)

w were tested (for details please consult Bonsignorio and Zerek 2021). Figures 6 and 7 illustrate some examples of BSP training trajectories with and without injected noise.

Concerning the deep models training process, the parameter l , as explained in Sect. 4, was chosen equal to 3, thus the N trajectories yielded a total number $N^* = 1801$ of input/output pairs used to train the neural models.

The deep neural models employ GRU neural units, as said in Sect. 4, for the recurrent layers. In particular, for the Cartesian control we used a network with 2 layers of 50 GRU units, while for the motor control, due to the more complex dynamics and larger output, we employed 5 layers of 500 GRU units. The network parameters have been optimized with the ADAM algorithm (Goodfellow et al. 2016), using batches made of 16 samples. All the network models have been implemented with PyTorch².

For each neural controller, 20 tests were performed without noise injection and 20 with noise injection with $w = 0.035$. Each set of 20 experiments was equally distributed among the 4 different initial conditions (i.e., red blob initially in view in all the 4 corners of the image). Hence, a total number of 80 test experiments was performed. The test starting points in ‘Ref1’ and ‘Ref2’ are always different with respect to those of the training trajectories. In general,

“starting from Ref n ” actually means “starting from a random point in the corner area Ref n ”. Concerning the tests starting from ‘Ref3’ and ‘Ref4’ (i.e., the corners never used during the generation from the training trajectories), they have been included to evaluate the ability of the neural controller to learn and extrapolate, not only to imitate. After these tests, we conducted some in-depth analysis on the MDL controller, focusing on: i) the effect of varying the value of the neural controller parameter l ; ii) the effect of a larger noise ($w = 0.05$) acting on the system.

To evaluate the performance of the neural controllers, we employ the norm of the Cartesian error (expressed in [m]) between H₂Arm end-effector current position and its desired final position (right in front of the red blob). This is used to compute two main performance indices:

- the number of motion steps required to reach the goal (here represented by the error threshold level $e^* = 0.03$ m), starting from an initial common distance (here set to 0.18 m);
- the actual execution time (in seconds) required to reach the goal as above.

² <http://www.pytorch.org>.

Table 1 Comparison of the mean number of steps, the mean execution time and the mean time-per-step between the training and test trajectories for both neural controllers, both with and without additional noise. Note that the training trajectories correspond to the 97 successful BSP experiments described in Bonsignorio and Zereik (2021)

Test trajectories				
Neural Controller	Noise	Average steps	Average time [s]	Av. mean time/step [s]
CDL	No	18.25	6.98	0.38
MDL	No	19.45	7.53	0.39
CDL	Yes	37.1	14.66	0.39
MDL	Yes	31.7	12.33	0.38
Training trajectories				
Neural Controller	Noise	Average steps	Average time [s]	Av. mean time/step [s]
CDL	No	15.74	73.83	5.29
MDL	No			
CDL	Yes	13.91	91.08	8.55
MDL	Yes			

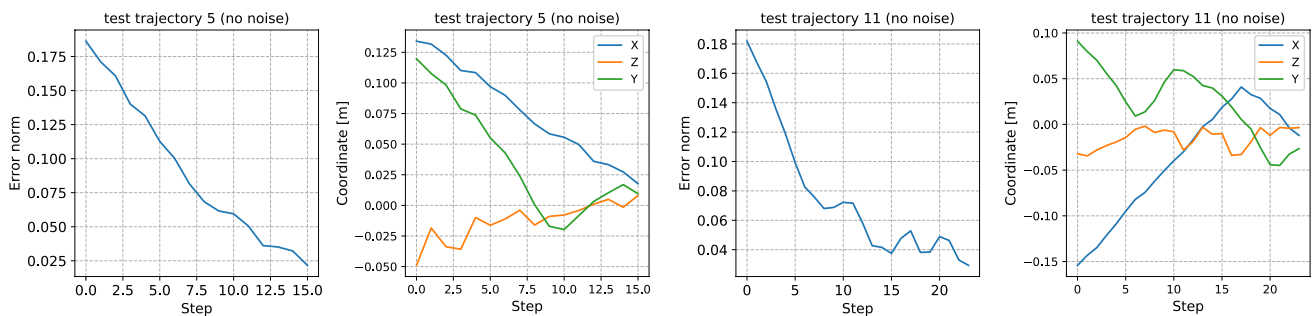


Fig. 8 Norm of the error and corresponding component-wise behavior in two example trajectories - Cartesian control without injected noise. Notice that trajectory 5 starts from Ref1, while trajectory 11 starts from Ref3

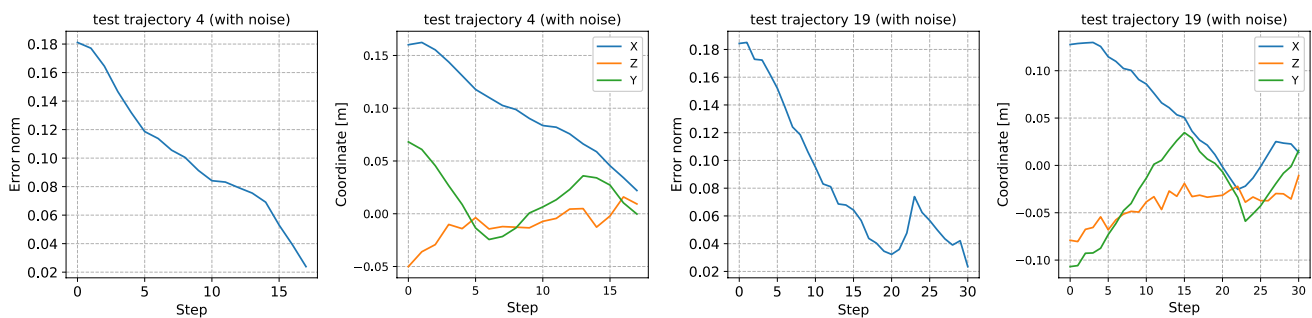


Fig. 9 Norm of the error and corresponding component-wise behavior in two example trajectories - Cartesian control with injected noise. Notice that trajectory 4 starts from Ref1, while trajectory 19 starts from Ref4

5.2 Results

Both the Cartesian and motor controls allowed to obtain 100% success rate in all tests, with and without injected noise. Table 1 reports a summary at a glance of the obtained results. In particular, it reports the average number of steps and the average time needed to reach the goal over the above-mentioned sets of 20 test trajectories for both the

Cartesian and the motor deep-learning controllers, both with and without noise. The mean time-per-step of the single trajectory, averaged over the set of test trajectories, is also reported. For a comparison, the average number of steps, the average execution time and the mean time-per-step are reported in the table also for the 97 trajectories used for the training, obtained with the BSP algorithm.

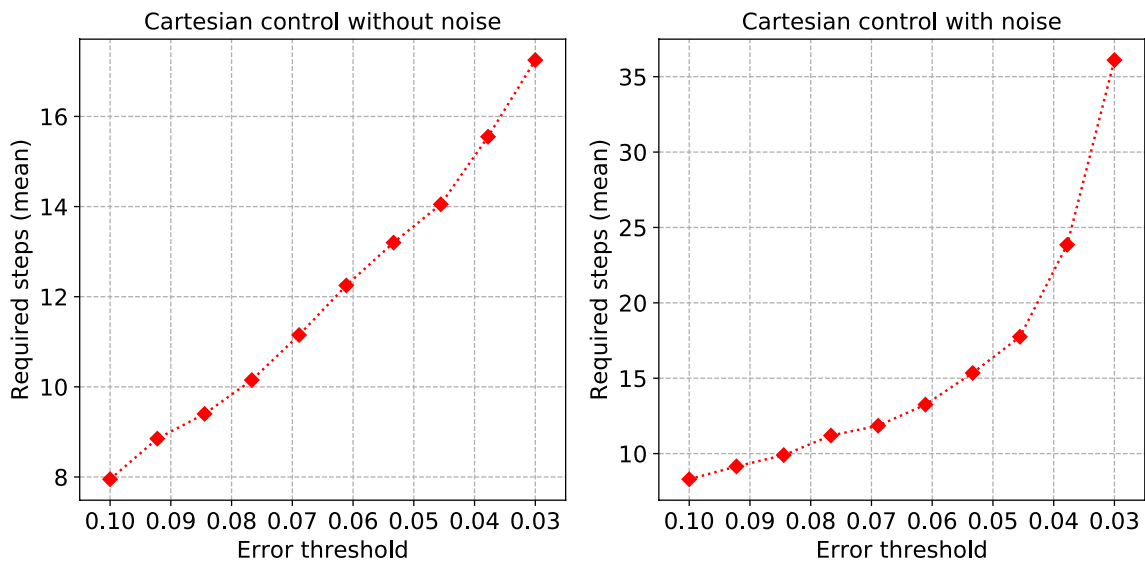


Fig. 10 Mean over the 20 test trajectories of the number of steps required to reach decreasing error levels, without (left) and with (right) injected noise - Cartesian control

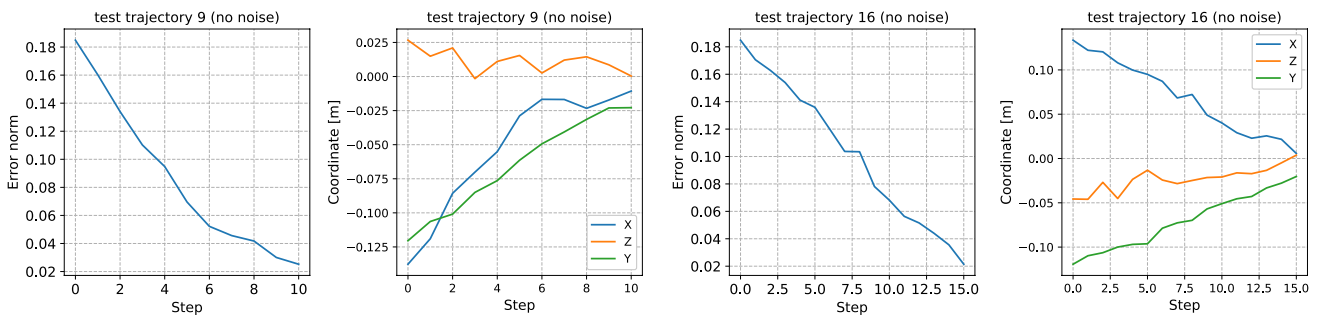


Fig. 11 Norm of the error and corresponding component-wise behavior in two example trajectories - motor control without injected noise. Notice that trajectory 9 starts from Ref1, while trajectory 16 starts from Ref4

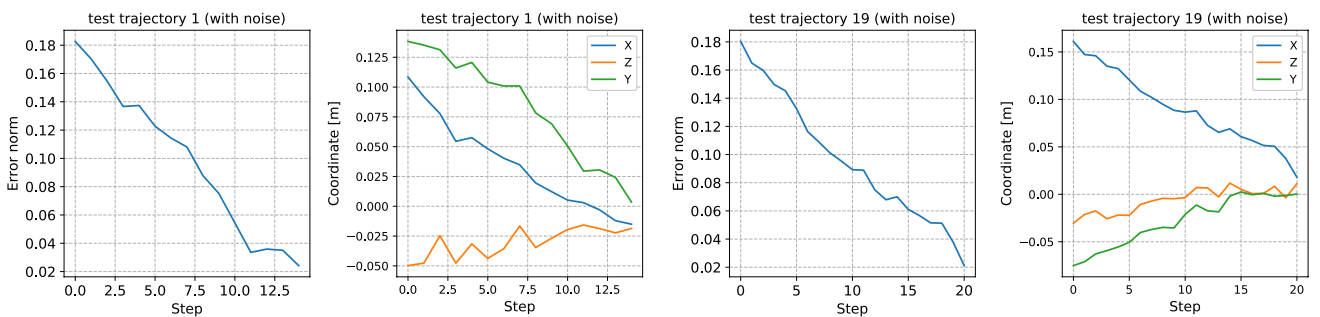


Fig. 12 Norm of the error and corresponding component-wise behavior in two example trajectories - motor control with injected noise. Notice that trajectory 1 starts from Ref2, while trajectory 19 starts from Ref4

Overall, the table shows how the proposed methodology proved to be quite effective in providing a controller able to successfully execute the task. In fact, we can see that, even if

it takes a few more steps, the neural controller always yields convergence almost 10 times faster than the original BSP algorithm in terms of execution time.

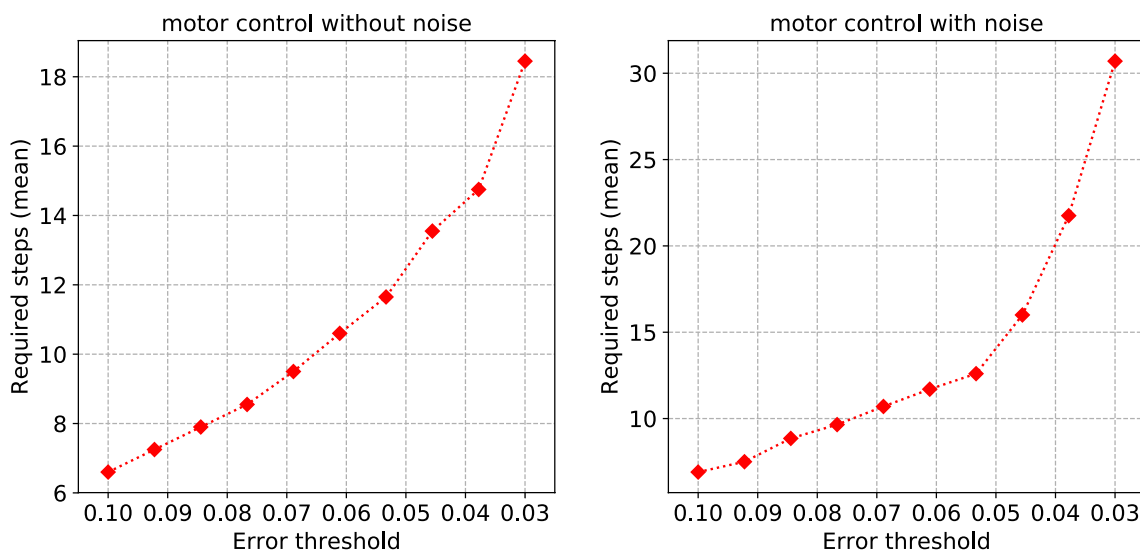


Fig. 13 Mean over the 20 test trajectories of the number of steps required to reach decreasing error levels, without (left) and with (right) injected noise - motor control

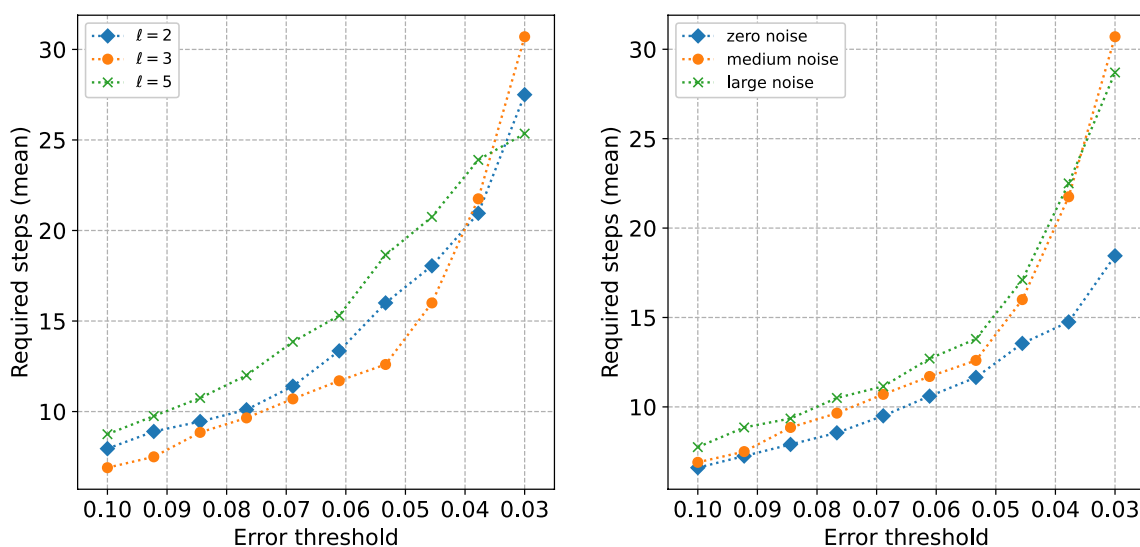


Fig. 14 System behaviour while facing variation in: neural controller parameter l (left) and additional noise magnitude (right)

Concerning the Cartesian control, Figs. 8 and 9 report the norm and corresponding component-wise behavior of the error in two example trajectories, without and with injected noise, respectively.

Figure 10 illustrates the mean (over the 20 test trajectories) number of steps required to reach decreasing error levels, both for the noise-free and the noisy tests. These graphs show how relaxing the final error threshold requirement (e.g., setting it to 0.05 m) the system would reach the goal in a considerably smaller number of steps.

Regarding the motor control, Figures 11 and 12 report the norm and corresponding component-wise behavior of the error in two example trajectories, without and with injected noise, respectively.³

Figure 13 illustrates also for this control scheme the mean (over the 20 test trajectories) number of steps required to reach decreasing error levels, both for the noise-free and the noisy tests.

³ Please note that all the trajectories (and the corresponding error norm) of the 80 performed test experiments are reported in a Supplemental Material file to this article.

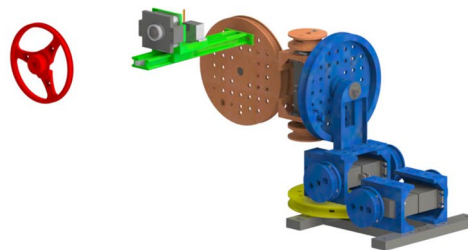
(a) DeepMind Reacher(b) H₂Arm visual-servoing

Fig. 15 Comparison among the proposed H₂Arm visual-servoing task and the “Reacher” task of the DeepMind Control Suite. The main characteristics of each task are: a) *DeepMind Reacher* – a 2-link planar structure that has to reach a target, executed only in simulation, with known proprioceptive measures, known target location, simulated scenario with known noise structure, many training data needed, AI directly on image pixels. Tasks are strongly observable, position and velocity observations depend only on the current state. Sensor readings only depend on the previous transition, see Tassa

et al. (2018). Courtesy of DeepMind. b) *H₂Arm* – 4-link 3D manipulator, experimented in real world, without proprioceptive information (the only sensor on-board is the wrist-mounted camera), additional noise injected in some of the experiments, very few needed training data (97 BSP trajectories logged in previous tests where the arm was controlled by the BSP algorithm only, without any neural controller), images are pre-processed by a vision algorithm and AI works on measures estimated by vision

5.3 Additional experiments

After the first test campaign, a series of ad-hoc additional tests have been performed to investigate the robustness of the controller. In particular, we investigated how changing the value of parameter l in the neural controller affects the system behavior, as well as testing the performance of the controller in presence of larger noise. For these tests we focused on the MDL scheme, which is the most interesting one. In fact, it does not require the pseudo-inversion of the robot Jacobian matrix that, in case of more complex robots, can generally be of large size or, in case of soft and compliant robots, could be highly inaccurate. In the first round of additional experiments we performed the same tests as in the previous experimental campaign using $l = 2$ and $l = 5$. For each parameter value, 40 experimental runs were performed, 20 with and 20 without additional noise, and with the blob starting position equally distributed among the 4 image corners. Performance of the system when using $l = 2$ and $l = 5$ are reported in Fig. 14 (left); the system behavior is quite similar in all the cases, with a slight improvement in the case $l = 3$, showing that the parameter l is not a crucial factor for the success of the method, which is 100% in all cases.

Then we conducted a last case study aimed at assessing the MDL system behavior while facing additional noise with a larger magnitude. Thus, 20 more runs (5 with the red blob initial position in each image corner) were performed injecting in the system the already described Pierson-Moscowitz

distributed spectrum with $w = 0.05$. Success rate is again 100%; the system behavior is depicted in Fig. 14 (right). From the plots, it is clear how the system behavior remains quite the same when facing a larger noise (the number of required steps is slightly higher but the difference is very small), thus underlining the system robustness to environmental disturbance.

5.4 Discussion

All the performed tests were successful and had quite good performance in terms of convergence and execution time. Regarding the CDL-BSP, all 20 test trajectories without noise are such that the goal is reached in few steps; only in some cases it shows some oscillations. These oscillations are mainly due to the camera estimation error and, in any case, they do not prevent task completion, even if the threshold below which the task is considered completed is quite low (0.03 m). Not surprisingly, the most oscillating component is the one along axis z of frame $\langle c \rangle$, namely the distance of the end-effector from the red blob along the camera optical axis. The trajectories with noise exhibit more severe oscillations, and such behavior is again mostly due to the noise and the estimation error on axis z of camera frame $\langle c \rangle$. The MDL-BSP scheme turns out to yield similar performance, with respect to the Cartesian one, in the noise-free case; only it takes on average some more steps and some more time (the mean time difference is 0.55 s). Instead, in the noisy case

the motor deep-learning controller employs, on average, less steps and less time with respect to the CDL-BSP (the mean time difference is 2.33 s).

This difference is probably due to the fact that the CDL controller relies on a classical inverse kinematics approach to achieve the transition between the Cartesian and the joint operative spaces, while the MDL controller directly generates reference velocity for motors, without the need of an inverse kinematics computation. The first case likely implies a higher sensitivity to noise, since this last is filtered by the pseudo-inverse of the Jacobian matrix, while in the MDL case the reference commands are directly generated and sent to motors.

As already hinted, we computed the mean time per each step. As expected, this value is constant for both neural controllers, both with and without noise. On the contrary, it varies in the case of the standard BSP approach. This is due to the difference in terms of execution time between the neural and standard control strategies: the time employed by neural controllers to compute the next command to be sent to the robot is negligible with respect to the motion execution time, whereas the BSP computation time (needed to solve the optimization problem at each sample time) is in general greater than the motion execution time (and potentially even higher in case of replanning).

For the sake of providing an example of comparable task, Figure 15 illustrates the DeepMind “Reacher” task (Tassa et al. 2018) along with our visual-servoing one, highlighting the main differences.

6 Conclusions and future work

The proposed deep neural controllers turned out to be quite effective for the considered visual-servoing blob centering task. DL-BSP allowed to obtain 100% success rate in all our tests both with Cartesian and motor control schemes, proving to be robust towards both camera estimation errors (above all on depth z component wrt frame $\langle c \rangle$) and additional Pierson-Moskowitz noise directly injected on the red blob 3D position measurement at each sample time. This confirms that the combination of H₂Arm structure and the proposed control schemes is a good candidate for the control of light inaccurate arms. Both controllers exhibited the capability to extrapolate the tasks learned by imitation. In fact, in all the trajectories starting in the Ref3 and Ref4 corners, the arm end-effector always reached the desired final position in front of the red blob, even if no trajectory starting in those corners was provided in the training set. Looking at Table 1, we can notice that the controller trained on the Cartesian dataset is faster than the one trained on the motor velocity dataset in tests without noise, while the latter is faster in tests with noise; the same alternate behavior can

be reported analysing the required steps for both controllers (the Cartesian control scheme needs less steps than the motor one for the case without additional noise, while it takes more steps when additional noise is injected into the system). The performance of both controllers slightly deteriorates when additional noise is injected into the system, but the times remain remarkably smaller than the standard BSP control scheme behavior.

The motor control strategy, by directly generating commands to the arm motors, can result more efficient in terms of online computational burden, since it does not require the pseudo-inversion of the robot Jacobian matrix, as already hinted. Overall, in our experiments the deep learning imitation approach allowed to retain the optimality and strong robustness properties of the BSP algorithm, while allowing fast execution times that make it actually applicable to real-time operations. Thus, it appears to be a quite promising option to enable the successful operation of low-cost, modular, inaccurate and non-repeatable robotic structures such as H₂Arm.

Moreover, according to preliminary experiments that we additionally performed by manually displacing the red blob in the scene, we expect that the same approach will work for moving targets too, only relying on the same dataset and without further training. In fact, the system showed good performance also in such preliminary tests, whose results are comparable to the previously conducted experiments.

Future work will be aimed at performing more complex tasks, investigate the robustness to different training dataset, testing in unstructured environments (and marine contexts in particular) also comparing against more advanced imitation learning schemes (e.g., inverse reinforcement learning). From an experimental point of view, the middle-to-long term objective of the present research activity will focus on testing the proposed control strategy on a marinized manipulator (possibly with a more accurate structure allowing for ground-truth) in a relevant underwater environment, to compare system performance.

An interesting issue is to get a better understanding of the reduced training effort of the imitation deep learning network when the training sets are obtained through an application of an optimized algorithm such as the belief space planning (whether we consider the Cartesian or motor velocities approach). We have implemented our learning and control strategy on an open-source real-world arm performing a task similar to that of the ‘reacher’ scenario in Tassa et al. (2018) with a more complex robot on a very low-cost low-accuracy platform with no proprioceptive sensors and very basically actuated motors. We will investigate if and how our approach can be scaled to more complex tasks, in order to determine the real reach of the proposed methodology.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s41315-022-00262-y>.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

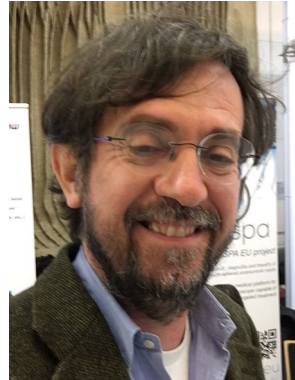
References

- Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the Twenty-First International Conference on Machine Learning, pp. 1–8 (2004)
- Agha-Mohammadi, A.A., Chakravorty, S., Amato, N.M.: Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research* **33**, 268–304 (2014)
- Bain, M., Sammut, C.: A framework for behavioural cloning. In: Furu-kawa, K., Michie, D., Muggleton, S. (eds.) *Machine Intelligence* vol. 15, pp. 813–816 (1999)
- Betts, J.T.: Practical methods for optimal control and estimation using nonlinear programming, vol. 19, pp. 132–134 (2010)
- Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to End Learning for Self-Driving Cars (2016)
- Bonsignorio, F.: Quantifying the evolutionary self-structuring of embodied cognitive networks. *Artif. Life* **19**(2), 267–289 (2013)
- Bonsignorio, F., Zereik, E.: A simple visual-servoing task on a low-accuracy, low-cost arm: an experimental comparison between belief space planning and proportional-integral-derivative controllers. *IEEE Robot. Autom. Mag.* **28**(3), 117–127 (2021)
- Bonsignorio, F., Hsu, D., Johnson-Roberson, M., Kober, J.: Deep learning and machine learning in robotics [From the Guest Editors], Special Issue. *IEEE Robot. Autom. Mag.* **27**(2), 20–21 (2020)
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734 (2014)
- Ebert, F., Dasari, S., Lee, A.X., Levine, S., Finn, C.: Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. arXiv preprint [arXiv:1810.03043](https://arxiv.org/abs/1810.03043) (2018)
- Finn, C., Levine, S., Abbeel, P.: Guided cost learning: Deep inverse optimal control via policy optimization. In: Proceedings of the 33rd International Conference on Machine Learning - Volume 48, pp. 49–58 (2016)
- Giusti, A., Guzzi, J., Cireşan, D.C., He, F., Rodríguez, J.P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G.D., Scaramuzza, D., Gambardella, L.M.: A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics Autom. Lett.* **1**(2), 661–667 (2016)
- Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, (2016)
- Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3389–3396 (2017). IEEE
- Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Advances in Neural Information Processing Systems 29, pp. 4565–4573 (2016)
- Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **32**, 1238–1274 (2013)
- Macciò, D.: Local linear regression for efficient data-driven control. *Knowl.-Based Syst.* **98**, 55–67 (2016)
- Macciò, D., Cervellera, C.: Local Models for data-driven learning of control policies for complex systems. *Expert Syst. Appl.* **39**(18), 13399–13408 (2012)
- Marchese, A.D., Katzschmann, R.K., Rus, D.: Whole arm planning for a soft and highly compliant 2d robotic manipulator. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 554–560 (2014). IEEE
- Ng, A.Y., Russell, S.J.: Algorithms for inverse reinforcement learning. In: Proceedings of the 70th International Conference on Machine Learning, pp. 663–670 (2000)
- Odetti, A., Bibuli, M., Bruzzone, G., Cervellera, C., Ferretti, R., Gaggero, M., Zereik, E., Caccia, M.: A preliminary experiment combining marine robotics and citizenship engagement using imitation learning. 21st IFAC Proceedings Volumes (2020)
- Okur, B., Aksoy, O., Zergeroglu, E., Tatlicioglu, E.: Nonlinear robust control of tendon-driven robot manipulators. *J. Intell. Robot. Syst.* **80**(1), 3–14 (2015)
- OpenAI, T.: OpenAI Gym Website. (2020). Accessed 2022/09/12 07:05:00. <https://gym.openai.com>
- Pierson, W.J., Moskowitz, L.: A proposed spectral form for fully developed wind seas based on the similarity theory of SA Kitaigorodskii. *J. Geophys. Res.* **69**(24), 5181–5190 (1964)
- Platt, R., Tedrake, R., Kaelbling, L., Lozano-Perez, T.: Belief space planning assuming maximum likelihood observations. In: Proceedings of Robotics: Science and Systems, Zaragoza, Spain (2010). <https://doi.org/10.15607/RSS.2010.VI.037>
- Pololu, M.: Pololu Drivers. (2020). Accessed on 2022/09/12 07:05:00. <https://www.pololu.com/docs/OJ40/>
- Pomerleau, D.A.: ALVINN: an autonomous land vehicle in a neural network, pp. 305–313 (1989)
- Rahmatizadeh, R., Abolghasemi, P., Bölöni, L., Levine, S.: Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 3758–3765 (2018). IEEE
- Rolf, M., Neumann, K., Queißer, J.F., Reinhart, R.F., Nordmann, A., Steil, J.J.: A multi-level control architecture for the bionic handling assistant. *Adv. Robot.* **29**(13), 847–859 (2015)
- Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, vol. 15, pp. 627–635 (2011)
- Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., Hebert, M.: Learning Monocular Reactive UAV Control in Cluttered Natural Environments (2012)
- Rus, D., Tolley, M.T.: Design, fabrication and control of soft robots. *Nature* **521**(7553), 467–475 (2015)
- Rusu, A.A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R., Hadsell, R.: Sim-to-real robot learning from pixels with progressive nets.

Accessed 2022/09/12 07:05:00(2016). <https://arxiv.org/abs/1610.04286>

- Sammur, C., Hurst, S., Kedzier, D., Michie, D.: Learning to fly. In: Proceedings of the 9th International Workshop on Machine Learning, pp. 385–393 (1992)
- Seita, D., Ganapathi, A., Hoque, R., Hwang, M., Cen, E., Tanwani, A.K., Balakrishna, A., Thananjeyan, B., Ichnowski, J., Jamali, N., Yamane, K., Iba, S., Canny, J., Goldberg, K.: Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 9651–9658 (2020). <https://doi.org/10.1109/IROS45743.2020.9341608>
- Srinivas, A., Laskin, M., Abbeel, P.: CURL: Contrastive Unsupervised Representations for Reinforcement Learning. Accessed 2022/09/12 07:05:00(2020). <https://arxiv.org/abs/2004.04136>
- Stoelen, M.F., Bonsignorio, F., Cangelosi, A.: Co-exploring actuator antagonism and bio-inspired control in a printable robot arm. In: International Conference on Simulation of Adaptive Behavior, pp. 244–255 (2016). Springer
- Subudhi, B., Morris, A.S.: Soft computing methods applied to the control of a flexible robot manipulator. *Appl. Soft Comput.* **9**(1), 149–158 (2009)
- Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., Riedmiller, M.: DeepMind Control Suite. Accessed 2022/09/12 07:05:00(2018). <https://arxiv.org/abs/1801.00690>
- Thuruthel, T.G., Falotico, E., Renda, F., Laschi, C.: Learning dynamic models for open loop predictive control of soft robotic manipulators. *Bioinspiration Biomim.* **12**(6), 066003 (2017)
- Wang, H., Yang, B., Liu, Y., Chen, W., Liang, X., Pfeifer, R.: Visual servoing of soft robot manipulator in constrained environments with an adaptive controller. *IEEE/ASME Trans. Mechatron.* **22**(1), 41–50 (2016)
- Wulfmeier, M., Ondruska, P., Posner, I.: Maximum Entropy Deep Inverse Reinforcement Learning (2015)
- Xie, X., Cheng, L., Hou, Z., Ji, C.: Adaptive neural network control of a 5 dof robot manipulator. In: 2010 International Conference on Intelligent Control and Information Processing, pp. 376–381 (2010). IEEE
- Zereik, E., Gagliardi, F., Bibuli, M., Sorbara, A., Bruzzone, G., Caccia, M., Bonsignorio, F.: Belief space planning for an underwater floating manipulator. In: Moreno-Diaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) Computer Aided Systems Theory, EUROCAST 2015: 15th International Conference, LNCS 9520, pp. 869–876 (2015)
- Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., Abbeel, P.: Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–8 (2018). IEEE
- Zhu, Y., Wang, Z., Merel, J., Rusu, A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramár, J., Hadsell, R., de Freitas, N., et al.: Reinforcement and imitation learning for diverse visuomotor skills. arXiv preprint [arXiv:1802.09564](https://arxiv.org/abs/1802.09564) (2018)
- Ziebart, B.D., Maas, A., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. In: Proc. AAAI, pp. 1433–1438 (2008)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Fabio Bonsignorio is the ERA Chair in AI for Robotics and head of the AIFORS Lab, in LAMOR at FER at the University of Zagreb. AIFOR lab aims to take intelligent soft robotics to the next stage. He is CEO and Founder of Heron Robots (advanced robotic solutions), see www.heronrobots.com. He has been Visiting Professor at the Biorobotics Institute of the Scuola Superiore Sant'Anna in Pisa in the period 2014–2019. He has been professor in the Department of System Engineering

and Automation of the University Carlos III of Madrid until 2014. In 2009 he was awarded the Banco de Santander Chair of Excellence in Robotics at the same university. He is a Founding Director of euRobotics aisbl, the private part of SPARC, and now part of ADRA. He is a past elected member of the Research Board of Directors of SPARC. He coordinated and has been the main teacher of the Shanghai Lectures (www.shanghailectures.org) since the 2013 edition. The Shanghai Lectures are an advanced network MOOC teaching initiated several years ago by Rolf Pfeifer. He has pioneered and introduced the topic of Reproducible Research and Benchmarking in Robotics and AI, where is one of the leading experts, if not the leading one. He is a pioneer in the applications of Blockchain technologies in Robotics.



Cristiano Cervellera received the M.Sc. Degree in electronic engineering and the Ph.D. Degree in electronic engineering and computer science from the University of Genoa, Genoa, Italy, in 1998 and 2002, respectively. He is currently a senior researcher at the Genoa branch of the Institute of Marine Engineering of the Italian National Research Council, Genoa, Italy. He is author of several works in journals and international conferences in the fields of machine learning, intelligent systems, operations

research and automation. His current research interests include computational intelligence methods for decision systems, robotics, approximate dynamic programming, optimal control, number-theoretic methods for optimization and learning, intelligent transportation systems.



Danilo Macciò received the M.Sc. degree in telecommunication engineering and the Ph.D. degree in mathematical engineering from the University of Genoa, Genoa, Italy, in 2005 and 2009, respectively. He is currently a Researcher with Institute of Marine Engineering, National Research Council of Italy, Genoa. His current research interests include machine learning, optimization of transportation systems, statistical estimation, and numeric solutions of functional optimization prob-

lems. He serves as Associate Editor of the Neurocomputing journal.



Enrica Zereik received the M.Sc. degree in information technology engineering and the Ph.D. degree in electronic and computer engineering, robotics and telecommunications from the University of Genoa, Genoa, Italy, in 2006 and 2010, respectively. She is currently a Researcher with the Genoa branch of the Institute of Marine Engineering (INM), National Research Council of Italy (CNR), focusing her research activity on underwater manipulation, computer vision, advanced

algorithms for navigation guidance and control of marine vehicles and intelligent systems, evaluation indices and metrics for the experimental assessment of marine platforms performance, reproducibility of experiments, and coordination and control algorithms for cooperative multi-robot systems. She is the PI for CNR of the Joint Lab heron@cnr, constituted between CNR-INM and Heron Robots. She has authored one monograph and several papers in international journals and conferences. She serves as Associate Editor for the IEEE Robotics and Automation Magazine.