Self-Configuring Robot Path Planning with Obstacle Avoidance via Deep Reinforcement Learning

Bianca Sangiovanni, Gian Paolo Incremona, Marco Piastra and Antonella Ferrara

***Abstract*—This work proposes a hybrid control methodology to achieve full body collision avoidance in anthropomorphic robot manipulators. The proposal improves classical motion planning algorithms by introducing a Deep Reinforcement Learning (DRL) approach trained *ad hoc* for performing obstacle avoidance, while achieving a reaching task in the operative space. More specifically, a switching mechanism is enabled whenever a condition of proximity to the obstacles is met, thus conferring to the dual-mode architecture a self-configuring capability in order to cope with objects unexpectedly invading the workspace. The proposal has been finally tested relying on a realistic robot manipulator simulated in a V-REP environment.**

***Index Terms*—Deep learning, path planning, robot control, collision avoidance.**

I. Introduction

A. Background

As robots are required to perform increasingly demanding tasks in different and complex environments, great focus is put in robotics research to ensure that such systems are able to move and interact with the surroundings without endangering equipment and, most importantly, humans [1]. In fact, in the context of the so-called physical Human-Robot Interaction (pHRI) or in case of robots operating alone in cluttered environments, obstacle avoidance remains a core issue [2], [3].

A number of contributions documents the interest in handling physical interaction between the robots and the environment. Artificial potential fields, introduced in [4], detect obstacles in the proximity and, based on that, generates an artificial force field that is then transformed into joint torques. This approach was later improved in [5], [6], while real-time, adaptive motion planning was instead explored in [7], [8]. Furthermore, among the possible strategies, the majority relies on Optimal Control Problems (OCPs) [9]–[11]. It is worth highlighting that, in order to implement potential fields methods, the dynamical model of the manipulator must be known or accurately estimated, whereas planning methods require constant sampling and online planning of the trajectory in order to find an obstacle free path. Both requirements might be hard to satisfy in presence of multiple moving obstacles or unknown environment.

A recent alternative to the mentioned methods is instead given by Reinforcement Learning (RL) approaches. Specifically, Deep Reinforcement Learning (DRL), following the work in [12], has been researched as a promising approach for solving hard to engineer tasks, as in cases where it is difficult to have an accurate enough model description of the considered system. By letting the robot interact with its environment, for any given control operation, it is possible to find an optimal strategy to successfully conclude the task [13], [14]. In [15], [16], for instance, such algorithms have been used to train the agent to grasp sparse objects via convolution neural networks for pose estimation. In [17], and then in [18], data efficient methodologies for dexterity operations have been applied. In [19], further improvements have been made for enabling a robot to accomplish a stacking task by using soft Q-learning. Although RL approaches have been successfully applied to systems with discrete action space, many real-world applications require an agent to select optimal actions from continuous spaces. Indeed, discrete actions could not be adequate for devising strategies where a very small change in an action can significantly affect the outcome. This motivates recent researches dealing with DRL adaptation to systems featuring continuous state and action spaces. Specifically, continuous variants of Q-learning are Deep Deterministic Policy Gradient (DDPG) [20] and Normalized Advantage Function (NAF) [21], which have also been employed for end-to-end control of robotics systems. Yet, despite the recent improvements, DRL for robotics systems raises concerns over safety issues: long training times are required, and a high level of domain randomization is needed to prevent undesirable effects and unexpected behaviors. For this reason, the majority of the research in the field is carried out in simulation [22]. Therefore, despite the recognized potential, it is still difficult and expensive to achieve near-optimal behaviors using fully autonomous, end-to-end control strategies with DRL.

B. Contribution

The goal of this paper is to present a hybrid dual-mode architecture which enables the use of motion planning without the burden of taking obstacles into account, and only use an end-to-end DRL-based control strategy once more complex tasks are required. In the proposed case study, with respect to the previous research on collision avoidance discussed in [23], the full body end-to-end collision avoidance approach is extended to a more general scenario with obstacles moving

This is the final version of the accepted paper submitted to IEEE Control Systems Letters. B. Sangiovanni, M. Piastra and A. Ferrara are with Dipartimento di Ingegneria Industriale e dell'Informazione, University of Pavia, 27100 Pavia, Italy (e-mail: bianca.sangiovanni01@universitadipavia.it, marco.piastra@unipv.it, antonella.ferrara@unipv.it).

G. P. Incremona is with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: gianpaolo.incremona@polimi.it).

The authors gratefully acknowledge the full help provided by the master student Nikolas Sacchi for the simulations.

in multiple directions, exploiting experience transfer during training. Our intention is to remove the need for prolonged training to enable the robot to reach a point in space and reduce the level of uncertainties that come with end-to-end approaches. On the other hand, using DRL for performing the obstacle avoidance removes the need for model-based approaches to ensure that the robot full body does not collide with obstacles, which are hard to hand-engineer. This is convenient when obstacles are not known *a-priori* or we do not have an accurate knowledge of the system dynamics.

II. Model of the Robot and the Interaction Scenario

This work focuses on the situation in which a robot manipulator is requested to move in a populated industrial environment, possibly invaded by obstacles. During the motion, the robot could collide with obstacles, while reaching its target or executing a specific task. In the following, we assume that, apart from the encoder fastened on the robot joints, a vision system is present to detect objects motions with respect to a world frame. Furthermore, the objects move randomly so that physical interaction between the environment and the robot can unexpectedly happen.

A. Robot model

Let us consider an industrial manipulator with an open kinematic chain, and let $q \in \mathbb{R}^n$ be the joint variables related to the motor positions, with n being the number of degrees of freedom (i.e., of joints). Given the end-effector pose vector $x_\mathrm{e} = \begin{bmatrix} p_\mathrm{e} & \Phi_\mathrm{e} \end{bmatrix}^\top \in \mathbb{R}^m$, with m being the dimension of the operative space, p_e being the position and Φ_e the orientation in the workspace, the direct kinematics [24] is indicated as $x_\mathrm{e} = k(q)$, with $k(q) \in \mathbb{R}^m$ being a nonlinear function depending on the joint variables. Since in practice it is convenient to provide joint position and/or velocity references to the internal control loops, the dynamic model of the robot can be expressed as

$$\ddot{q} = f(q, \dot{q}) \;, \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is a function of path coordinates and its time derivatives.

B. Motion planner

Let us concentrate on a general tracking problem in which we assume that the robot moves in the free-motion operative space. The goal is to track a smooth reference trajectory $x_\mathrm{e}^\star$, starting from $x_{\mathrm{e}0}$ towards the target point at x^*. Relying on classical approaches, such as those in [24], motion planning constitutes a crucial aspect, and, in order to find a collision-free trajectory, it can be performed in two ways: i) in the configuration space, by defining motion independently for each joint from an initial configuration q_0 to a final one q^* or ii) in the operative space, by determining $x_\mathrm{e}^\star$. Although planning trajectories in the joint space is simpler and computationally lighter, planning trajectories in the operative space provides a more natural task description and obstacles can be accounted in the design phase of the path. However, if a collision can occur, it entails high computational costs due to the online calculation of the inverse kinematics to take into account the obstacles position. In the following we provide an alternative solution to overcome this problem, proposing a self-configuring path-planning to rapidly cope with obstacles invading the workspace. Moreover, from here on, we assume that local joints controllers are capable to perfectly track the desired joint path, namely $q^\star$, as well as its derivative $\dot{q}^\star(t)$, that is $q(t) = q^\star(t)$ and $\dot{q}(t) = \dot{q}^\star(t)$, $\forall\, t \geq 0$.

C. Collision avoidance problem

Consider now model (1) where $\dot{q}$ is assumed to be the input action, namely a in the following. Furthermore, the workspace strictly depends on the manipulator geometry and the mechanical limits on the joints, so that input and state constraints are of the form $h(q, \dot{q}) \leq 0$ with $h : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^\ell$, and ℓ being the number of constraints. Finally, let $\mathcal{W}(q(t)) \subset \mathbb{R}^m$ be the space occupied by the robot at the instant t, and $\mathcal{O} \subset \mathbb{R}^m$ be the space occupied by the obstacles. Having in mind to execute a possible industrial task (e.g., spot welding, pick and place, or point-to-point motions), our goal is to find a velocity control sequence $\dot{q}^\star$ over the horizon T, which makes the robot end-effector move from the initial state $x_{\mathrm{e}0}$ to a target point x^*, (eventually following a predefined trajectory $x_\mathrm{e}^\star$), while avoiding the obstacles $\mathcal{O} \subset \mathbb{R}^m$ with minimum deviation from the target point and applying minimum control action. Therefore, our finite-horizon Collision Avoidance Optimal Control Problem (CAOCP) is given by

$$\begin{aligned} &\min_{a} \int_0^T c_1 R_\mathrm{T}(x_\mathrm{e}(q), x^*) + c_2 \|\dot{q}\|^2 \,\mathrm{d}t \\ \text{s.t.}\quad & \ddot{q} = f(q, \dot{q}), \quad \dot{q} = a, \quad x_\mathrm{e}(q) = k(q) \\ & h(q, \dot{q}) \leq 0 \\ & \mathcal{W}(q(t)) \cap \mathcal{O} \equiv \varnothing \\ & x_\mathrm{e}(0) = x_{\mathrm{e}0} \end{aligned} \tag{2}$$

where the function $R_\mathrm{T}(x_\mathrm{e}(q), x^*)$ is defined as the following *Huber-Loss function*:

$$R_\mathrm{T} = \begin{cases} \frac{1}{2} d^2 & \text{for } d < \delta \\ \delta \left(d - \frac{1}{2}\delta \right) & \text{otherwise} \end{cases} \tag{3}$$

where $d = \|x_\mathrm{e} - x^*\|$ is the Euclidean distance between the tip and the target and δ is a parameter that determines the smoothness. Moreover, the positive constants c_1 and c_2 are suitably selected weights. The key difficulty in solving (2) is given by the presence of the collision avoidance constraints $\mathcal{W}(q(t)) \cap \mathcal{O} \equiv \varnothing$, which are in general non-convex and non-differentiable.

III. DRL for Collision Avoidance

A. Preliminaries on DRL

The RL framework [25] relies on the concept that an *agent* (i.e., the robot) autonomously learns how to accomplish a given task by iteratively interacting through actions with the environment. At any given time t, the agent observes (i.e., through sensors and cameras) the environment, represented by a certain *state* $s_t \in \mathcal{S}$, with $\mathcal{S}$ being the state space, and according to a certain *policy* $\pi(a|s)$ performs a certain action $a_t \in \mathcal{A}$, with $\mathcal{A}$ being the action space, thus changing the environment's state. When entering a new state, the agent receives a *reward* r_t, that is a scalar feedback on 'how well' the agent has performed. The agent's goal is to maximize, over a window of length T, the expected cumulative reward in the long run, i.e., $R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$, where the term $0 \leq \gamma \leq 1$ is the *discount rate*, depending on how one wants to prioritize early rewards over later ones. To this end, a *value function* V^π is considered, that is the expected cumulative reward the agent can receive in the long run, starting from a given state and following the policy π thereafter. Specifically, the value of a state can be measured in two ways: state-value functions $V^\pi(s)$ and action-value functions $Q^\pi(s,a)$, whether it depends on the state alone or both state and action.

In case of continuous action problems with a large number of states, such as those commonly found in robotic systems, a parametric approximator of the Q-function (Q-learning) is generally used. This learning method allows the convergent, incremental updating of a suitable approximation $\tilde{Q}$ towards the optimal $Q^\star$, as experience progresses. A way to build such approximator is a Deep Neural Network (DNN) [26], i.e., a parametric function that can model complex non-linear relationships. The so-called NAF algorithm proposed in [21] is hereafter used due to its effectiveness for high-dimensional systems with continuous action space (see [20] and [23] for a deeper discussion). In order to find the policy $\tilde{\mu}(s) = \text{argmax}_a \tilde{Q}(s,a)$, the approximator is chosen as $\tilde{Q}\left(s, a|\theta^{(Q)}\right) = \tilde{A}\left(s, a|\theta^{(A)}\right) + \tilde{V}\left(s|\theta^{(V)}\right)$, with $\tilde{A}$ and $\tilde{V}$ being parametric approximators of the *advantage function* $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ and of the value function $V^\pi(s)$, respectively, while $\theta^{(\cdot)}$ are vectors of parameters. Then, the advantage function is selected as the quadratic expression $\tilde{A}\left(s, a|\theta^{(A)}\right) = -0.5\left(a - \tilde{\mu}(s|\theta^{(\mu)})\right)^\top P(s|\theta^{(P)})\left(a - \tilde{\mu}(s|\theta^{(\mu)})\right)$, where P is a state dependent, positive-definite square matrix. Hence, having the Q-function quadratic in the action, by computing its maximum, one can easily achieve the optimal policy given by $a = \tilde{\mu}(s|\theta^{(\mu)})$.

B. Recast of the CAOCP into the RL framework

Making reference to the CAOCP in (2), we want to recast it using the RL framework elements. Assuming to not completely know the robot dynamics, along with the manipulator, the environment contains a target point in the operative space and randomly moving obstacles. Depending on the kind of the considered experiment, the above elements are arranged in different configurations. According to the RL settings, the state space $\mathcal{S}$ is defined as

$$\mathcal{S} = \{q,\, \dot{q},\, p_\text{e},\, p^\star,\, p_\text{o},\, \dot{p}_\text{o}\} \ , \tag{4}$$

where all quantities are vectors, with $p^\star$ being the target point position, p_o being the obstacles positions and $\dot{p}_\text{o}$ their velocity. The action space $\mathcal{A}$ is instead

$$\mathcal{A} = \{\dot{q}^\star\} \,, \tag{5}$$

where $\dot{q}^\star$ is the vector of reference velocities for each joint. An action issued at time t means that the robot must reach the requested velocities in the following time step $t+1$. The reward function is designed as follows

$$r = -(c_1 R_\text{T} + c_2 R_\text{A} + c_3 R_\text{O}) \ , \tag{6}$$

where the minimization equivalent to the CAOCP is achieved by maximizing the reward r, where a negative sign appears differently from (2). Specifically, R_T is defined as in (3), and R_A is a regularization term, computed as follows

$$R_\text{A} = \|a\|^2 \ , \tag{7}$$

which has the purpose of encouraging smaller control actions. Finally, letting d_o be the minimum distance between the whole robot body and the obstacles, R_O is defined as

$$R_\text{O} = \min\left(k_r \frac{R_\text{T}}{c_2} \,;\, \left(\frac{d^\star}{d_\text{o} + d^\star}\right)^g\right) \tag{8}$$

where $d^\star$ is a constant parameter ensuring that $0 < R_\text{O} < 1$, and g is a hyperparameter aimed at increasing the decay rate of the reward component when the robot is far from the obstacles [23]. Moreover, k_r is a positive integer that ensures a minimum, fixed proportion between R_O and R_T. The relative weights of the three terms above can be tuned via the constant parameters c_1, c_2 and c_3. In the context of machine learning there are predefined parameters, called *hyperparameters*. In practice, these values are determined through a preliminary search activity aiming to identify the most effective combination. We refer to [23] for more details.

C. DRL algorithm with experience replay

In this section the so-called *transfer learning* is discussed. This concept generally refers to transferring knowledge acquired in a particular training scenario to a different, possibly more complex one. This means that if tasks change, although the training procedure should be repeated, it can be facilitated by previously acquired experience. Using the NAF algorithm, there are two different possibilities to perform transfer learning: i) *model transfer*, that is reusing the learned parameters of the action-value function as initializers for the action-value function in the subsequent training activity, or ii) *experience transfer*, that is reusing the set of collected quadruplets $\{s_t, a_t, r_t, s_{t+1}\}$ as initializers for an *experience replay buffer*, which collects all the samples produced by the training process throughout the episodes.

The evaluation of the overall approach with NAF algorithm for obstacle avoidance is hereafter performed by considering three scenarios, with one obstacle for the sake of simplicity, but without loss of generality: S1) the target is kept fixed in a predefined position while the obstacle moves randomly, at fixed speed or stays immobile, along a linear path (fixed target, 1D movement); S2) the target is still kept fixed in a predefined position, but the obstacle now moves randomly on a plane while avoiding areas closer to the target (fixed target, 2D movement); S3) the target is randomly initiated at the beginning of each episode, and the obstacle moves randomly on a plane while avoiding areas closer to the target (random target, 2D movement).

D. DRL policy validation

Considering scenario S3, both experience and model transfer from scenarios S1 and S2 are performed. For the robot manipulator considered in the case study, Figure 1 shows that experience transfer allows for faster convergence (i.e., in less episodes) and improved performance (i.e., higher average cumulative reward) with respect to model transfer, and both approaches outperform ex-novo training of the policy, as is evident in Table I, where the corresponding average cumulative rewards and the standard deviations (between brackets) are reported after 250 episodes (i.e., after convergence). Moreover, the percentage variation of the reward, namely $\Delta R_{\%}$, with respect to ex-novo training is indicated.

TABLE I
EFFECTIVENESS OF TRANSFER LEARNING MODES

training	R	$\Delta R_{\%}$
ex-novo	-2.52×10^4 (4.69×10^3)	–
model	-1.65×10^4 (4.59×10^3)	34.56%
experience	-9.05×10^3 (2.56×10^3)	64.13%

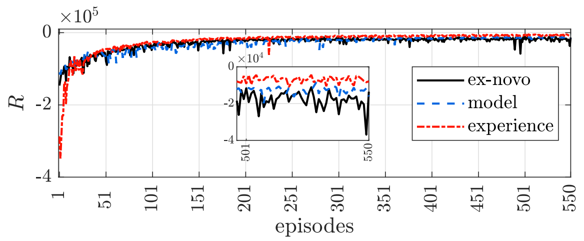

Fig. 1. Cumulative reward in case of ex-novo training (solid line), model transfer (dashed line) and experience transfer (dot-dashed line), and close-up when the rewards converge after 500 episodes

The achieved policy was tested after training for 500 different target positions, spanning the workspace of interest, with random initial robot configuration and random initial obstacle position. The procedure was repeated 30 times and an average of the selected performance indices was computed. More in detail, the following were evaluated: i) the *failure rate* i_{f} computed as the percentage of time instants over an episode in which the distance between obstacle and robot is measured equal to zero (that is, a collision event occurs), and ii) the root mean square (rms) value of the distance between the end-effector position x_{e} and the target one x^*, namely $i_{\mathrm{t}} = \mathrm{rms}(d)$, in order to have a measure of precision of the executed task. A graphic rendering of these indices is reported in Figure 2, where concerning the failure rate i_{f} we get the overall average of 0.36%, while as for i_{t} the average is 0.16 m, justified by the fact that the robot moves away from the target every time the obstacles invade its operation space, thus increasing the distance between the end-effector and the target position.

Remark 1. Both indices are related to environmental conditions (e.g., position initialization, shape and number of obstacles), and manipulability of the robot. □

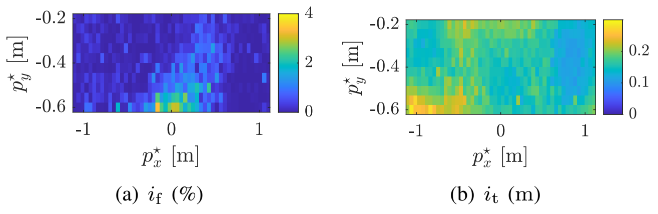

Fig. 2. Indices averaged for 30 simulations per target position of the trained policy with end-to-end approach. a) failure rate i_{f}. b) rms value i_{t} of the distance between end-effector and target

IV. HYBRID DUAL-MODE STRATEGY

The main objective of the proposed strategy, presented in this section and summarized in Algorithm 1, consists of the following motion tasks: T1) to follow a desired trajectory in order to reach a pre-specified target; T2) to avoid unexpected obstacles invading the workspace, while reaching the target.

Algorithm 1 Hybrid motion planning approach

```
Input: threshold $\epsilon$, current joints' position $q$, target pose $x^*$
Output: a collision-free motion reference for (1)
 1: repeat
 2:     compute $x_{\mathrm{e}}$ as $x_{\mathrm{e}} = k(q)$
 3:     compute the metric $\mu(d_{\mathrm{o}})$
 4:     if $\mu(d_{\mathrm{o}}) < \epsilon$ then
 5:         use DRL and pose $\dot{q}^\star = a_t$, as in (5)
 6:     else
 7:         let $x_{\mathrm{e}0} = x_{\mathrm{e}}$
 8:         use SBL starting from the initial condition $x_{\mathrm{e}0}$ to the
            final pose $x^*$
 9:         let $(q^\star, \dot{q}^\star) = (q^\star, \dot{q}^\star)^\diamond$
10:     end
11: until $x_{\mathrm{e}} \neq x^*$
12: return $(q^\star, \dot{q}^\star)$ (SBL) or $\dot{q}^\star$ (DRL)
```

Starting from an initial configuration, the motion planner finds a trajectory that connects the initial pose $x_{\mathrm{e}0}$ of the end-effector with the pose of the target point x^*. During the execution of the motion solved by the motion planner, if the metric chosen to evaluate the risk of collision with the obstacles, namely $\mu(d_{\mathrm{o}})$ (e.g., the minimum distance between the robot and obstacles), is below a fixed threshold, called ϵ, suitably selected by the designer for the sake of safety, the system hands over the control to the DRL policy. Once this condition is no longer satisfied, i.e., the obstacles, which can have different shape and behavior, are considered at safe distance, the motion planner is once again initialized with starting point as the current end-effector pose x_{e}.

Remark 2. Note that we assume that the internal control is ideal. However, since the proposed learning approach is exclusively based on data from sensors, independently of the inner control law, a certain degree of robustness is guaranteed in case of reasonable control errors. □

The motion planning from the initial pose of the end-effector to the target point in the operational space is done accordingly to the Single-Query Bi-Directional Probabilistic

Roadmap Planner with Lazy Collision Checking (SBL) algorithm [27], i.e., a sample based planning algorithm that relies on the construction of explorable trees in the configuration space. SBL is computationally light and is prone to find the shortest path [28]. In the proposed approach, the SBL path $(q^\star, \dot{q}^\star)^\diamond$ is computed without accounting for the presence of obstacles in the operative space, significantly speeding up calculations and reducing cases where the planner does not find a feasible trajectory. Collision avoidance is instead performed using the policy obtained by training the agent. Hence, at each step, given s_t as in (4), the policy produces the target velocities $\dot{q}^\star$ for each joint in the next time step as actions a_t, according to (5). The policy is deployed after a suitable number of training episodes and the convergence of the cumulative reward.

V. Motion Tests

A. Environment specifications

Experiments have been carried out in V-REP [29], interfaced with TensorFlow, through the PyRep plugin [30], using a 6 joints robot by Comau[1]. The robot must complete task T1, with random initial conditions. Throughout its actions, the robot must perform also task T2, with randomly moving obstacles (planarly or spatially), with different shape and behavior. Minimum distances measurements $d_{\mathrm{o}i}, i \in \mathbb{N}$ indicating the ith obtacle, are provided by the simulator. The metric $\mu = \min_i\{d_{\mathrm{o}i}\}$ is used for switching DRL and SBL, with threshold $\epsilon = 0.16$ chosen heuristically. The time step is equal to $50\,\mathrm{ms}$.

B. Results

Figure 3 shows the performance indices introduced in Section III-D in three different cases: i) single obstacle, planar motion, ii) single obstacle, spatial motion, iii) multiple obstacles of different shape, planar motion. Referring to Figure 3-(a,b) for case (i), it can be observed that the average failure rate i_{f} is 0.24%, while the average of i_{t} is $0.093\,\mathrm{m}$, showing improvements with respect to the end-to-end approach discussed in Section III-D. Figure 3-(c,d), for case (ii), shows comparable results with average values $i_{\mathrm{f}} = 0.31\%$ and $i_{\mathrm{t}} = 0.091\,\mathrm{m}$. Figure 3-(e,f), in case (iii), presents instead average values $i_{\mathrm{f}} = 1.35\%$ and $i_{\mathrm{t}} = 0.16\,\mathrm{m}$. As an example, in Figure 4, distance between the end-effector and the target point and distances between the robot and two obstacles are reported in case of genuine end-to-end DRL (a), and of our proposal (b). As expected, when the used metric is below the threshold, the DRL policy is used (shadow windows), otherwise the motion is solved by SBL planner. Differently from end-to-end method, the proposal ensures more stable movements during the reaching phase, maintaining the position of the end-effector exactly on the target. On the other hand, the end-to-end DRL strategy tends to deviate from such reference, thus reducing the precision of the executed task. Finally, in order to show the validity of the proposal in industrial field, in Figure 5, a spot-welding task with circular path and two different obstacles of different shape and behavior is illustrated.

[1]A video showing the performance of the hybrid approach is at the link: https://www.youtube.com/watch?v=xU43i8mryMY.

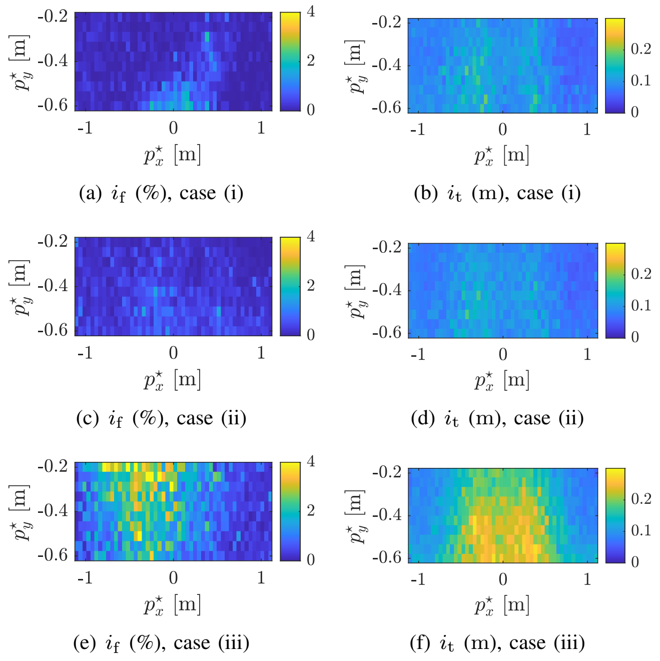

Fig. 3. Indices averaged for 30 simulations per target position of the trained policy with the proposed hybrid approach for cases (i), (ii) and (iii)

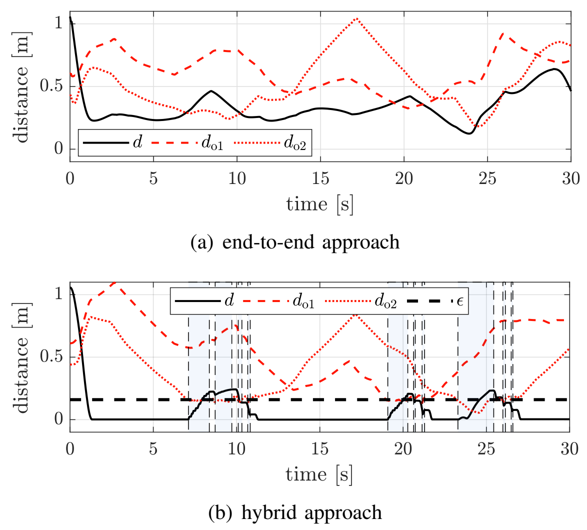

Fig. 4. Distance from target d, distance from obstacles $d_{\mathrm{o}i}$, $i = 1, 2$, and threshold ϵ, in case of end-to-end strategy (a), and of hybrid approach (b, with DRL used in the shadow windows when $\mu = \min_i\{d_{\mathrm{o}i}\} < \epsilon$)

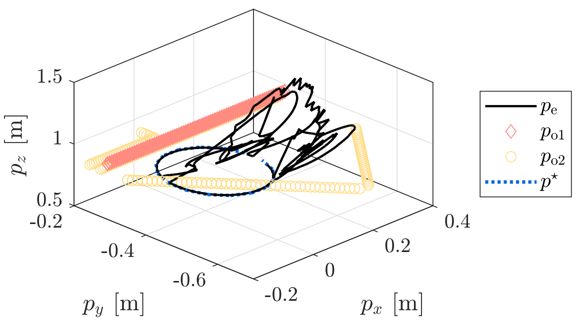

Fig. 5. Hybrid approach deployed to perform a spot-welding task

C. Comparison with a model-based approach

The proposed hybrid approach has been also tested using both the trained DRL strategy and the model-based approach in [4]. Figure 6 shows that, for the same scenario, both approaches perform successful avoidance. It is worth noticing that, while the model-based approach requires intensive computation to perform the torque compensation derived from the artificial potential field, our strategy has the significant advantage to be model-free. Indeed, apart from the off-line training phase, it can be deployed in real-time requiring only the available sensor measurements from the robot itself and the surrounding environment, while providing velocity references to the robot joints.

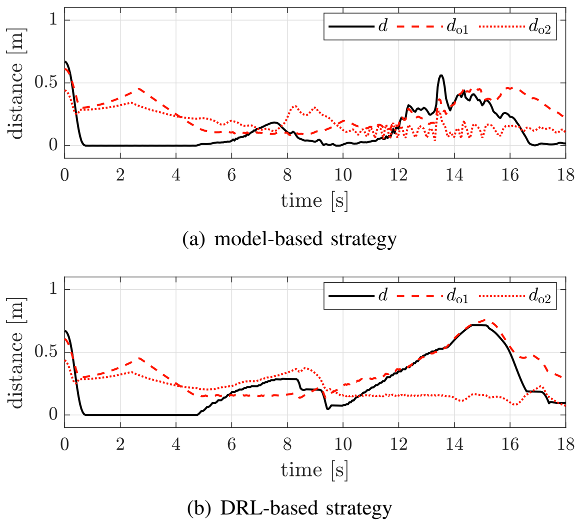

Fig. 6. Distance from target d, distance from obstacles d_{oi}, $i = 1, 2$, in case of model-based strategy [4] (a), and of DRL-based approach (b)

VI. Conclusions

In this paper, a dual-mode, hybrid control structure was proposed for robot manipulators to perform different tasks while avoiding collisions. The structure consists of a collision unaware motion planner and a DRL policy trained to avoid obstacles, by directly controlling the joint velocities. The most suitable mode is determined by a given metric, which confers to the whole structure a self-configuring capability.

References

[1] A. Bicchi, M. A. Peshkin, and J. E. Colgate, *Safety for Physical Human–Robot Interaction*. Berlin, Heidelberg: Springer, 2008, pp. 1335–1348.

[2] A. De Luca and F. Flacco, "Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration," in *4th IEEE RAS & EMBS Int. Conf. on Bio. Robot. and Biomech.*, Rome, Italy, Jun. 2012, pp. 288–295.

[3] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE T. Robot.*, vol. 33, no. 6, pp. 1292–1312, 2017.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. of Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[5] C. W. Warren, "Global path planning using artificial potential fields," in *IEEE Int. Conf. on Robot. and Autom.*, vol. 1, Scottsdale, AZ, USA, May 1989, pp. 316–321.

[6] P. Ogren, L. Petersson, M. Egerstedt, and X. Hu, "Reactive mobile manipulation using dynamic trajectory tracking: design and implementation," in *39th IEEE Conf. on Decision and Control*, vol. 3, San Francisco, CA, USA, Dec. 2000, pp. 3001–3006.

[7] L. M. Capisani, T. Facchinetti, A. Ferrara, and A. Martinelli, "Obstacle modelling oriented to safe motion planning and control for planar rigid robot manipulators," *J. Intell. Robot. Syst.*, vol. 71, no. 2, pp. 159–178, 2013.

[8] L. Balan and G. M. Bone, "Real-time 3d collision avoidance method for safe human and robot coexistence," in *IEEE/RSJ Int. Conf. on Intell. Robot. and Syst.*, Beijing, PRC, Oct. 2006, pp. 276–282.

[9] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell, "Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints," in *IEEE/RSJ Int. Conf. on Intell. Robot. and Syst.*, Hamburg, Germany, Sep. 2015, pp. 1024–1030.

[10] Y. Li, K. P. Tee, R. Yan, W. L. Chan, and Y. Wu, "A framework of humanrobot coordination based on game theory and policy iteration," *IEEE T. Robot.*, vol. 32, no. 6, pp. 1408–1418, 2016.

[11] Y. Wang, Y. Sheng, J. Wang, and W. Zhang, "Optimal collision-free robot trajectory generation based on time series prediction of human motion," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 226–233, 2018.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[13] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. of Robot. Res.*, vol. -, no. -, pp. 1–37, 2013.

[14] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Adv. Robot.*, vol. 31, no. 16, pp. 821–835, 2017.

[15] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. of Robot. Res.*, vol. 37, no. 4–5, pp. 421–436, 2017.

[16] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. of Robot. Res.*, vol. 34, no. 4-5, pp. 705–724, 2015.

[17] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," in *Robot.: Sci. and Syst. V*, Los Angeles, CL, USA, Jun. 2011, pp. 57–64.

[18] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," 2017.

[19] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *IEEE Int. Conf. on Robot. and Autom.*, St. Paul, MN, USA, May 2018, pp. 6244–6251.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[21] S. Gu, T. P. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *33rd Int. Conf. on Mach. Learn.*, New York, NY, USA, Jun. 2016.

[22] N. Snderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. Corke, "The limits and potentials of deep learning for robotics," *Int. J. of Robot. Res.*, vol. 37, no. 4-5, pp. 405–420, 2018.

[23] B. Sangiovanni, A. Rendiniello, G. P. Incremona, A. Ferrara, and M. Piastra, "Deep reinforcement learning for collision avoidance of robotic manipulators," in *European Control Conference*, Lymassol, Cyprus, Jul. 2018, pp. 2063–2068.

[24] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[25] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. Cambridge: MIT Press, 1998, vol. 2, no. 4.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[27] G. Sánchez and J.-C. Latombe, *A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking*. Berlin, Heidelberg: Springer, 2003, pp. 403–417.

[28] J. Meijer, Q. Lei, and M. Wisse, "An empirical study of single-query motion planning for grasp execution," in *IEEE Int. Conf. on Adv. Intell. Mechatronics*, Munich, Germany, Jul. 2017, pp. 1234–1241.

[29] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: a versatile and scalable robot simulation framework," in *IEEE Int. Conf. on Intell. Robot. and Syst.*, Tokyo, Japan, Nov. 2013, pp. 1321–1326.

[30] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing V-REP to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.