Why learn if you can infer? Robot arm control with Hierarchical Active Inference

Corrado Pezzato
VERSES
corrado.pezzato@verses.ai

Christopher Buckley
VERSES
christopher.buckley@verses.ai

Tim Verbelen
VERSES
tim.verbelen@verses.ai

Abstract

Recently deep reinforcement learning (RL) approaches have become successful in a wide range of domains, including robot control. Using learning instead of classical control approaches is appealing, as it avoids dealing with redundancy in over-actuated arms, hard-coding obstacle avoidance, and performing inverse kinematics calculations. However, this comes at the cost of excessive training data to fit a black-box model. In this paper, we cast motor control as an inference problem on a generative model that pertains to the robot arm's kinematic chain structure, which might be a more bio-mimetic implementation. We demonstrate that we retain both the attractive properties of RL and the efficiency of more classical forward kinematics approaches without requiring expensive training, achieving superior success rates as the degrees of freedom of the arm increase.

1 Introduction

A long-standing challenge in robot control is how to generate robust control signals in configuration space (i.e. joint positions or torques), for reaching a particular point in task space (i.e. Cartesian coordinates). Traditional approaches rely on inverse kinematics (IK) solvers or operational space controllers (OSC) [10]. However, these struggle with redundancy, i.e. when multiple solutions exist for more than 6 degrees of freedom (DOF) [6, 12], and cannot reason about constraints such as avoiding collisions between the robot arm and obstacles in the environment. Recently, deep reinforcement learning (RL) has been proposed to address these limitations, by training a policy to output joint commands directly [1, 8, 9]. Despite their impressive results, such learning-based systems typically require long training times, using hand-crafted reward functions and without safety constraints [7].

In contrast, the brain is believed to maintain an internal forward model, predicting sensory outcomes from motor actions [14]. In optimal control theory, this is complemented with an inverse model to realize control signals that minimize a well-defined value function [15, 16]. When replacing value functions with prior beliefs, optimal control can be formulated in terms of predictive coding [3]. This turns motor control into an (active) inference problem, where a generative forward model maps to desired prior beliefs, and control signals are inferred that realize those desires [13]. By defining the generative model as a hierarchical structure that mimics the structure of the arm's kinematic chain, we can obtain kinematic inversion through inference [11]. This is consistent with recent evidence that the Cerebellum consists of hierarchical cortical processing loops [4] and is mainly involved in forward modeling [18].

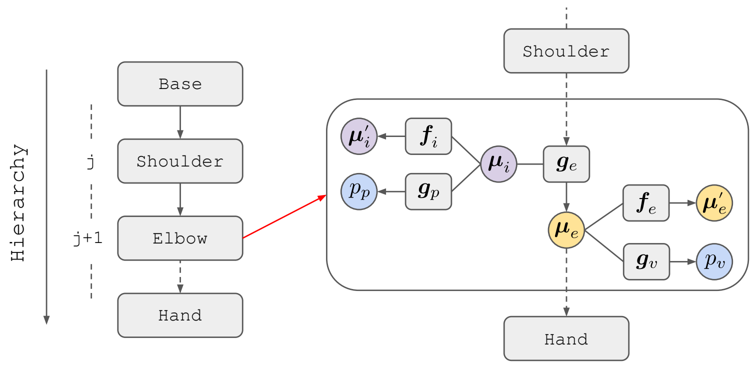


Figure 1: Overview of the hierarchical active inference approach for n-DOF robot arm control. Intrinsic and extrinsic beliefs $\boldsymbol{\mu}_i$, $\boldsymbol{\mu}_e$ are internal representations of joint angles and Cartesian poses respectively. They generate proprioceptive and visual predictions p_p and p_v at their level according to the generative models $\boldsymbol{g}_v$, $\boldsymbol{g}_p$. They are also linked through a kinematic generative model $\boldsymbol{g}_e$. The functions $\boldsymbol{f}_i$ and $\boldsymbol{f}_e$ describe the dynamics and are used to guide goal-directed behavior.

In this paper, we extend this principle for controlling n-DOF robotic arms in 3D space and compare it against trained RL policies. We demonstrate that our controller is more accurate than the policies trained with RL, and although slower in reaching a goal, on average yields shorter path lengths. We argue that active inference is well-suited to combine the flexibility of learning approaches with the efficiency of traditional robotic control methods that utilize forward models.

2 Methodology

Active inference unifies both action and perception because both drive agents to minimize surprise [2]. In the context of motor control, an agent is equipped with prior beliefs in an extrinsic frame (i.e. Cartesian coordinates) and infers motor commands that minimize prediction errors [13]. We propose a hierarchical kinematic generative model, which is fit for controlling any n-DOF robotic arm.

2.1 Generic kinematic generative model

The structure of a hierarchical active inference (HAIF) agent is depicted in fig. 1. Each level of the hierarchy has the same structure, which is repeated for each link of a robot arm. The main contribution of this paper is the definition of a generic kinematic generative model $\boldsymbol{g}_e$. In the proposed model, the HAIF agent comprises an intrinsic belief $\boldsymbol{\mu}_i$ about joint angles and links' lengths, as well as an extrinsic belief $\boldsymbol{\mu}_e$ about a link's absolute Cartesian position and orientation, for each joint j:

$$\boldsymbol{\mu}_i^j = \begin{bmatrix}\theta^j, & l^j\end{bmatrix}^\top \qquad \boldsymbol{\mu}_e^j = \begin{bmatrix}x^j, & y^j, & z^j, & q_w^j, & q_x^j, & q_y^j, & q_z^j\end{bmatrix}^\top = \begin{bmatrix}\boldsymbol{t}^j & \boldsymbol{q}^j\end{bmatrix}^\top . \tag{1}$$

As opposed to [11], we consider quaternions to represent generic rotations instead of Euler angles. This choice avoids translating often between Euler angles and Rotation matrices in a generic 3D case. In general, the kinematic generative model $\boldsymbol{g}_e$ computes the extrinsic beliefs at the current level j given the current intrinsic beliefs $\boldsymbol{\mu}_i^j$ and the extrinsic beliefs from the level below $\boldsymbol{\mu}_e^{j-1}$, i.e. $\boldsymbol{\mu}_e^j = \boldsymbol{g}_e(\boldsymbol{\mu}_i^j, \boldsymbol{\mu}_e^{j-1})$. The goal is to obtain a set of equations to describe how the internal beliefs of active inference agents are generated and updated over time. Following [11], the biologically plausible belief update equations are:

$$\dot{\boldsymbol{\mu}}_i^j = \begin{bmatrix}\boldsymbol{\mu}_i^{j'} + \pi_p^j \boldsymbol{\varepsilon}_p^j + \partial_{\boldsymbol{\mu}_i} \boldsymbol{g}_e^\top \pi_e^{j+1} \boldsymbol{\varepsilon}_e^{j+1} + \partial \boldsymbol{f}_i^{j\top} \pi_{\boldsymbol{\mu}_i}^j \boldsymbol{\varepsilon}_{\boldsymbol{\mu}_i}^j \\ -\pi_{\boldsymbol{\mu}_i}^j \boldsymbol{\varepsilon}_{\boldsymbol{\mu}_i}^j\end{bmatrix} \tag{2}$$

$$\dot{\boldsymbol{\mu}}_e^j = \begin{bmatrix}\boldsymbol{\mu}_e^{j'} - \pi_e^j \boldsymbol{\varepsilon}_e^j + \partial_{\boldsymbol{\mu}_e} \boldsymbol{g}_e^\top \pi_e^{j+1} \boldsymbol{\varepsilon}_e^{j+1} + \pi_v^j \boldsymbol{\varepsilon}_v^j + \partial \boldsymbol{f}_e^{j\top} \pi_{\boldsymbol{\mu}_e}^j \boldsymbol{\varepsilon}_{\boldsymbol{\mu}_e}^j \\ -\pi_{\boldsymbol{\mu}_e}^j \boldsymbol{\varepsilon}_{\boldsymbol{\mu}_e}^j\end{bmatrix}, \tag{3}$$

where $\pi_p, \ \pi_e, \pi_v$ are precision parameters for proprioceptive, extrinsic, and visual models, and $\boldsymbol{\varepsilon}_p, \ \boldsymbol{\varepsilon}_e$ and $\boldsymbol{\varepsilon}_v$ are the proprioceptive, extrinsic, and visual prediction errors respectively. Finally,

$\boldsymbol{\varepsilon}^j_{\boldsymbol{\mu}_i} = \boldsymbol{\mu}^{j'}_i - \boldsymbol{f}^j_i(\boldsymbol{\mu}^j_i)$ and $\boldsymbol{\varepsilon}^j_{\boldsymbol{\mu}_e} = \boldsymbol{\mu}^{j'}_e - \boldsymbol{f}^j_e(\boldsymbol{\mu}^j_i)$ are dynamics prediction errors, with precision $\pi_{\boldsymbol{\mu}_i}$ and $\pi_{\boldsymbol{\mu}_e}$. These are used to achieve goal-directed behavior and collision avoidance as explained later.

In the equations above, we assumed to be able to observe joint positions, velocities, and link positions such that $\boldsymbol{g}_p$ and $\boldsymbol{g}_v$ are identity mappings. Link positions can be computed from joint positions via forward kinematics or estimated via visual input. We now have to find a suitable form for $\boldsymbol{g}_e$ to easily compute the gradients with respect to intrinsic and extrinsic beliefs. As in our model, $\boldsymbol{g}_e$ describes the 3D position and orientation of the subsequent link of a kinematic chain given the pose of the previous one, the natural starting point to find $\boldsymbol{g}_e$ is considering generic transformation matrices. To compute the absolute position and orientation of the current link j given the absolute position and orientation of the previous one, we can write:

$$ {}^{w}T_j = \left[\begin{array}{c|c} R^{j-1}R^j & \boldsymbol{t}^{j-1} + R^{j-1}\boldsymbol{t}^j \\ \hline \mathbf{0} & 1 \end{array}\right], \tag{4}$$

where w indicates the world frame as an absolute reference, R represents a rotation matrix, and $\boldsymbol{t}$ a translation vector. The world frame can be the base link of a robot arm. From eq. (4), we note that the resulting absolute rotation of a link is the multiplication of two rotation matrices. However, we can express this as a quaternion multiplication $\boldsymbol{q}^{j-1} \cdot \boldsymbol{q}^j$. Similarly, we can rotate a vector $\boldsymbol{t}^j$ by a quaternion $\boldsymbol{q}^{j-1}$ corresponding to R^{j-1} such that we can define the generative model $\boldsymbol{g}_e$ as:

$$\boldsymbol{g}_e(\boldsymbol{\mu}^j_i, \boldsymbol{\mu}^{j-1}_e) = \begin{bmatrix} \boldsymbol{t}^{j-1} + \boldsymbol{h}(\boldsymbol{q}^{j-1} \cdot [0\ \boldsymbol{t}^j] \cdot \boldsymbol{q}^{j-1*}) \\ \boldsymbol{q}^{j-1} \cdot \boldsymbol{q}^j \end{bmatrix}, \tag{5}$$

where $\boldsymbol{q}^{j-1*}$ is the conjugate quaternion, $''\cdot''$ represents the Hamilton product, and $\boldsymbol{h}()$ is a function that returns the imaginary coefficients of a quaternion. In our HAIF agent, the translation $\boldsymbol{t}^{j-1}$ and quaternion $\boldsymbol{q}^{j-1}$ are given by the extrinsic beliefs $\boldsymbol{\mu}^{j-1}_e$. The translation vector $\boldsymbol{t}^j$ and rotation $\boldsymbol{q}^j$ are instead dependent on the kinematic properties of the current link j and the joint angle and length $\theta^j,\ l^j$. Considering a generic Denavit–Hartenberg (DH) transformation matrix

$$\left[\begin{array}{ccc|c} \cos\theta^j & -\sin\theta^j\cos\alpha^j & \sin\theta^j\sin\alpha^j & l^j\cos\theta^j \\ \sin\theta^j & \cos\theta^j\cos\alpha^j & -\cos\theta^j\sin\alpha^j & l^j\sin\theta^j \\ 0 & \sin\alpha^j & \cos\alpha^j & d^j \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{6}$$

we note that the translation vector is simply $\boldsymbol{t}^j = [l^j\cos\theta^j, l^j\sin\theta^j, d^j]$. According to the DH convention, the rotational part of the transformation matrix is the composition of a rotation θ^j about the previous z-axis and a rotation of α^j around the x-axis. We can then write:

$$\boldsymbol{q}^j = \left[\cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2},\quad \cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2},\quad \cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2},\quad \cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2}\right]. \tag{7}$$

The generative model in eq. (5) can then be fully specified as a function of the intrinsic and extrinsic beliefs, and the gradients can be computed in closed form, see appendix A for full details.

2.2 Arm model, goals, and control

Given the generic kinematic generative model in eq. (5) and the belief update equations eqs. (2) and (3), we can control arbitrary serial kinematic chains. To do so, the first step is to define the DH parameters for a robot manipulator. This standard and well-known procedure describes a kinematic chain defining $l^j,\ \alpha^j,\ d^j$ to populate eq. (6) for every link.

To realize goal-directed behavior, we can define attractive goals and repulsive forces as in [11]. Goals can be both intrinsic (joint positions) or extrinsic (Cartesian poses), and they can be combined to define future desired states $\boldsymbol{\mu}^*$. Goals act as attractors, forming dynamic functions $\boldsymbol{f}_a = \kappa_a(\boldsymbol{\mu}^* - \boldsymbol{\mu})$ that linearly minimize the distance between the desired and current states. The desired states can be defined flexibly in terms of the current beliefs as

$$\boldsymbol{\mu}^* = N\boldsymbol{\mu} + \boldsymbol{n}^*, \tag{8}$$

where N achieves dynamic behaviors, such as keeping a limb vertical by imposing the $x,\ y$ coordinates of a link to be the same as the previous one, while $\boldsymbol{n}^*$ imposes an attractor to a static configuration. For example, reaching a position target with link j would result in $N = \mathbf{0}$ and $\boldsymbol{n}^* = [x^*,\ y^*,\ z^*,\ \boldsymbol{q}^j]$.

The same idea can be used for collision avoidance through repulsive forces where a repulsive state $\boldsymbol{\mu}^!$ has to be avoided. Note that $\boldsymbol{\mu}^!$ can concern intrinsic beliefs, to realize joint limit avoidance or extrinsic beliefs for collision avoidance with the environment. We define joint limit avoidance as:

$$\boldsymbol{f}_{r,\theta}(\boldsymbol{\mu}) = \begin{cases} 0, & \text{if } ||\boldsymbol{e}_\theta|| > \gamma_\theta \\ k_{r,\theta}\zeta(1/\gamma_\theta - 1/||\boldsymbol{e}_\theta||), & \text{otherwise} \end{cases}, \tag{9}$$

where $\boldsymbol{e}_\theta = \boldsymbol{\mu}_\theta^! - \boldsymbol{\mu}_\theta$, $\boldsymbol{\mu}_\theta$ is the slice of beliefs about joint angles, $\boldsymbol{\mu}_\theta^!$ are the joint limits, and γ_θ is a chosen threshold. The variable $\zeta \in \{-1,\ 1\}$ is negative for lower limits and positive for upper limits. The collision avoidance strategy is instead the same as [11]:

$$\boldsymbol{f}_{r,obst}(\boldsymbol{\mu}) = \begin{cases} 0, & \text{if } ||\boldsymbol{e}_{obst}|| > \gamma_{obst} \\ k_{r,obst}(1/\gamma_{obst} - 1/||\boldsymbol{e}_{obst}||)\boldsymbol{e}_{obst}/||\boldsymbol{e}_{obst}||^3, & \text{otherwise} \end{cases}, \tag{10}$$

where $\boldsymbol{e}_{obst} = \boldsymbol{\mu}_{pos}^! - \boldsymbol{\mu}_{pos}$, $\boldsymbol{\mu}_{pos}$ is the slice of beliefs about link positions, and $\boldsymbol{\mu}_{pos}^!$ is the position of an obstacle. Goal attractors and repulsive forces for joint limits and collision avoidance are then summed together to form the dynamics function of a single level. This allows one to achieve behaviors such as reaching a target while avoiding an obstacle. Hyperparameters are manually chosen to achieve sufficient performance in the test cases but could be automatically optimized. The control action is computed by minimizing the proprioceptive component of the free energy with respect to the control signals [11]:

$$\dot{\boldsymbol{a}} = -\partial_a \mathcal{F}_p = -\partial_a \tilde{\boldsymbol{s}}_p \pi_p \tilde{\boldsymbol{\varepsilon}}_p \tag{11}$$

where $-\partial_a \tilde{\boldsymbol{s}}_p$ is the partial derivative of proprioceptive observations with respect to the control, and $\tilde{\varepsilon}_p = \tilde{\boldsymbol{s}}_p - \boldsymbol{g}_p(\boldsymbol{\mu})$ are the generalized proprioceptive prediction errors.

3 Experiments

We evaluate the ability of HAIF to perform reaching tasks in a 3D environment and compare the performance with a policy trained with RL. We also test how our method generalizes when the arm has to reach a target while avoiding moving obstacles or needs to infer the lengths of its links.

Reaching tasks We benchmark our solution according to the RL reaching benchmark from [1], where the goal is to control a robot in joint space to reach a target position in Cartesian space. The authors in [1] provide a reusable open-source implementation of various RL algorithms with several examples for training an agent to reach a target position. We evaluate reaching performance in two cases: 1) a single fixed goal, and 2) a randomly generated goal position within a constrained workspace. We utilize the same robot arms as in [1], namely a 5DOF WidowX and a 7DOF Jaco arm. All experiments and training are performed on a laptop with Intel Core i9 CPU and RTX 4090 GPU.

To solve the tasks, we train a total of four different RL agents with five seeds, one per task and robot, using the provided PPO implementations, see appendix B for details. To apply HAIF to the two different robots and tasks, instead, we only need to specify the DH parameters for the

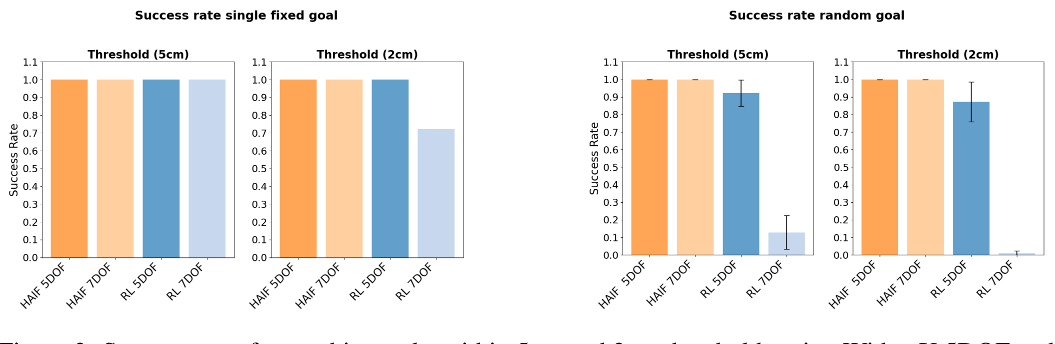


Figure 2: Success rates for reaching tasks within 5cm and 2cm thresholds using WidowX 5DOF and Jaco 7DOF robot arms.

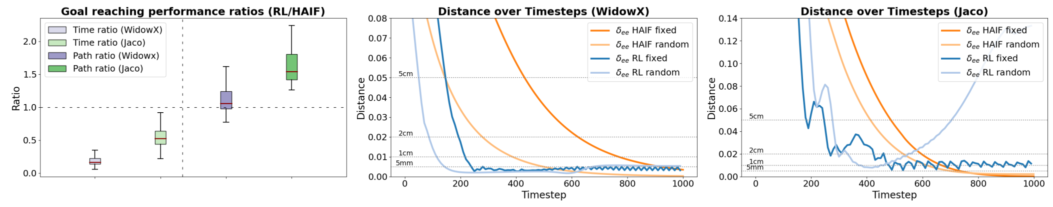


Figure 3: Average time to goal (ratio) and path length (ratio) (left). Example of end effector distance δ_{ee} to goal for WidowX (center) and Jaco (right) while reaching a random goal.

different kinematic chains and provide an attractor for the end effector to the goal position as $\boldsymbol{\mu}^* = [x^*,\ y^*,\ z^*,\ \boldsymbol{q}^{ee}]$. We perform 100 trials per task, and we consider different evaluation metrics: *success rate*, *relative time to goal*, and *relative path length*. The success rate is computed for four different thresholds of 5cm, 2cm, 1cm, and 0.5 cm. An episode is considered successful if the end-effector is within a threshold at the end of the episode. The relative time to goal is the average time taken to reach goals defined as the ratio between the methods RL/HAIF (the higher the better). The path length is the average path length to reach goals defined as the ratio between RL/HAIF (the higher the better). Both the time to goal and path length are computed as soon as the end-effector is within 5cm of the target.

For the fixed goal-reaching and random goal-reaching tasks, the success rates for the 5cm and 2cm thresholds are reported in fig. 2. The results for the 1cm and 0.5cm thresholds can be found in the appendix in fig. 8. In the considered reaching tasks, our method achieves a success rate of 100% for each of the selected thresholds. The performance of the RL agents, however, deteriorates as the number of DOFs of the robot arm increases and when goals are randomly assigned. The time to goal, path length, and an example of end effector distance to goal over time is shown in fig. 3. On average the RL controller moves faster, but the path length is similar or longer. Our controller also displays a smooth trajectory, converging precisely to the target, whereas the RL policies are more jittery and oscillate around the target at the distance they got reward during training.

Generalization To emphasize the generality of our method, we apply HAIF to a different arm, a Fetch 7-DOF to 1) reach tasks in the presence of dynamic obstacles and 2) reach with unknown link lengths. To solve these tasks, we only need to provide a new set of DH parameters for Fetch. For the collision avoidance task, we spawn two dynamic obstacles with a constant velocity of 0.1 to 0.3m/s aiming towards the arm. An example of dodging the obstacles is reported in fig. 4. For links' length estimation tasks, instead, we initialize the HAIF agent with wrong beliefs about these parameters, such that the resulting kinematic chain is incorrect. We randomly sample length values in a range $[0,\ 0.2]m$ for every link. As can be seen in fig. 5, the estimated (red) kinematic chain converges to the true one (blue). Future work will extend the model to estimate the full set of DH parameters.

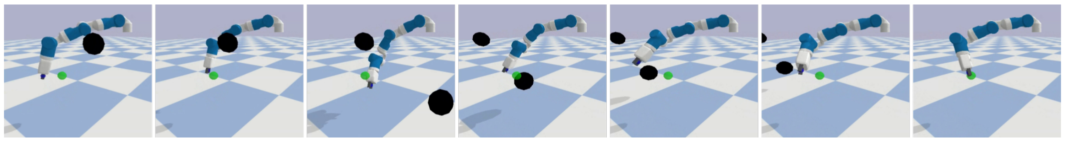


Figure 4: Obstacle avoidance of two dynamic obstacles while reaching a target using 7-DOF Fetch arm. We assume the positions of the obstacles are observed.

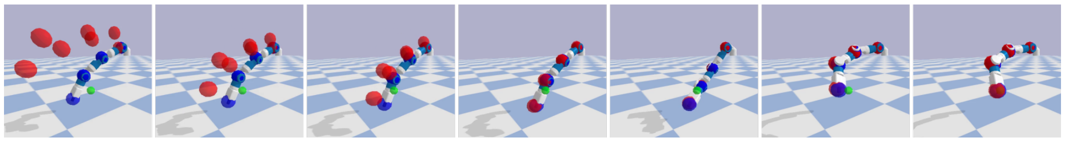


Figure 5: Link lengths (red) are inferred while reaching a goal.

4 Conclusion and Discussion

This paper presented a hierarchical active inference controller for n-DOF robot arm control. We demonstrate that casting control as inference yields robust performance, without requiring long training times typical of contemporary reinforcement learning methods. Although black-box models optimized with gradient descent are attractive as a one-size-fits-all solution, we argue that their strength, i.e. the lack of predefined priors, is also their weakness for particular, well-defined applications, as they have to learn these prior structures solely from data. In the case of robot control, writing down the generative model using knowledge of the structure of the kinematic chain, enables us to tackle the problem with only inference. Note that in our model only the number of joints is hard-wired in the hierarchical structure of the model, but e.g. the link lengths can be inferred from observations. Future work will aim to further the benchmark against various RL methods on more challenging manipulation tasks, extend the work to full body control of mobile manipulators, and interface with a vision perception pipeline for vision-based active inference [17].

References

[1] Pierre Aumjaud, David McAuliffe, Francisco J. Rodríguez Lera, and Philip Cardiff. rl_reach: Reproducible reinforcement learning experiments for robotic reaching tasks. *Software Impacts*, 8:100061, May 2021.

[2] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, January 2010.

[3] Karl Friston. What is optimal about motor control? *Neuron*, 72(3):488–498, 2011.

[4] Shlomi Haar and Opher Donchin. A revised computational neuroanatomy for motor control. *Journal of Cognitive Neuroscience*, 32(10):1823–1836, October 2020.

[5] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, et al. Stable baselines, 2018.

[6] J. Hollerbach and Ki Suh. Redundancy resolution of manipulators through torque optimization. *IEEE Journal on Robotics and Automation*, 3(4):308–316, August 1987.

[7] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: challenges, representations, and algorithms. *J. Mach. Learn. Res.*, 22(1), jan 2021.

[8] Visak Kumar, David Hoeller, Balakumar Sundaralingam, Jonathan Tremblay, and Stan Birchfield. Joint space control via deep reinforcement learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, September 2021.

[9] Pietro Mazzaglia, Nicholas Backshall, Xiao Ma, and Stephen James. Redundancy-aware action spaces for robot learning, 2024.

[10] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, June 2008.

[11] Matteo Priorelli, Giovanni Pezzulo, and Ivilin Peev Stoianov. Deep kinematic inference affords efficient and scalable control of bodily movements. *Proceedings of the National Academy of Sciences*, 120(51):e2309058120, December 2023.

[12] H. Seraji. Configuration control of redundant manipulators: theory and implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472–490, 1989.

[13] Stewart Shipp, Rick A. Adams, and Karl J. Friston. Reflections on agranular architecture: predictive coding in the motor cortex. *Trends in Neurosciences*, 36(12):706–716, December 2013.

[14] Hirokazu Tanaka, Takahiro Ishikawa, Jongho Lee, and Shinji Kakei. The cerebro-cerebellum as a locus of forward model: A review. *Frontiers in Systems Neuroscience*, 14, 2020.

[15] Emanuel Todorov. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915, August 2004.

[16] Emanuel Todorov and Michael I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, November 2002.

[17] Toon Van de Maele, Tim Verbelen, Pietro Mazzaglia, Stefano Ferraro, and Bart Dhoedt. Object-centric scene representations using active inference. *Neural Computation*, 36(4):677–704, March 2024.

[18] Fatemeh Yavari, Shirin Mahdavi, Farzad Towhidkhah, Mohammad-Ali Ahmadi-Pajouh, Hamed Ekhtiari, and Mohammad Darainy. Cerebellum as a forward but not inverse model in visuomotor adaptation task: a tdcs-based and modeling study. *Experimental Brain Research*, 234(4):997–1012, December 2015.

A Generative model details

A.1 Quaternions vs Euler angles

Quaternions and Euler angles are both methods for representing rotations in three-dimensional space, but they differ significantly in their properties, applications, and advantages. Three angles represent Euler Angles, typically denoted roll, pitch, and yaw. They describe rotations around the X, Y, and Z axes sequentially. Quaternions represent rotations using a four-dimensional structure with one scalar part and a three-dimensional vector part. They consist of four components $\boldsymbol{q} = [q_w,\ q_x,\ q_y,\ q_z]$. Euler angles are subject to singularities while quaternions are a continuous representation. Apart from this issue, using quaternions in the proposed kinematic generative model is advantageous because they allow to combine rotations through quaternion multiplication. On the contrary, summing Euler angles directly does not yield a meaningful combined rotation in the general 3D case.

A.2 Kinematic generative model

The generic kinematic model in eq. (5) can be expressed as

$$\boldsymbol{g}_e(\boldsymbol{\mu}_i^j, \boldsymbol{\mu}_e^{j-1}) = \begin{bmatrix} \boldsymbol{t}^{j-1} + \boldsymbol{h}(\boldsymbol{q}^{j-1} \cdot [0\ \boldsymbol{t}^j] \cdot \boldsymbol{q}^{j-1*}) \\ \boldsymbol{q}^{j-1} \cdot \boldsymbol{q}^j \end{bmatrix}, = \begin{bmatrix} x^{j-1} + x_{tf} \\ y^{j-1} + y_{tf} \\ z^{j-1} + z_{tf} \\ q_{w,\ tf} \\ q_{x,\ tf} \\ q_{y,\ tf} \\ q_{z,\ tf} \end{bmatrix}, \tag{12}$$

where x_{tf}, y_{tf}, z_{tf} and $q_{*,\ tf}$ are the transformed translations and rotation. Computing the Hamilton products yields the following expressions for the transformed positions

$$\begin{aligned}
x_{tf} \quad &= \quad {q_w^{j-1}}^2 l^j \cos\theta^j + {q_x^{j-1}}^2 l^j \cos\theta^j - {q_y^{j-1}}^2 l^j \cos\theta^j - {q_z^{j-1}}^2 l^j \cos\theta^j \\
&\quad\quad +2q_x^{j-1} q_y^{j-1} l^j \sin\theta^j + 2q_x^{j-1} q_z^{j-1} d^j + 2q_w^{j-1} q_y^{j-1} d^j - 2q_w^{j-1} q_z^{j-1} l^j \sin\theta^j, \\
y_{tf} \quad &= \quad {q_w^{j-1}}^2 l^j \sin\theta^j - {q_x^{j-1}}^2 l^j \sin\theta^j + {q_y^{j-1}}^2 l^j \sin\theta^j - {q_z^{j-1}}^2 l^j \sin\theta^j \\
&\quad\quad +2q_x^{j-1} q_y^{j-1} l^j \cos\theta^j + 2q_y^{j-1} q_z^{j-1} d^j - 2q_w^{j-1} q_x^{j-1} d^j + 2q_w^{j-1} q_z^{j-1} l^j \cos\theta^j, \\
z_{tf} \quad &= \quad {q_w^{j-1}}^2 d^j - {q_x^{j-1}}^2 d^j - {q_y^{j-1}}^2 d^j + {q_z^{j-1}}^2 d^j \\
&\quad\quad +2q_x^{j-1} q_z^{j-1} l^j \cos\theta^j + 2q_y^{j-1} q_z^{j-1} l^j \sin\theta^j - 2q_w^{j-1} q_y^{j-1} l^j \cos\theta^j + 2q_w^{j-1} q_x^{j-1} l^j \sin\theta^j,
\end{aligned}$$

and orientation:

$$
\begin{aligned}
q_{w,\,tf} &= q_w^{j-1}\cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2} - q_x^{j-1}\cos\frac{\theta^j}{2}\sin\frac{\alpha^j}{2} - q_y^{j-1}\sin\frac{\theta^j}{2}\sin\frac{\alpha^j}{2} - q_z^{j-1}\sin\frac{\theta^j}{2}\cos\frac{\alpha^j}{2},\\
q_{x,\,tf} &= q_w^{j-1}\cos\frac{\theta^j}{2}\sin\frac{\alpha^j}{2} + q_x^{j-1}\cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2} + q_y^{j-1}\sin\frac{\theta^j}{2}\cos\frac{\alpha^j}{2} - q_z^{j-1}\sin\frac{\theta^j}{2}\sin\frac{\alpha^j}{2},\\
q_{y,\,tf} &= q_w^{j-1}\sin\frac{\theta^j}{2}\sin\frac{\alpha^j}{2} - q_x^{j-1}\sin\frac{\theta^j}{2}\cos\frac{\alpha^j}{2} + q_y^{j-1}\cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2} + q_z^{j-1}\cos\frac{\theta^j}{2}\sin\frac{\alpha^j}{2},\\
q_{z,\,tf} &= q_w^{j-1}\sin\frac{\theta^j}{2}\cos\frac{\alpha^j}{2} + q_x^{j-1}\sin\frac{\theta^j}{2}\sin\frac{\alpha^j}{2} - q_y^{j-1}\cos\frac{\theta^j}{2}\sin\frac{\alpha^j}{2} + q_z^{j-1}\cos\frac{\theta^j}{2}\cos\frac{\alpha^j}{2}.
\end{aligned}
$$

A.3 Gradients of the kinematic generative model

To compute the updates for intrinsic and extrinsic beliefs in eqs. (2) and (3), the gradients of the generative model with respect to $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_e$ have to be computed. It holds in general that:

$$
\frac{\partial \boldsymbol{g}_e}{\partial \boldsymbol{\mu}_i} = \begin{bmatrix} \partial_\theta \boldsymbol{g}_e \\ \partial_l \boldsymbol{g}_e \end{bmatrix} \in \mathbb{R}^{2\times 7}, \qquad \frac{\partial \boldsymbol{g}_e}{\partial \boldsymbol{\mu}_e} = \begin{bmatrix} \partial_{x,y,z} \boldsymbol{g}_e \\ \partial_{\boldsymbol{q}} \boldsymbol{g}_e \end{bmatrix} \in \mathbb{R}^{7\times 7} \tag{13}
$$

Thanks to the choice of using quaternions as singularity-free orientation representation, these gradients are easy to compute since the terms in the generative model are either linear or quadratic in the parameters, or they appear as arguments of sine and cosine functions. Note that the intrinsic beliefs could also be augmented to keep track of all the DH parameters so that one could estimate not only link lengths but for instance their relative orientation.

A.4 DH parameters for the robot arms

The DH parameters for the different robot arms based on the URDF files of the WidowX, Jaco, and Fetch robot arms are reported in tables 1 to 3.

Table 1: Jaco DH table

$\boldsymbol{l}$	$\boldsymbol{\alpha}$	$\boldsymbol{d}$	$\boldsymbol{\theta}_i$
0.0	90°	0.2755	θ_1
0.0	90°	0.0	θ_2 - 180°
0.0	90°	-0.410	θ_3
0.0	-90°	0.0	θ_4 - 180°
0.0	-90°	-0.3111	θ_5
0.0	90°	0.0	θ_6 + 180°
0.0	0°	-0.2638	θ_7

Table 2: Fetch DH table

$\boldsymbol{l}$	$\boldsymbol{\alpha}$	$\boldsymbol{d}$	$\boldsymbol{\theta}_i$
0.117	-90°	0.0	θ_1
0.0	-90°	0.0	θ_2 - 90°
0.0	90°	0.352	θ_3
0.0	-90°	0.0	θ_4
0.0	90°	0.3215	θ_5
0.0	-90°	0.0	θ_6
0.0	0°	0.30495	θ_7

Table 3: WidowX DH table

$\boldsymbol{l}$	$\boldsymbol{\alpha}$	$\boldsymbol{d}$	$\boldsymbol{\theta}_i$
0.0	-90°	0.125	θ_1
0.15	0°	0.0	θ_2 - 70.14°
0.14203	0°	0.0	θ_3 + 70.14°
0.0	-90°	0.0	θ_4 - 90°
0.0	0°	0.1145	θ_5

B RL agents

B.1 Environment details

We train four separate RL agents: two for WidowX 5DOF and two for Jaco 7DOF, to reach either a fixed goal or a randomized one in the workspace. The training environment is depicted in fig. 6.

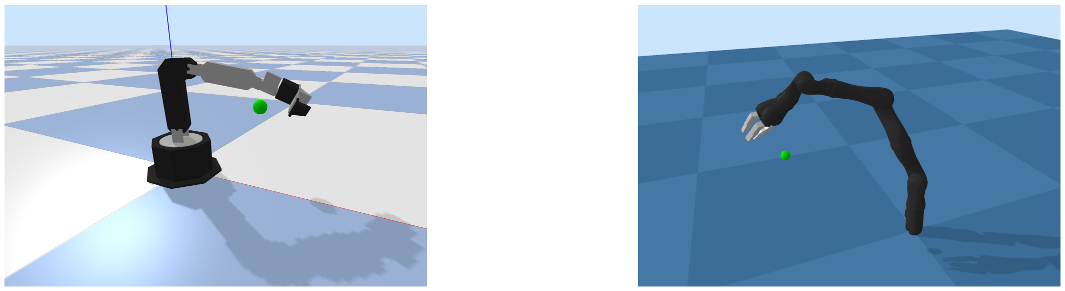


Figure 6: WidowX 5DOF and Jaco 7DOF training environments from [1].

The benchmark in [1] is built on top of Stable Baselines 3 [5] and it supports various model-free RL algorithms. We chose Proximal Policy Optimization (PPO) because it performs well on robotic control tasks, including robot arm reaching [1]. We report the settings we utilized:

- **Environment names**: `widowx_reacher-v5` (fixed goal), `widowx_reacher-v28` (random goal), `ReachingJaco-v3` (fixed goal), `ReachingJaco-v7` (random goal),
- **Reward Function**: **REW15** in [1]. We chose a combination of sparse and dense rewards because it was indicated as the best choice for minimizing distance in reaching tasks.

$$r = \begin{cases} -d_t, & \text{if } d \geq \epsilon \\ 1, & \text{if } d < \epsilon \end{cases} \tag{14}$$

  where
  - r: Reward
  - d_t: Distance at time t
  - ϵ: Threshold for sparse reward (0.001)
- **Observation space**: **OBS5** in [1].

$$[ETx, ETy, ETz, EGx, EGy, EGz, Gx, Gy, Gz, A1, A2, A3, A4, A5, A6] \tag{15}$$

  where
  - Ei : End effector coordinate along the i axis
  - Gi : Goal coordinate along the i axis
  - EGi : Vector End effector - Goal along the i axis
  - ETx : Vector End effector - Torso along the i axis
  - Ai : Angular position of joint i
- **Action space**: **ACT2** in [1]. Relative joint position (Continuous position control and compute physics for collision)

$$[\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6] \tag{16}$$

  where δ_i : Increment from previous joint position (in rad)

B.2 Performance metrics

We trained five different seeds per RL agent and we report the mean rewards obtained from the benchmark results [1] in fig. 7:

C Additional results

We report below additional results for success rates of our method and the RL agents in case of tighter thresholds to reach a goal, namely 1cm and 0.5cm, see fig. 8. Additionally, we report the evaluation of the time to goal and path length ratios seed by seed in fig. 9.

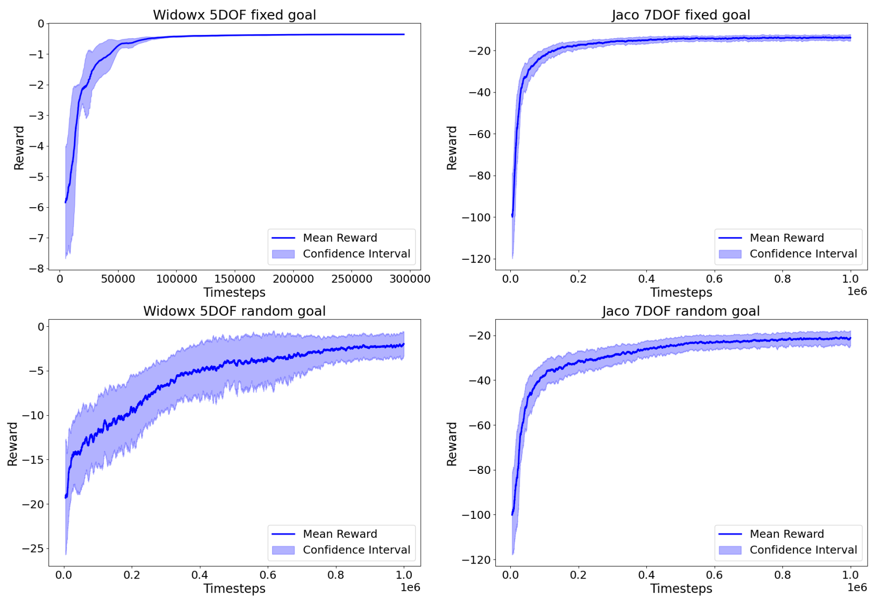


Figure 7: Mean rewards for the trained RL agents for reaching tasks.

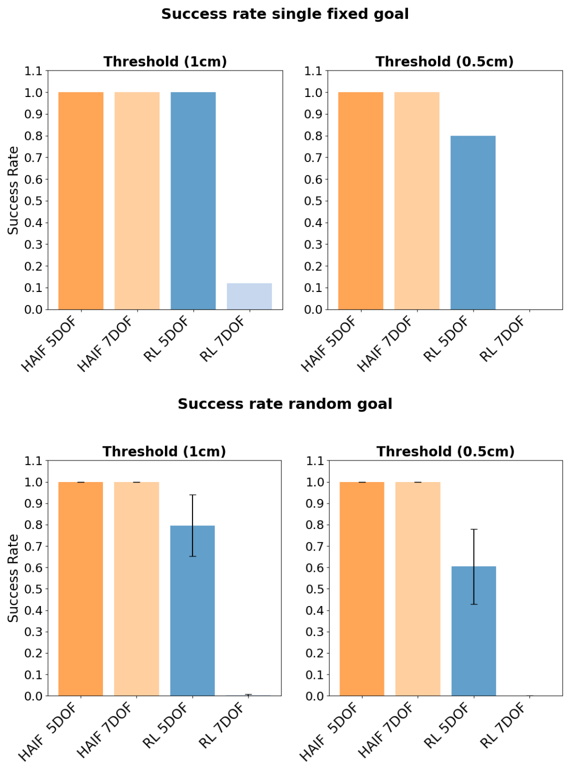
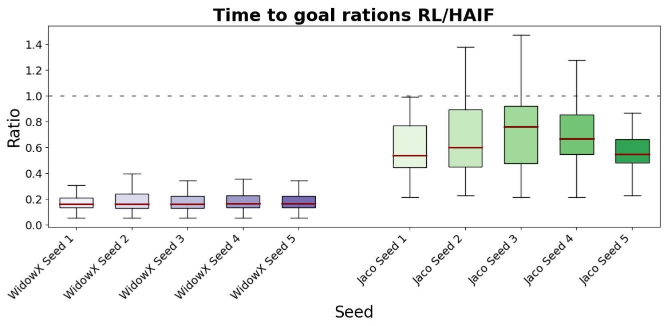


Figure 8: Success rates for reaching tasks within 1cm and 0.5cm thresholds using WidowX 5DOF and Jaco 7DOF robot arms.



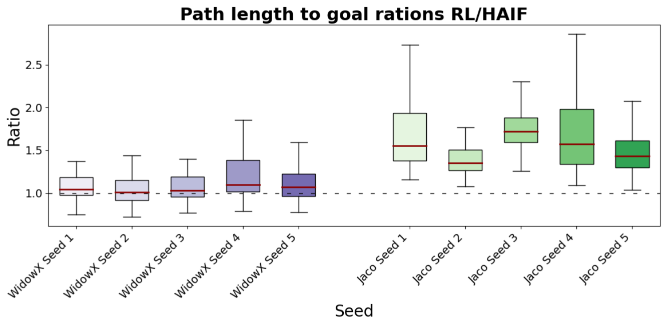


Figure 9: Time to goal and path length ratios (RL/HAIF) for different seeds. The higher the better, where unit value means equal performance.