nature machine intelligence

Article

https://doi.org/10.1038/s42256-023-00752-z

Hierarchical generative modelling for autonomous robots

Received: 12 October 2021

Accepted: 29 September 2023

Published online: 02 November 2023

Check for updates

Kai Yuan[1,5], Noor Sajid[2,5], Karl Friston[2,3] & Zhibin Li[4] ✉

Humans generate intricate whole-body motions by planning, executing and combining individual limb movements. We investigated this fundamental aspect of motor control and approached the problem of autonomous task completion by hierarchical generative modelling with multi-level planning, emulating the deep temporal architecture of human motor control. We explored the temporal depth of nested timescales, where successive levels of a forward or generative model unfold, for example, object delivery requires both global planning and local coordination of limb movements. This separation of temporal scales suggests the advantage of hierarchically organizing the global planning and local control of individual limbs. We validated our proposed formulation extensively through physics simulation. Using a hierarchical generative model, we showcase that an embodied artificial intelligence system, a humanoid robot, can autonomously complete a complex task requiring a holistic use of locomotion, manipulation and grasping: the robot adeptly retrieves and transports a box, opens and walks through a door, kicks a football and exhibits robust performance even in the presence of body damage and ground irregularities. Our findings demonstrated the efficacy and feasibility of human-inspired motor control for an embodied artificial intelligence robot, highlighting the viability of the formulized hierarchical architecture for achieving autonomous completion of challenging goal-directed tasks.

Humans can control their bodies to produce intricate motor behaviours that align with their objectives, for example, navigating in an environment with a mixed sequential use of legs and hands in a coherent manner. These tasks require the coordination of multiple processes, including motor planning and execution[1]. To realize this coordination, human motor control unfolds at nested timescales at different levels of the neuronal hierarchy[2,3], for example, a high-level plan to arrive at a particular place can entail multiple, individual, reflexive low-level limb movements for walking. In the areas of robotics, hierarchical elements have been applied to control systems to achieve diverse motor behaviours[4]. The core principles to achieve human-like motor control have been derived and summarized previously[5], by relating these elements to the human nervous system.

In robotics, past research has been conducted to achieve similar capabilities as humans, such as assembly in aircraft manufacturing[6], space missions[7], as well as the computational model of active inference for robust robot behaviours[8]. These have been achieved by using mainly three approaches: human commands, planning and learning.

While human commands have been used for disaster response[9] or installation in construction works[10]—the high-level commands are provided by a human, either via tele-operation[9] or by a predefined task sequence[10]. In this paradigm, the autonomous execution of a task builds

[1]Embodied AI Lab, Intel Labs, Munich, Germany. [2]Wellcome Centre for Human Neuroimaging, Queen Square Institute of Neurology, University College London, London, UK. [3]VERSES Research Lab, Los Angeles, CA, USA. [4]Department of Computer Science, University College London, London, UK. [5]These authors contributed equally: Kai Yuan, Noor Sajid. ✉e-mail: alex.li@ucl.ac.uk

Table 1 | Summary of the key principles of hierarchical motor control[5], with exemplar realizations in human motor control and our robotic system

Principle	Description	Hierarchical generative models	Human motor control	Our robotics system for autonomous operations
Information factorization	Different information is processed by distinct sub-systems.	Factorized distribution of appropriate latent states within the generative model.	Different sensory signals are routed to different parts in the hierarchy, for example, what and where streams. These neuronal pathways can be characterized as factorized states responsible for sub-systems.	Only task-relevant sensory signals are used by individual levels, with irrelevant states hidden across levels. This speaks to an explicit factorization of sensory signals and which parts of the system have access to them.
Partial autonomy	Lower hierarchical levels can semi-autonomously produce outputs with minimum input from levels above.	The result of factorizing state space into multiple levels can independently accomplish sub-goals at a (relatively) fast temporal scale.	Semi-autonomous coordination of joint movement at lower levels (that is, brainstem and spinal cord). These operate at a faster temporal scale and do not require continuous input for higher levels.	Full autonomy and stability guaranteed at individual levels. Explicitly, we introduce stable mid-level and low-level motions for random higher-level inputs. This ensures that lower levels can independently perform fast movements.
Amortized control	Re-execute appropriate behaviours rapidly using learnt movements.	Learnt probability distributions that parameterize this generative model can be used for amortized control. That allows for habitual control based on previously learnt distributions.	The cerebellum is responsible for amortized control of deliberative and goal-directed behaviours, evoking fast habitual control for repeated actions.	The system learnt policies (that is, action-state mappings) that provide habitual control for rapidly re-executing appropriate actions.
Multi-joint coordination	Degenerate coupling of different components operating as a whole for motor control.	Result of state factorizations that introduce flexible mapping across and within each level.	Different neuronal ensembles have distinct influences, for example, the red nucleus controls movements of the arms. Much like factorized states, these neuronal ensembles come together to produce intricate movements.	The system is equipped with multiple sub-structures (or policy mappings) that are responsible for specific actuator movement. Together these come across, and within levels, to produce particular motor movements.
Temporal abstraction	Abstraction of time across hierarchical levels.	A feature of hierarchical generative models, where higher levels evolve slower than and constrain the level below.	Different levels evolve at different temporal and spatial scales, with the primary motor cortex responsible for planning (slow timescale) and spinal cord responsible for generation (fast timescale)	The three levels of the system evolve at different temporal scales, much like any hierarchical generative model. The high-level planning is at a slow timescale, mid-level stability control at medium timescale and low-level joint control at a fast timescale.

We omit the principle of modular objectives here (sub-systems trained to optimize specific objectives distinct from the global task objective) because a factorized generative model architecture leads to distinct factor specific objectives at each level in the hierarchy.

and relies on the use of planning explicit task sequences, which uses limited sensory feedback to replan online. Such an approach is thus not yet fully autonomous and is vulnerable to uncertainties when the environment is likely to change during the interaction.

For planning methods, such as trajectory optimization[11] or task planning[12], a model of the environment is needed to optimize a motion sequence. Such a planning framework passes commands in a top-down approach and has a separation between planned motions and the control of their executions. Consequently, the planning framework unilaterally connects with the control and lacks having feedback from the low-level layer[9]. Therefore, this approach is restricted to a limited range of scenarios, where the required execution is close to the ideal planning, for example, quasi-static, kinematic motions or well-defined environments. However, for situations where the environment model deviates from the real world, feedback is indispensable and is required for corrective actions to counterbalance changes that are not planned beforehand. Since the control is responsible for execution and interaction with the environment, the lack of feedback from the lower control layers prevents a wider generalization to other environments; hence limiting the applicability with respect to autonomous behaviours and human-level motor control.

To overcome the aforementioned limitations, learning approaches—such as hierarchical reinforcement learning[13]—is an alternative approach to accomplish tasks that require solving a discrete sequence of sub-tasks in a close-loop fashion, such as path-following for quadruped locomotion[14], interactive navigation with mobile manipulators[15] and hierarchical navigation tasks[13]. The remaining challenges[16] lie inter alia in finding the right levels of abstraction, and how to find a proper hierarchical structure with meaningful sub-behaviours. This is especially challenging for robotics, where the state and action space is complex, and behaviours are abstract and often hard be quantified explicitly. Hence, hand-crafted sub-behaviours, such as the theory of options used in Sutton et al.[17], prevent adequate exploration during reinforcement learning which is needed for autonomous operations.

This study investigates and presents five core principles of human motor control, based on which we formalize the design of a framework that generates autonomous behaviours. Our proposed hierarchical generative model adheres to the core principles of hierarchical motor control[5], and the resulted capability can tackle several challenges that hierarchical reinforcement learning has not yet overcome. Compared to hierarchical reinforcement learning, our ensuing hierarchical control structure offers the possibilities to (1) create a transparent and flexible approach to interpret and implement robotic decision-making, (2) roll-out individual policies inside the hierarchical structure and improve their overall performance and (3) identify and mitigate the cause of performance deficits.

This work achieves human-level motor control by pursuing the notion that structural dependencies (cf., interregion communication as observed in human motor control[5]) are necessary for autonomous robotic systems to optimize and adapt future actions in uncertain environments. Human motor control is generated through nested hierarchies comprising distinct, but functionally interdependent,

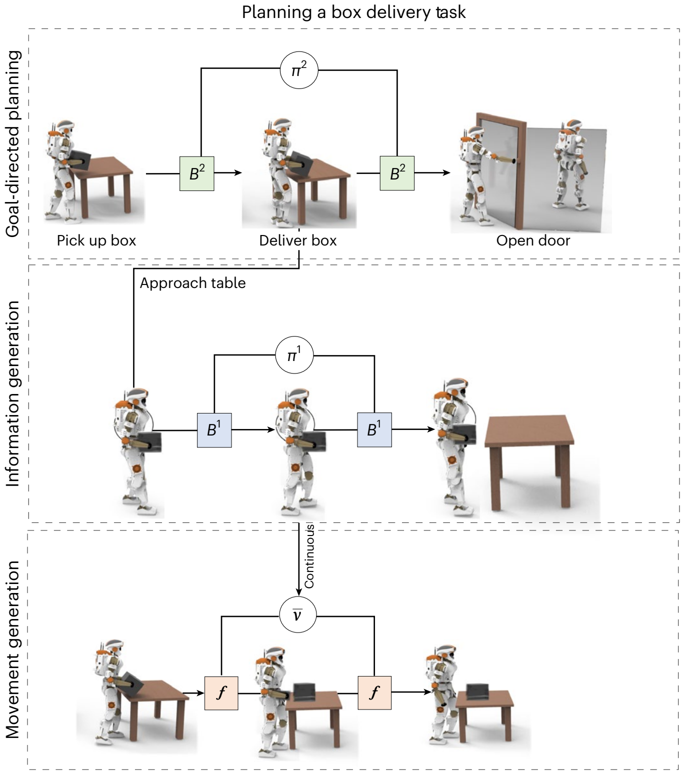

Fig. 1 | Pictorial representation of a hierarchical generative model for moving boxes. A generative model represents the conditional dependencies between states and how they cause outcomes. For simplicity, we express this as filled squares that denote actions, and circles that represent action sequences. The key aspect of this model is its hierarchical structure that represents sequences of action over time. Here, actions at higher levels generate the initial actions for lower levels that then unfold to generate a sequence of actions (cf., associative chaining). Crucially, lower levels cycle over a sequence for each transition of the level above. It is this scheduling that endows the model with a deep temporal structure. Particularly, planning (first row; highest level) to 'deliver the box' generates the actions for the information coordination level (second row; middle level) that is, 'movement towards the table'. This in turn determines the initial actions for movement generation (third row; lowest level) of arms to 'place the box' on the table. Here, a single action is generated at each timestep by sampling from action sequences (that is, sequential policies) that are generated up to a specified time horizon. π^1, mid-level action sequence; π^2, high-level action sequence; B^1, mid-level action; B^2, high-level action; f, joint torque actions; v, joint torque action sequence.

processing structures, for example, from the motor cortex to the spinal cord down to neuromuscular junctions[18]. These nested hierarchies can be interpreted as a hierarchical generative model[5,19].

Our model is a particular instantiation of such hierarchical motor control models, as contrasted by the prior studies[8,16,20,21]. To develop further, our extension has introduced and incorporated multi-level planning, asymmetric interregion communication and temporal abstraction analogous into the computational models of human motor control[5,22].

In this work, we characterize motor control as an outcome of a learnt hierarchical generative model; in particular, generative models that include the consequences of action. This proposal inherits from hierarchical functional organization of human motor control and ensuing planning as inference[21,23,24], active inference[25–28] or control as inference[29,30]. Briefly, hierarchical generative models are a description of how sensory observations are generated, that is, encodings of sensorimotor relationships relevant for motor control[31,32]. Importantly, this gives for free the five core principles of hierarchical motor control introduced in ref. 5. See Table 1 for further details.

This hierarchical formulation can facilitate multi-level planning that operates at different levels of temporal and spatial abstraction[5,32] (Fig. 1). This follows from the functional integration of separate planning (that is, choosing the next appropriate actions), motor generation (that is, executing the selected actions) and control (that is, realizing high-level plans as motor movements), as provided by the hierarchical generative model[32]. As a result, the requisite architecture can be considered as a series of distinct levels, where each provides appropriate motor control[33] (Fig. 1). In our construction, the lowest level predicts the proprioceptive signals—generated using a forward model of the mechanics—and the kinetics that undergirds motor execution. This kinetics can be regarded as realizing an equilibrium position or desired set point, without the explicit modelling of task dynamics (cf., the equilibrium point hypothesis[34]). The level above generates the necessary sequence of fixed points that are realized by the lower level. This sequence speaks to the stability control that a human has over limbs, to perambulate in an upright manner over, for example, the centre of gravity. The highest level then pertains to planning[35], and different states represent endpoints of an agent's plan, for example, move a box from a table to another.

To validate our proposition, we introduce a hierarchical generative model for autonomous robotics. It enables context-sensitive, robust task abilities by combining spatial-temporal levels and state-of-the-art tools (that is, reinforcement learning, model-predictive control and impedance control). Our model has three distinct levels for planning and motor generation, emulating a simplified functional architecture of human motor control. Importantly, each level comprises separate but functionally integrated modules, which have partial autonomy supported by asymmetric interregional communication[36], that is, the lower levels can independently perform fast movements. Such a structure provides a flexible, scaled-up construction of a hierarchical generative model, using established robotic tools (Methods). The ensuing levels in the model hierarchy were optimized sequentially and evolved at different temporal scales. However, only the middle level planner had the access to state feedback, which allowed for a particular type of factorization (that is, functional specialization) in our generative model. We reserve further details in the later sections.

The Article is organized as follows. In the Results section, we demonstrate that our (implicit) hierarchical generative model for motor control, which entails a bidirectional propagation of information between different levels of the generative model, can perform tasks remarkably similar to humans. Here, an implicit hierarchical generative model refers to a forward model whose explicit inversion corresponds to control as inference (without the need for an inverse model). In the Discussion section, we discuss the effectiveness of our hierarchical generative model, how it may benefit potential applications, and provide an outlook for future work. Lastly, in the Methods section, we provide details of our implementation.

Results

Our implicit hierarchical generative model enables a robot to learn how to complete a loco-manipulation task autonomously in simulation. We validate this model in three distinct scenarios: (1) a sequential task with two-step decision-making that involves moving a box from one table to another and opening a door by pressing a button (Fig. 1); (2) transporting a box between conveyor belts and activating the second belt by pushing a button (Fig. 2a); and (3) executing a penalty kick by approaching and kicking a football into a goal (Fig. 2b). The learned policy demonstrates generality and robustness to uncertainty (Fig. 3a–e), while evincing the core principles of hierarchical motor control.

Our implicit hierarchical generative model can successfully and autonomously achieve locomotion, manipulation and grasping movements like humans, and solve all these complex tasks coherently with internal consistency. In contrast, we demonstrate how a flat

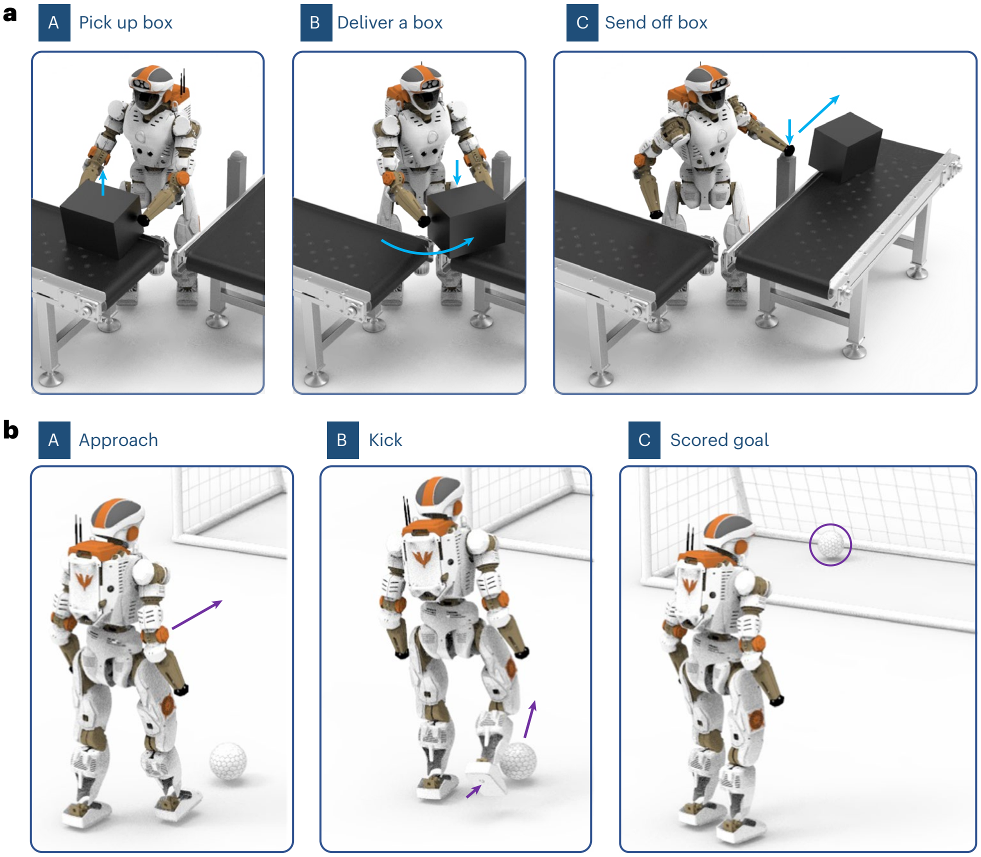

Fig. 2 | Manipulation and locomotion tasks to validate the hierarchical generative model. a, A manipulation task, where the robot picks up the box (A), delivers it (B) and finally sends it off by activating the button (C). **b**, A penalty kick, where the robot approaches (A) the ball and kicks it into the goal (B and C).

architecture fails (Supplementary Fig. 6) in this regard (Supplementary Information Section 5).

The high-level policy determines the action sequence necessary for task completion and sends commands to the lower levels responsible for limb coordination and joint control. Here, independently, the locomotion policy can facilitate adaptation to perturbations, for example, recovering from pushes or locomotion over different types of terrains. Contrariwise, the low-level joint controller provides robustness to sudden and hard contacts with the ground absorbing high-frequent impacts.

To assess the robustness and generality of this hierarchical scheme, we introduced several perturbations that were not encountered during training (Fig. 3). First, we introduced external perturbation by placing obstacles (that is, 5 kg box, Fig. 3a) in front of the robot and pushing its pelvis (Fig. 3b). The mid-level locomotion policy withstood both perturbations, moved the obstacle out of the way and took a step to recover balance after the push. To test the performance further, we modified the environment with unseen conditions by adding a 5° inclined surface (Fig. 3c) and a low-friction glass plate (friction coefficient of 0.3, Fig. 3d) in front of the door. The robot could complete the task after each perturbation. More interestingly, we lesioned the robot by amputating its right foot (Fig. 3e). Despite this handicap that was never encountered and with only a stump touching the ground in place of its right foot our hierarchical control was sufficiently robust to deal with this situation and the robot was able to keep balance and complete the task.

Next, we evaluate whether the ensuing control architecture satisfies the key principles of hierarchical motor control (Table 1) that underwrite robust task performance.

Information factorization

In this system, factorization exists across model levels and policy controls, each responsible for a particular sort of information processing. This factorization ensures that external perturbations have minimum impact on task performance.

Since the information factorization defines the role for each sub-system, thus, any failures in performance can be isolated and fine-tuned for future tasks. For example, if the robot falls over while walking to a goal, the locomotion policy can be identified as the root cause, and hence improving the locomotion policy will resolve the issue without needing to modify the high-level planner or the manipulation policy. Further examples include oscillation of the robot limbs, which can be attributed to the low-level joint control; or walking in the wrong direction, which was due to the command from the high-level policy.

From a theoretical perspective, factorization of this sort corresponds to the structure of the generative model that can be decomposed into factors of a probability distribution (in physics and probabilistic inference, this is called a mean field approximation). Almost universally, this results in certain conditional independencies that minimize the complexity of model inversion; namely, planning as inference or control as inference[21,23,26]. This is important because it precludes overfitting and ensures generalization. From a biological perspective, this kind of factorization can be regarded as a functional segregation that is often associated with modular architectures and functional specialization in the brain[36].

Partial autonomy

The system is designed with partial autonomy, that is, minimum interference or support from other levels. Specifically, we implement a

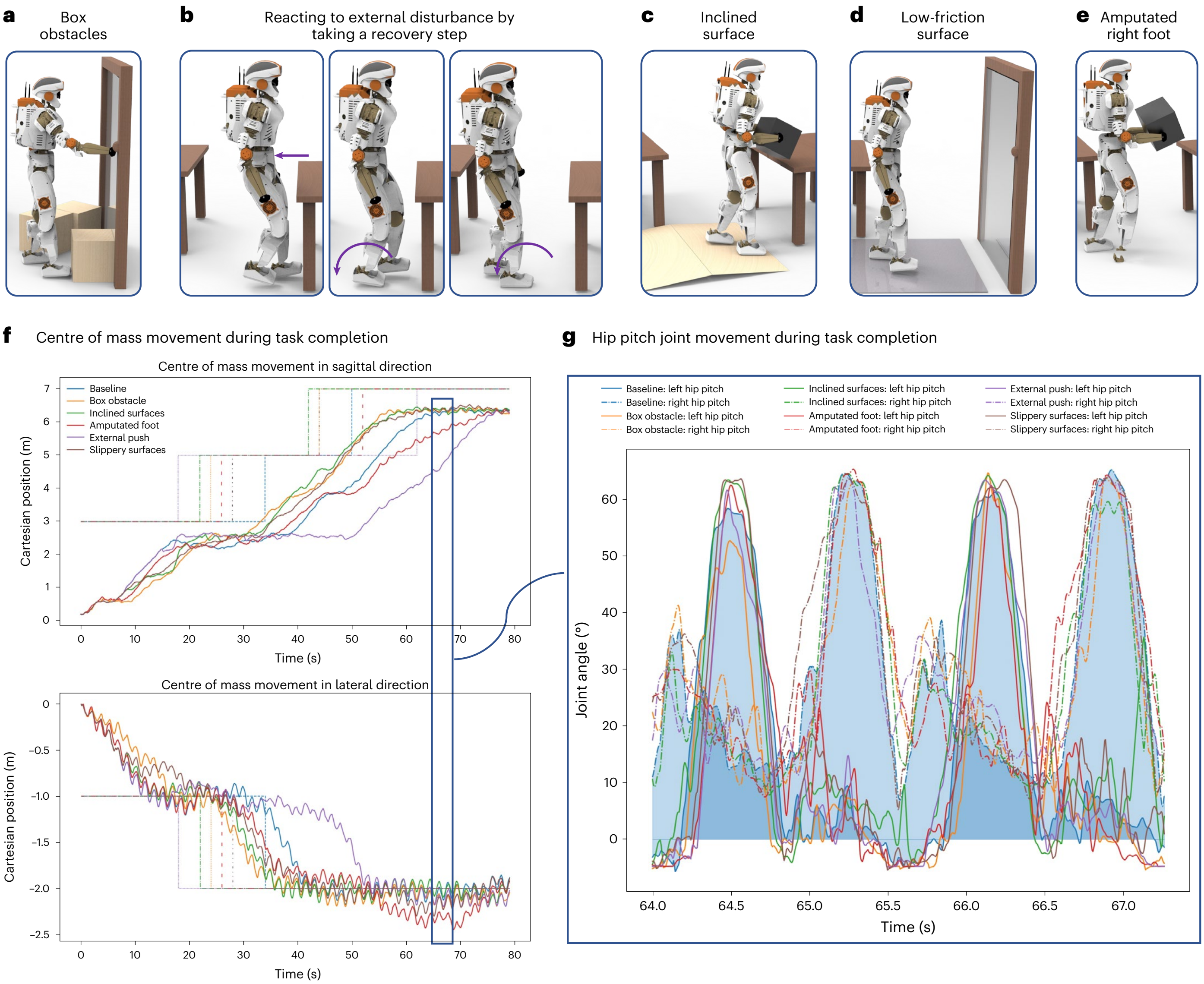

Fig. 3 | Robustness of the system in the presence of perturbations and environmental changes. a–**e**, Illustration of how the robot completes the task in perturbation test scenarios that it has not encountered during training and demonstrates the robustness of our proposed method. From left to right, we place 5 kg box-obstacles in front of the robot (**a**), push the robot from the front (**b**), alter the floor with an inclined (**c**) and slippery (**d**) surface and lesion the robot by removing the right foot (**e**). **f**, Sagittal and lateral CoM movement is shown under different perturbations demonstrating the amortized control. **g**, Hip pitch joint movement, which has the biggest effect on the motion during biped locomotion. The hip pitch joint motion is used to show how the policy adapts to the perturbation and rapidly re-executes a motion to counteract the perturbation.

clear separation between the highest and intermediate levels, though they are learned together. This is particularly relevant because the high-level planning level could send unrealizable action sequences to the mid-level stability controller. Without partial autonomy, the robot can become unstable and unable to learn to move appropriately, given such random or potentially unstable high-level commands.

Figure 4 illustrates a case when the robot is provided with random commands to both the arms and legs. This causes the robot to walk in random directions (Fig. 4a) and the arms move around randomly (Fig. 4b). Despite imperfect motion tracking, the robot does not fall over and can complete the tasks despite incoherent intentions.

Amortized control

After training, the robot engages in amortized control with the ability to re-execute appropriate behaviours rapidly using previously learnt movements. We observed this behaviour in the baseline and perturbed task settings (inset trajectories in Fig. 3f), where the amortized locomotion policy was used to complete the task without the need of additional learning.

Multi-joint coordination

The robot has multiple sub-structures that are responsible for specific controls and work together in different ways to generate motor movements. Supplementary Fig. 5a demonstrates this multi-joint coordination when pressing the button to open the door in the presence of an obstacle (Task 2). To achieve this, the right arm motions had to coordinate appropriately according to the initial hand position. Also, the shoulder roll (Supplementary Fig. 5b, orange line) and elbow (Supplementary Fig. 5b, red line) had to adjust and adapt differently from the baseline. Explicitly, these do not yield a fixed motion, instead, the manipulation policy coordinates these joints based on the centre of mass (CoM). Therefore, during the baseline reaching motion, the arms

https://doi.org/10.1038/s42256-023-00752-z

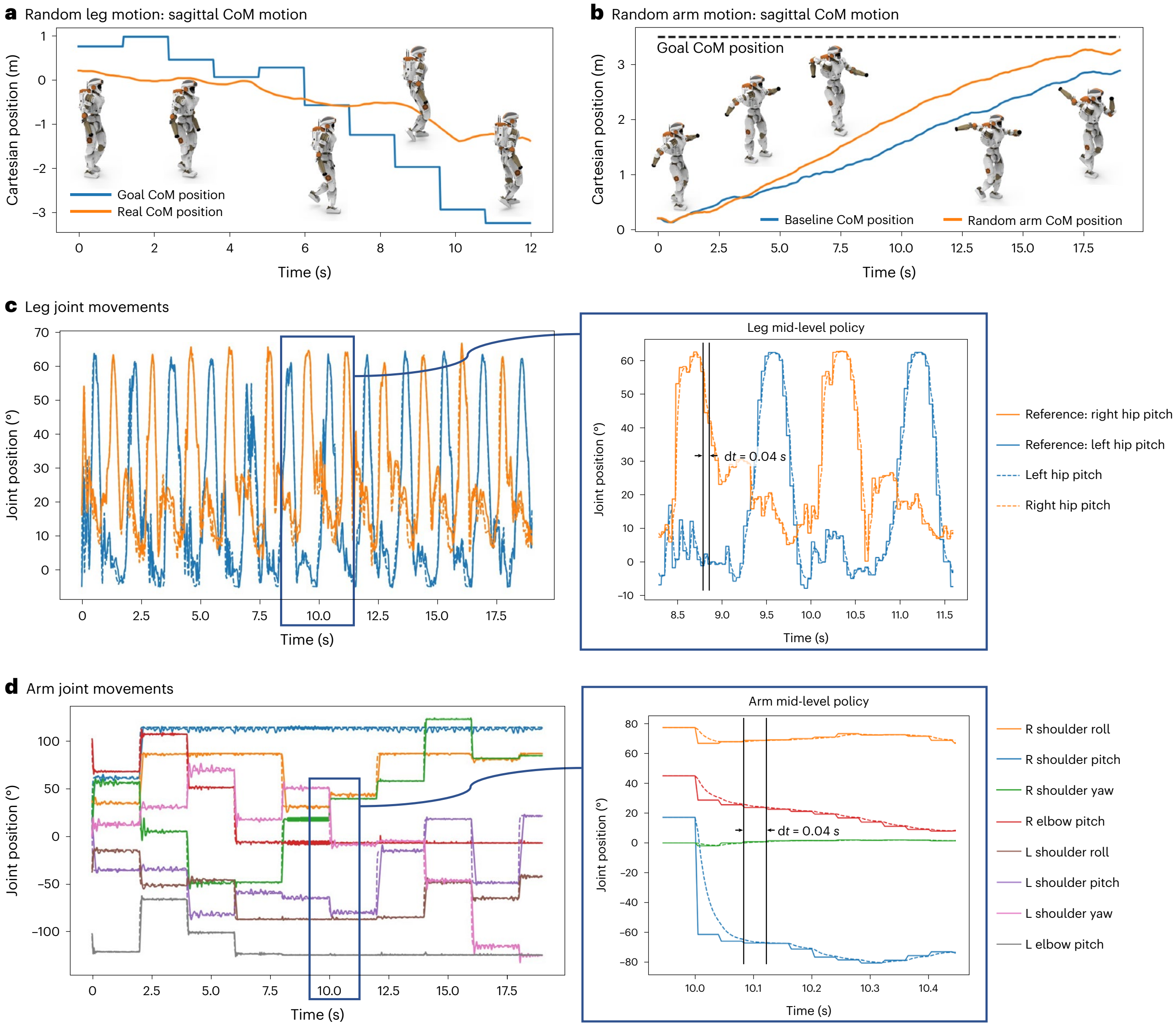

Fig. 4 | State and temporal dynamics of the robot during task performance with random high-level commands. a,**b**, Sagittal motion of the CoM while following random leg (**a**) and arm (**b**) commands, respectively. From the robot snapshots corresponding to the time they're shown, the partial autonomy of the mid-level stability controllers can be seen, that is, a good performance of the individual levels despite random and fast-changing command inputs. **c**,**d**, Leg (**c**) and arm (**d**) movements, respectively. Here, the separation of temporal scales during planning can be seen, where the high-level commands are provided at 0.5 Hz and the mid-level commands are realized at 25 Hz. The joint commands are realized at 500 Hz on the joint actuators. In the inset plots of **c**,**d** the joint position trajectories evolve similarly as postulated in the equilibrium point hypothesis.

move differently than that in the case of an obstructed box, where the CoM is in a different position because boxes are obstructing the door.

Temporal abstraction and depth

By design ('Implicit generative models'), the three system levels evolve at different temporal scales. Figure 4 illustrates these distinct scales as the robot perambulates. The highest policy level has a slow timescale of 0.5 Hz (Fig. 4a). This allows the lower levels to carry out the command in a partially autonomous way, that is, uninterrupted. Conversely, the mid-level stability control of limbs has a faster timescale at 25 Hz (inset trajectories of Fig. 4c,d). This is needed to generate rapid predictions for the locomotion and manipulation policies. Finally, the low-level joint control executes these control commands at a frequency of 500 Hz on the actuator level.

Discussion

Hierarchical generative models of motor control

Our hierarchical generative model is an abstract computational representation of the functional architecture of human motor control (Fig. 5). Here, we briefly discuss its computational neuronal homologues, focusing on predictions of primary afferent signals from muscles, and consider the corresponding principles for human motor control. The inversion of forward models that underwrite human motor control generates continuous proprioceptive predictions at the lowest level and propagates information to the highest levels that are responsible for planning. Accordingly, our formulation provides an implicit generative model that can be used by a model-based robotic agent, including reinforcement learning and active inference[37], to infer its environment dynamics.

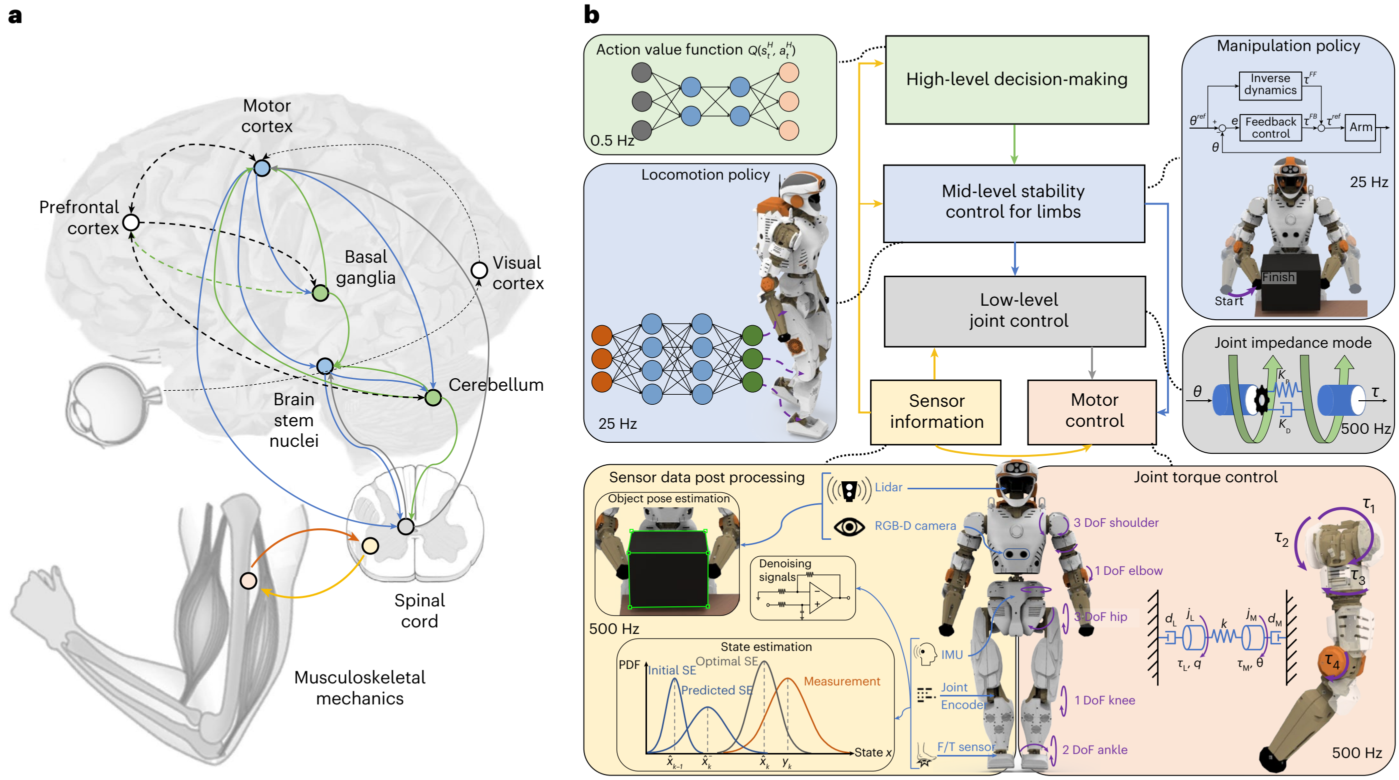

Fig. 5 | Algorithmic realizations of hierarchical control as inference. **a**, Schematic of a (high-level) generative model that underwrites human motor control. **b**, The implicit generative model for a robotics system. The green nodes in **a** and green boxes in **b** refer to the highest levels of human motor control and our implicit generative model, respectively. In the generative model, high-level decision-making is realized as a neural network learned through deep reinforcement learning. The blue nodes in **a** correspond to the middle level of human motor control and the blue boxes in **b** are intermediate level realizations, implemented as a deep neural network policy learned through deep reinforcement learning for locomotion and an inverse kinematics and dynamics policy for manipulation. On the lowest level, depicted in grey nodes and boxes, a joint impedance controller calculates the torques required for the actuation of the robot. Yellow and light red denote sensor information and motor control, respectively. For clarity, we limit our exposition to key regions in **a**, based on prior literature, where these are drawn using the solid lines. The dotted lines represent the processing of a separate outcome modality for human motor control, that is, the visual input. Lastly, the prefrontal cortex is connected via the dashed lines to denote its supporting role during human motor control. Dotted lines in **b** indicate the realizations of the corresponding principles, while dashed lines indicated message parsing. Please refer to 'Implicit hierarchical generative model for a robotics system' for the algorithmic implementation of **b**. SE, state estimation; PDF, probability density function; IMU, inertial measurement unit; τ, torque of individual joints; d_L and d_M, damping parameters; j_L and j_M, inertia parameters; k, stiffness parameter; F/T, force/torque; τ_L and τ_M, torques on L and M, respectively.

The generative model's lowest and fastest level includes the spinal cord and the brainstem. These areas are responsible for evaluating the discrepancy between the proprioceptive inputs (primary afferents) and descending predictions of these signals. This discrepancy (namely, prediction error) drives the muscle contraction via classical motor reflexes and their accompanying musculoskeletal mechanics[38,39]. On this view, classical reflexes are realised by equilibrium or setpoints from descending predictions of proprioceptive input[40–42]. This is instantiated in our model at the low-level joint control, which receives current joint position and sensor information to calculate the desired torque necessary for achieving a targeted and predicted position (supplied by the mid-level controller) via the motor control. Here, the joint controller has partial autonomy to compute the desired torque, similar to neuronal ensembles (that is, the red nucleus) controlling low-level arm movements.

At an intermediate level, one could consider the role of the cerebellum. The cerebellum receives ascending inputs from the spinal cord, and other areas, and integrates these to fine-tune motor activity. In other words, it does not initiate movement, but contributes to its coordination, precision and speed, through a fast non-deliberative mode of operation. Therefore, it can be thought of as being responsible for amortized (habitual) control of motor behaviour, which is characterized by subcortical and cortical interactions[43–45]. The cerebellum receives information from the motor cortex, processes this information and sends motor impulses to skeletal muscles (via the spinal cord). The mid-level of our generative model is used for similar coordination and stability control of locomotion and manipulation policies that yield multi-joint coordination. It fine-tunes pelvis and hand targets, given descending policy from the higher level, to determine exact joint location (measured in radians). Like the cerebellum[46], this level can coordinate multiple joint movements semi-autonomously over time.

Higher levels of the generative model include the cerebral cortex, among other neuronal systems. The cortex has access to factorized sensory streams of exteroceptive, interoceptive and proprioceptive signals (for example, visual, auditory, somatosensory, etc) and can coordinate, contextualize or override habitual control elaborated in lower levels. Specifically, the primary motor cortex is responsible for deliberative planning, control and execution of voluntary movements: for example, when learning a new motor skill before its habituation or amortization.

These are instantiated as ascending tracts that cross over to the opposite side of the system, for example, the spinocerebellar tract that is responsible for sending sensory signals regarding arms and limb movements. Conversely, descending tracts carry appropriate motor information to the lower levels, for example, the pyramidal tracts responsible for sending conscious muscle movements. The role of the cortex is instantiated at the highest level of our model, with access to processed sensor information to aid decision-making. Specifically, we introduce asymmetric interregion connections with connections from the low-level sensor information to this high level, and from this high level to the mid-level stability control. Anatomically, these correspond to extrinsic white-matter connections in the brain which, in predictive coding and variational message passing schemes, are responsible for belief updating and planning as inference[32].

Future directions

By providing robots with a new level of task autonomy for both locomotion and manipulation skills—with appropriate triage procedures—humans can be relieved from the necessity of sending low-level commands for control and decisions to robots, for example, foot and hand contacts, as commonly seen in a shared autonomy and semi-autonomous paradigms. Consequently, we can overcome potential limitations coming from human errors and the reliance on the communication bandwidth. One example is the large number of robots that fell during the DARPA Robotics Challenge Finals in 2015 (ref. 47), where robots had very little autonomy and relied on close supervision by humans, such that the whole scheme became error-prone and vulnerable, which suffered from erroneous human decision-making, lack of local robot autonomy against environmental uncertainties and disturbances and so on.

With this goal in mind, we will explore the future implementation of our hierarchical generative model on physical robots. Given the extensive validation of our current work in physics simulations, deploying the existing model and its components on real-world robots would be possible by using additional simulation to reality (sim-to-real) techniques to bridge the sim-to-real gap. To tackle this challenge, we plan to use techniques that show potential for a seamless sim-to-real transfer, minimizing the necessity for extensive adaptations. Particularly, established methods such as domain randomization and action filtering can be used, which are proven to be effective in enabling a successful sim-to-real transfer[48,49].

Future work will evaluate the use of hierarchical generative models under more nuanced planning objectives, and different autonomous robotics systems. Because of the modular factorization of the implicit hierarchical generative model, policies at various levels can be replaced and further upgraded with an alternative controller or a learned policy. For example, replacing our Q-learning planner with more sophisticated schemes which are designed to handle aleatoric and epistemic uncertainties (that is, expected free energy[50–52]). This type of future work can improve the performance in volatile conditions[51].

Furthermore, robustness can be evaluated through robotic neuropsychology[53] that is, introducing in-silico lesions by perturbing various approximations and policies and investigating their effect on the ensuing inference and behaviour. These computational lesions can be introduced in both simulated and physical robots, where lesions of this sort can change functional outcomes. For example, perturbations on the minimum-jerk optimization solution (that is, computational lesion) at the mid-level stability control would lead to cerebellar tremors for the arms.

Methods

Here we present the hardware implementation for inverting the implicit hierarchical generative model for autonomous robot control. The specification of the robot platform can be found in Supplementary Table 1. First, we detail the task that is completed autonomously by inverting the generative model, that is, using the model to predict sensor inputs and using actuators to resolve the ensuing (proprioceptive) prediction errors. Next, we elaborate on the details of generative model including high-level decision-making, mid-level stability control and low-level joint control.

Please refer to Supplementary Information Section 6 for additional notes on the implementation.

Tasks of interest

To demonstrate how inversion of a hierarchical generative model solves complex tasks that require a particular sequence and coordination of locomotion and manipulation skills, we designed a task that demanded both coordination of limbs and reasoning about the sequence of actions. This task comprised four sub-tasks (Supplementary Fig. 1): picking up a box from the first table, delivering the box to the second table, opening the door and walking to the destination or goal position. To complete the task, all the sub-tasks had to be carried out in an exact sequence.

Our proposed framework allowed the robot to learn successful task completion through interactions with the environment in simulation. This was achieved by designing a reward (or utility) function for the high-level policy, such that cumulative maximization of reward leads to task completion ('High-level decision-making'). For the mid- and low-level policies, a combination of control policies and imitation learning was used.

Implicit hierarchical generative model for a robotics system

Following the key principles of hierarchical motor control in Table 1 and the generative model in Fig. 1, we constructed a generative model for a humanoid robot comprising three levels: high-level decision-making, mid-level stability control and low-level joint control. The structure of the hierarchical generative model is shown in Fig. 5b. This hierarchical architecture rests on conditional independencies that result in factorized message passing between hierarchical levels.

Here, the temporal depth and structure of motor planning rests on specifying a hierarchical generative model, where level-specific policies are evaluated at different timescales. In this setting, each level assimilates[54] evidence from the level below, in a way that is contextualized or selected by (slow) constraints, afforded by the level above. A summary of the implicit hierarchical generative model for a robotics system can be seen in Supplementary Table 4 and Fig. 4.

The (implicit) hierarchical generative model is instantiated as:

$$
\begin{aligned}
& p\left(o_{0:T_1}, s^{1:3}_{0:T_n}, a^{1:3}_{0:T_n-1}\right) \\
& = \underbrace{\textstyle\prod_{n=1}^{N} p(s^n_0)p(a^n_0)}_{\text{state and action prior}} \\
& \times \textstyle\prod_{t_3=1}^{T_3}\prod_{t_2=1}^{T_2}\prod_{t_1=1}^{T_1} \Bigg[\underbrace{p\left(s^3_{t_3} | s^3_{t_3-1}, a^3_{t_3-1}, o_{t_1-1}\right)}_{\text{level 3 transitions}} \underbrace{p\left(a^3_{t_3} | s^3_{t_3-1}\right)}_{\text{level 3 policy}} \\
& \times \underbrace{p\left(s^2_{t_2} | s^2_{t_2-1}, a^2_{t_2-1}, o_{t_1-1}, a^3_{t_3}\right)}_{\text{level 2 transitions}} \underbrace{p\left(a^2_{t_2} | s^2_{t_2-1}\right)}_{\text{level 2 policy}} \\
& \times \underbrace{p\left(s^1_{t_1} | s^1_{t_1-1}, a^1_{t_1-1}, o_{t_1-1}, a^2_{t_2}\right)}_{\text{level 1 transitions}} \underbrace{p\left(o_{t_1} | s^1_{t_1}\right)}_{\text{likelihood}} \underbrace{p\left(a^1_{t_1} | s^1_{t_1-1}\right)}_{\text{level 1 policy}} \Bigg],
\end{aligned}
$$

where outcome $o_t \in O$, state $s_{t_n} \in S$, action $a_{t_n} \in A$ and p denotes a probability distribution. The superscript $n \in \{1,2,3\}$ indicates the level of the state s^n or action a^n, with $N = 3$ being the highest level and $n = 1$ being the lowest level. The subscript $t_n \in \{1, \ldots, T_n\}$ indicates the time at each level n evolving at different temporal scales: the highest level ($n = 3$) evolves at 0.5 Hz, the mid-level ($n = 2$) at 25 Hz and the lowest level ($n = 1$) at 500 Hz.

This temporal ordering denotes how different levels contextualize the level below: the high-level policy contextualizes the roll-out for the mid-level; mid-level policy contextualizes the low-level; and each level has access to previous outcomes. Briefly, the transition function is defined as an identity (using the previous outcome) for levels 2 and 3 until the next update (that is, 50 level 1 steps for level 2, and 1,000 level 1 steps for level 3). The pseudocode for optimizing each level can be found in Supplementary Fig. 4, along with a detailed overview of dependencies across levels.

The highest planning level, evolving at the slowest rate, selects an appropriate sequence of limb movements, which are needed to complete a particular sub-task. It decides where the hands should be and what direction to go. Practically, deep reinforcement learning is used to learn a high-level decision-making policy that generates targets (in a Cartesian space) for the mid-level stability control (cf., the equilibrium point hypothesis for human motor control[40] and active inference formulations of oculomotor control[32]).

These planning targets are realized at the level below that regulates the balance and stability of the robot during manipulation and locomotion. Manipulation is instantiated as a minimum-jerk model-predictive controller that moves the arms to the target positions provided by the high-level policy. Locomotion is implemented as a learnt mid-level policy, via deep reinforcement learning, that coordinates legs to reach the destination predicted by the higher level. Both policies are designed to ensure that infeasible setpoints from the high level are corrected for the mid-level stability control so that only stable joint target commands are supplied to the low-level joint controller.

Despite receiving inputs from other levels, each level has partial autonomy over its final predictions and goal. Furthermore, multi-joint coordination is realized by learning a policy that coordinates all joints of legs appropriately for the current state, while the arms coordinate their joints through inverse kinematics (IK).

The low-level joint controller is instantiated as joint impedance control and tracks the joint position commands afforded by the mid-level stability controller. Based on tuned stiffness and damping, the joint impedance control calculates the desired torque to attain target positions closely and smoothly. Lastly, the torque commands are tracked by the actuators, using embedded current control of onboard motor drivers.

Training process

The generative model was realized by implementing three levels of control in a hierarchical manner (Fig. 5): high-level decision-making, mid-level stability control and low-level joint control. All components were designed and trained separately, starting from the lowest level.

First, accurate and robust motor control needed to be guaranteed, such that the low-level joint position control could be realized. Stiffness and damping parameters were tuned to track the references accurately and compliantly, which provided the mid-level stability control. The mid-level stability control consisted of a manipulation and a locomotion policy, which were individually designed. The locomotion policy was trained to walk towards a commanded goal position, while the manipulation policy was designed to place the hands on a target position. Finally, the high-level decision-making policy was trained via deep reinforcement learning, which learnt to provide appropriate commands to these mid- and low-level policies.

Gradient-free optimization[55,56] was used to find (1) the best hyper-parameters sets for the mid-level manipulation policy and low-level joint controller and (2) network architecture for the high-level decision-making policy and mid-level locomotion policy.

High-level decision-making

We achieved high-level decision-making, the correct sequence and choices of robot actions, through training a deep neural network that approximated the action-value function $Q(s,a)$ over the environment and chose the action a that yielded the highest value in state s.

We used double Q-learning[57] to train a Q-network $Q(s,a;\phi)$, parametrized by weights ϕ, to approximate the true action-value function $Q(s,a)$. At run-time, the action a was obtained as the argument of the maximum Q-value $a = \mathrm{argmax}_a Q(s,a;\phi)$ in state s. Two separate Q-networks Q_1 and Q_2 were used for action selection and value estimation, respectively. Having two separate Q-networks has previously shown to improve training stability[57].

The network parameter ϕ_i was obtained by $\min_{\phi_i} L(\phi_i)$:

$$\min_{\phi_i} E\left[\left(r + \gamma Q_j(s', a^*;\phi_j) - Q_i(s,a;\phi_i)\right)^2\right],$$

with reward r, discount factor γ, network parameters ϕ_i and ϕ_j, Q-networks Q_i and Q_j, current state s, next state s' and best action $a^* = \mathrm{argmax} Q_i(s,a;\phi_i)$. During training, either network parameter ϕ_1 or ϕ_2 was randomly selected, trained and used for action selection, while the other network parameter was used to estimate the action-value. The tuple $(s,a,r,s') \approx U(D)$ was obtained from the experience replay by uniformly sampling from buffer D, which was updated by online action roll-out. The time horizon of the high-level decision-making system is implicitly specified with the discount factor γ that is used to calculate the return as $G_i = \sum_i \gamma r_i$. A way to interpret the discount factor with respect to planning horizon is the concept of half-life of the future reward, that is, when the current reward r_i is entering the return calculation as $\frac{1}{2} r_i$. With the standard discount factor $\gamma = 0.95$ used in this work, the policy looks ahead ~13.5 steps: $\gamma^{\mathrm{steps}} = 0.5 = 0.95^{\mathrm{steps}} \Rightarrow \mathrm{steps} = \frac{\log(0.5)}{\log(0.95)} \approx 13.5$. At a control frequency of 0.5 Hz, the prediction horizon is roughly 27 seconds.

Box delivery and opening door task. The high-level policy sent and updated the actions $a^3 \in \mathscr{A}^3 \subseteq \mathscr{R}^9$ at 0.5 Hz frequency, which were the positions in Cartesian space for the pelvis $a^3_{\mathrm{pelvis}} \in \mathscr{R}^3$, and left and right hands $a^3_{\mathrm{lh}}, a^3_{\mathrm{rh}} \in \mathscr{R}^3$. These actions a^3 were executed by the mid-level stability controller.

The states $s^3 \in \mathscr{S}^3 \subseteq \mathscr{R}^{12}$ were the vector $\mathbf{s}_{\mathrm{pelvis}} = p_{\mathrm{table}} - p_{\mathrm{pelvis}} \in \mathscr{R}^3$ from the table (origin of the coordinate system) to the current pelvis position p_{pelvis}, and the vectors $\mathbf{s}_{\mathrm{lh}} = p_{\mathrm{box}} - p_{\mathrm{lh}} \in \mathscr{R}^3$, $\mathbf{s}_{\mathrm{rh}} = p_{\mathrm{box}} - p_{\mathrm{rh}} \in \mathscr{R}^3$ from current hand positions $p_{\mathrm{lh,rh}}$ to the box's position. Lastly, three Boolean variables $o^3 \in O^3 \subseteq [0,1]^3$ were provided as the observation state when the door was open, the box was on the table or the box was being carried.

The reward terms r_i were determined based on the task completion, such as whether the robot had passed the delivery table, the arm joints were in the nominal position, the box was between the robot hands, the box was at the delivery table, the door was open and whether the robot was at the goal. The weights $w_i, i = 1,\ldots,6$ can be found in Supplementary Table 2 (top).

At each timestep, the reward r was the sum of sparse, Boolean states:

$$r = w_1 r_{\mathrm{pt}} + w_2 r_{\mathrm{jn}} + w_3 r_{\mathrm{bih}} + w_4 r_{\mathrm{bot}} + w_5 r_{\mathrm{do}} + w_6 r_{\mathrm{ag}},$$

with passed table reward r_{pt}, joints nominal reward r_{jn}, box in hand reward r_{bih}, box on table reward r_{bot}, door open reward r_{do} and at goal reward r_{ag}.

We terminated the episode early if the robot fell, or collided with itself, tables or the door. By terminating an episode early—when a sub-optimal state (for example, falling) is reached—the return is lower, and the policy is thus discouraged from entering similar sub-optimal states.

We initialized the robot in different positions in the environment, such as close to the final goal, in front of the door, or at the second table, to allow the robot to encounter such states that were hard to discover merely by exploration, as a particular sequence of actions were required to reach those states.

Penalty kick task. To perform a penalty kick, that is, approaching and shooting a ball, the high-level policy is trained similarly to Task 1. Cartesian space commands of the pelvis $a^3_{\mathrm{pelvis}} \in \mathscr{R}^3$ are generated by the high-level policy (actions $a^3 \in \mathscr{A}^3 \subseteq \mathscr{R}^3$) and executed by the mid-level stability controller.

The states $s^3 \in \mathscr{S}^3 \subseteq \mathscr{R}^4$ are the horizontal positions of the ball and pelvis. A reward $r = 1$ is given, whenever the ball surpasses the goal line, that is, a goal was scored. An episode is terminated early if the robot fell or collided with itself.

Transporting box and activating conveyor belt task. The box transportation task consists of two separate sub-tasks that need to be performed in a specific sequence: grasping a box from the first conveyor belt, transporting the box to a second conveyor belt by rotating the torso around the yaw axis, dropping the box off and sending it away on the conveyor belt by activating the button.

The action space $(a^3 \in \mathscr{A}^3 \subseteq \mathscr{R}^7)$ of the high-level policy includes Cartesian space commands of the left and right hands $a^3_{\mathrm{lh}}, a^3_{\mathrm{rh}} \in \mathscr{R}^3$ and torso yaw joint position commands $a^3_{\mathrm{ty}} \in \mathscr{R}^1$. These actions a^3 were executed by the mid-level stability controller.

The states $s^3 \in \mathscr{S}^3 \subseteq \mathscr{R}^{14}$ consists of the joint positions of the arms $(s_{\mathrm{joints}}{}^3 \in \mathscr{R}^8)$, Cartesian positions of the box $(s_{\mathrm{box}}{}^3 \in \mathscr{R}^3)$ and three Boolean values indicating whether the box is in contact with the hands, table and whether the button is pushed.

A reward is given for three cases: (1) box in hands (r_{bih}), (2) box on conveyor belt (r_{boc}) and (3) button pushed (r_{pb}) while the box is on the second conveyor belt. The resulting reward function with weights w_i (Supplementary Table 2 bottom) are:

$$r = w_1 r_{\mathrm{bih}} + w_2 r_{\mathrm{boc}} + w_3 r_{\mathrm{bp}}.$$

Mid-level stability control

The mid-level stability control level consisted of two components: the manipulation policy was realized as a model-predictive control (MPC) scheme for the arms, and a locomotion policy was learned through deep reinforcement learning for the legs.

Manipulation policy. As input into the policy, the manipulation policy received Cartesian target positions for the hands $a^3 = [a_{\mathrm{lh}}, a_{\mathrm{rh}}]$ from the high-level policy, current Cartesian position of the hands $s^2 = [p_{\mathrm{lh}}, p_{\mathrm{rh}}] \in \mathscr{S}^2 \subseteq \mathscr{R}^6$, and current, measured joint angles of the arms $o \subseteq \mathscr{R}^8$. The output $a^2 = q^d_{\mathrm{arms}} \in \mathscr{A}^2 \subseteq \mathscr{R}^8$ of the manipulation policy was target joint positions q^d_{arms} of the arms to the low-level joint controller.

The manipulation policy consists of two parts (flow diagram in Supplementary Fig. 2): MPC that generated a stable, optimal trajectory in Cartesian space and IK[58] that transformed desired actions from the Cartesian space to the joint space.

To provide the smoothest possible motions for the hands, we formulated the optimal control problem as the minimum-jerk optimization, while satisfying dynamics constraints on the hands. The optimal trajectory was then implemented in an MPC fashion. The MPC control applied the first control input of the optimal input trajectory and then re-optimized based on the new state at the next control loop[59]. In this way, MPC successively solved an optimal control problem over a prediction horizon N and achieved feedback control, while ensuring optimality.

For the hand position p, an objective function J was designed to minimize jerk $\dddot{p}$ (the input u of the system) with final time t_{f}:

$$J = \frac{1}{2}\int_0^{t_{\mathrm{f}}} \left(\frac{\mathrm{d}^3 p(t)}{\mathrm{d}t^3}\right)^2 \mathrm{d}t = \frac{1}{2}\int_0^{t_{\mathrm{f}}} u(t)^2 \mathrm{d}t.$$

The minimum-jerk MPC (MJMPC) solved the following constrained optimization problem at every timestep at a frequency of 25 Hz:

$$\begin{aligned}
\min_{u(t)} \quad & \frac{1}{2}\int_0^{t_{\mathrm{f}}} u(t)^2 \mathrm{d}t \\
\text{subject to} \quad & \frac{\mathrm{d}^3 p(t)}{\mathrm{d}t^3} = u \\
& [p(0), \dot{p}(0), \ddot{p}(0)] = [p_0, \dot{p}_0, \ddot{p}_0] \\
& [p(t_{\mathrm{f}}), \dot{p}(t_{\mathrm{f}}), \ddot{p}(t_{\mathrm{f}})] = [p_{\mathrm{f}}, \dot{p}_{\mathrm{f}}, \ddot{p}_{\mathrm{f}}] \\
& [p_{\mathrm{min}}, \dot{p}_{\mathrm{min}}, \ddot{p}_{\mathrm{min}}] \leq [p, \dot{p}, \ddot{p}] \leq [p_{\mathrm{max}}, \dot{p}_{\mathrm{max}}, \ddot{p}_{\mathrm{max}}],
\end{aligned}$$

with initial condition $[p_0, \dot{p}_0, \ddot{p}_0]$ and terminal condition $[p_{\mathrm{f}}, \dot{p}_{\mathrm{f}}, \ddot{p}_{\mathrm{f}}]$.

The resultant Cartesian trajectory p^d, that is, the trajectory that leads from the initial hand position p_0 to the final hand position $p_{t_{\mathrm{f}}}$, from MJMPC was transformed into joint position commands q^d_{arms} through IK. More formally, IK described a transformation $T{:}\mathscr{C} \to \mathscr{Q}$ from Cartesian space $\mathscr{C}$ to joint space $\mathscr{Q}$. The joint position commands q^d were then tracked by the low-level joint position controller as described in 'Low-level joint control'. The IK ensures feasible joint configuration on the robot even if the high-level decision policy or the MPC trajectory yield infeasible setpoints.

Locomotion policy. The locomotion policy $\pi(s;\theta)$ coordinated the 12 degrees of freedom (DoF) leg joints and was instantiated as a deep neural network (network parameters θ) that received robot states s as inputs and outputs 12 target joint positions q^d_{legs} for the legs. It was trained through Soft Actor-Critic (SAC)[60], an off-policy deep reinforcement-learning algorithm.

SAC optimized a maximum entropy objective $J_{\mathrm{SAC}(\pi)}$:

$$J_{\mathrm{SAC}}(\pi) = \sum_{t=0}^{T} \mathbb{E}\left[r(s_t, a_t) + \alpha \mathscr{H}(\pi(a_t|s_t))\right],$$

with reward r, state s_t and action a_t at time t, temperature parameter α and policy entropy $\mathscr{H}(\pi)$. The parameters θ for policy π_θ were obtained by minimizing $J_\pi(\theta)$:

$$J_\pi(\theta) = \mathbb{E}\left[\log \pi_\theta(a_t|s_t) - Q_\phi(s_t, a_t)\right].$$

The action-value function $Q_\phi(s_t, a_t)$ was obtained by minimizing the Bellman residual $J_Q(\phi)$:

$$J_Q(\phi) = \mathbb{E}\Big[1/2\big(Q_\phi(s_t, a_t) - \hat{Q}(s_t, a_t)\big)^2\Big],$$

with Bellman equation $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}\left[V_\psi(s_{t+1})\right]$ and discount factor γ. The estimation of the value function V_ψ was obtained by minimizing $J_V(\psi)$:

$$J_V(\psi) = \mathbb{E}\Big[1/2\big(V_\psi(s_t) - \mathbb{E}\left[Q_\theta(s_t, a_t) - \log \pi_\phi(a_t|s_t)\right]\big)^2\Big].$$

The training procedures, including the design of reward, action space and state space, are as in ref. 61. The actions $a^2 \in \mathscr{A}^2 \subseteq \mathscr{R}^{12}$ were the joint positions q_{legs} of the 12 DoF of the legs (for each leg: three DoF for hip, one DoF for knee and two for DoF ankle). The target joint positions q^d_{legs} were tracked by the low-level joint controller (low-level joint control).

The state $s^2 \in \mathscr{S}^2 \subseteq \mathscr{R}^{27}$ consisted of the target pelvis position a^3_{pelvis} (the walking destination), the proprioceptive information of the robot including pelvis orientation, linear and angular velocity of the pelvis, the force of both feet, joint positions of the legs and the gait phase. The gait phase indicates the phase of the periodic gait at any point in time, which is implemented as a two-dimensional vector on the unit-circle to describe the phase of periodic trotting. For more details regarding the gait phase state, please refer to ref. 61, where the gait phase is used to enable the imitation learning of periodic locomotion.

The reward comprised of an imitation term and a task term:

$$r = w_i r_{\text{imitation}} + w_t r_{\text{task}},$$

with weights w_i and w_t and reward terms $r_{\text{imitation}}$ and r_{task} for the imitation and task, respectively. The imitation term encourages human-like motions by rewarding motions that are close to a reference motion capture trajectory. The task reward term rewards motions that contribute towards achieving the task, that is, walking towards a goal while maintaining balance. We found that combining imitation learning with task-guided reward shaping led to improved sample-efficiency[61] with policy convergence after 1.6×10^6 steps, equating to 18 hours of real-time.

To encourage a state x to be close to a desired target value $\hat{x}$, the corresponding reward component was designed as the radial basis function (RBF) kernel $K(\hat{x}, x, \alpha)$:

$$K(\hat{x}, x, \alpha) = e^{-\alpha(\hat{x}-x)^2},$$

with hyperparameter α controls the width of the kernel.

The aim of $r_{\text{imitation}}$ was to imitate the joint position, feet pose and contact pattern of a reference motion capture trajectory as close as possible. This is achieved by the reward function $r_{\text{imitation}}$:

$$r_{\text{imitation}} = w_{\text{joint\_position}} r_{\text{joint\_position}} + w_{\text{pose}} r_{\text{pose}} + w_{\text{contact}} r_{\text{contact}}.$$

The reward components $r_{\text{joint\_position}}$ and r_{pose} use the RBF kernel to encourage the policy learning motions that are close to the reference joint positions and feet poses, respectively. The contact reward r_{contact} is a binary reward that is equal to one if the foot in the reference motion was in contact with the ground and zero otherwise. The weights used for the reward components can be found in Supplementary Table 3. The target references for joint position, feet pose and feet contact come from the motion capture study in ref. 62.

The reward term r_{task} rewarded upright posture and short distances to the goal position, and regularized the joint velocity and torque:

$$r_{\text{task}} = w_{\text{pose}} r_{\text{pose}} + w_{\text{goal}} r_{\text{goal}} + w_{\text{vel}} r_{\text{vel}} + w_{\text{torque}} r_{\text{torque}},$$

with the values of the weights w_{pose}, w_{goal}, w_{vel} and w_{torque} as in Supplementary Table 3, and reward components r_{pose}, r_{goal}, $r_{\text{joint_vel}}$and r_{torque} that respectively reward the torso pose to be upright, the distance vector between pelvis and goal to be as small as possible and the joint velocity and joint torque to be as small as possible. The RBF kernel is used for all reward components in r_{task}.

Low-level joint control

The low-level joint control tracked the target joint positions $q^d = [q^d_{\text{arms}}, q^d_{\text{legs}}]$ provided by the mid-level stability controller (flow diagram in Supplementary Fig. 3). It receives joint positions $q \in \mathscr{R}^{20}$, joint velocities $\dot{q} \in \mathscr{R}^{20}$ and target joint position targets $q^d = a^2 \in \mathscr{R}^{20}$ as input and outputs motor current $a^1 = I \in \mathscr{A}^1 \subseteq \mathscr{R}^{20}$.

It was implemented as a joint impedance controller that regulated around the set point to achieve accurate tracking of the desired joint motions q^d.

The joint impedance control calculated the desired joint torque τ^{d} using position q and its derivative $\dot{q}$, with the stiffness K_{P_i} and damping K_{D_i} gains:

$$\tau^{\text{d}} = K_{\text{P}_1}\left(q^d - q\right) - K_{\text{D}_1}\dot{q}.$$

At the actuator level, the motor driver implemented an internal current control to track the desired joint torque τ^{d} using a proportional-derivative law, where the desired motor current I was computed as:

$$I = K_{\text{P}_2}\left(\tau^{\text{d}} - \tau\right) - K_{\text{D}_2}\dot{\tau}.$$

Data availability

The data analysed in this work were generated using the code provided in our open-source repository, where source data is also provided. Further information can be found in our repository (Code availability) and in the repository https://doi.org/10.5281/zenodo.8374262.

Code availability

The code used in this work is available on https://github.com/Yunaik/hgm4robots.git.

References

1. Li, N., Chen, T.-W., Guo, Z. V., Gerfen, C. R. & Svoboda, K. A motor cortex circuit for motor planning and movement. *Nature* **519**, 51–56 (2015).
2. Honey, C. J. et al. Slow cortical dynamics and the accumulation of information over long timescales. *Neuron* **76**, 423–434 (2012).
3. Murray, J. D. et al. A hierarchy of intrinsic timescales across primate cortex. *Nat. Neurosci.* **17**, 1661–1663 (2014).
4. Merel, J. et al. Hierarchical visuomotor control of humanoids. Preprint at https://doi.org/10.48550/arXiv.1811.09656 (2018).
5. Merel, J., Botvinick, M. & Wayne, G. Hierarchical motor control in mammals and machines. *Nat. Commun.* **10**, 5489 (2019).
6. Kheddar, A. et al. Humanoid robots in aircraft manufacturing: the airbus use cases. *IEEE Robot. Autom. Mag.* **26**, 30–45 (2019).
7. Schmaus, P. et al. *IEEE Aerospace Conference* (IEEE, 2019).
8. Oliver, G., Lanillos, P. & Cheng, G. An empirical study of active inference on a humanoid robot. *IEEE Trans. Cogn. Develop. Syst.*, **14**, 462–471 (2021).
9. Johnson, M. et al. Team IHMC's lessons learned from the DARPA robotics challenge trials. *J. Field Rob.* **32**, 192–208 (2015).
10. Kumagai, I. et al. Toward industrialization of humanoid robots: autonomous plasterboard installation to improve safety and efficiency. *IEEE Robot. Autom. Mag.* **26**, 20–29 (2019).
11. Winkler, A. W., Bellicoso, C. D., Hutter, M. & Buchli, J. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robot. Autom. Let.* **3**, 1560–1567 (2018).
12. Toyer, S., Thiébaux, S., Trevizan, F. & Xie, L. Asnets: deep learning for generalised planning. *J. Artif. Intell. Res.* **68**, 1–68 (2020).
13. Hutsebaut-Buysse, M., Mets, K. & Latré, S. Hierarchical reinforcement learning: a survey and open research challenges. *Mach. Learn. Knowl. Extr.* **4**, 172–221 (2022).
14. Jain, D., Iscen, A. & Caluwaerts, K. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019).
15. Li, C., Xia, F., Martin-Martin, R. & Savarese, S. Hrl4in: hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Proc. Conference on Robot Learning* (eds. Kaelbling, L. P., Kragic, D. & Sugiura, K.) 603–616 (PMLR, 2020).
16. Findeisen, W. et al. *Control and Coordination in Hierarchical Systems* (Wiley, 1980).
17. Sutton, R. S., Precup, D. & Singh, S. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **112**, 181–211 (1999).
18. Uithol, S., van Rooij, I., Bekkering, H. & Haselager, P. Hierarchies in action and motor control. *J. Cogn. Neurosci.* **24**, 1077–1086 (2012).
19. Loeb, G. E., Brown, I. E. & Cheng, E. J. A hierarchical foundation for models of sensorimotor control. *Exp. Brain Res.* **126**, 1–18 (1999).
20. Tani, J. & Nolfi, S. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Netw.* **12**, 1131–1141 (1999).

21. Botvinick, M. & Toussaint, M. Planning as inference. *Trends Cogn. Sci.* **16**, 485–488 (2012).
22. Wolpert, D. M., Ghahramani, Z. & Jordan, M. Forward dynamic models in human motor control: psychophysical evidence. *Adv. Neural Inf. Process. Syst.* **7**, 43–50 (1994).
23. Attias, H. Planning by probabilistic inference. In *Proc. Ninth International Workshop on Artificial Intelligence and Statistics* (eds Bishop, C. M. and Frey, B. J.) 9–16 (PMLR, 2003).
24. Baker, C. L., Saxe, R. & Tenenbaum, J. B. Action understanding as inverse planning. *Cognition* **113**, 329–349 (2009).
25. Maisto, D., Donnarumma, F. & Pezzulo, G. Divide et impera: subgoaling reduces the complexity of probabilistic inference and problem solving. *J. R. Soc. Interface* **12**, 20141335 (2015).
26. Kaplan, R. & Friston, K. J. Planning and navigation as active inference. *Biol. Cybern.* **112**, 323–343 (2018).
27. Tani, J. Learning to generate articulated behavior through the bottom-up and the top-down interaction processes. *Neural Netw.* **16**, 11–23 (2003).
28. Matsumoto, T. & Tani, J. Goal-directed planning for habituated agents by active inference using a variational recurrent neural network. *Entropy* **22**, 564 (2020).
29. Haruno, M., Wolpert, D. M. & Kawato, M. Hierarchical MOSAIC for movement generation. *Int. Congr. Ser.* **1250**, 575–590 (2003).
30. Morimoto, J. & Doya, K. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Rob. Autom. Syst.* **36**, 37–51 (2001).
31. Baltieri, M. & Buckley, C. L. Generative models as parsimonious descriptions of sensorimotor loops. *Behav. Brain Sci.* **42**, e218 (2019).
32. Friston, K. J., Parr, T. & de Vries, B. The graphical brain: belief propagation and active inference. *Net. Neurosci.* **1**, 381–414 (2017).
33. Pezzulo, G., Rigoli, F. & Friston, K. J. Hierarchical active inference: a theory of motivated control. *Trends Cogn. Sci.* **22**, 294–306 (2018).
34. Feldman, A. G. & Levin, M. F. in *Progress in Motor Control* (ed. Sternad, D.) 699–726 (Springer, 2009).
35. Botvinick, M. M., Niv, Y. & Barto, A. G. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition* **113**, 262–280 (2009).
36. Parr, T., Sajid, N. & Friston, K. J. Modules or mean-fields? *Entropy* **22**, 552 (2020).
37. Lanillos, P. et al. Active inference in robotics and artificial agents: survey and challenges. Preprint at https://doi.org/10.48550/arXiv.2112.01871 (2021).
38. Parr, T., Limanowski, J., Rawji, V. & Friston, K. The computational neurology of movement under active inference. *Brain* **144**, 1799–1818 (2021).
39. Aitchison, L. & Lengyel, M. With or without you: predictive coding and Bayesian inference in the brain. *Curr. Opin. Neurobiol.* **46**, 219–227 (2017).
40. Feldman, A. G. New insights into action–perception coupling. *Exp. Brain Res.* **194**, 39–58 (2009).
41. Adams, R. A., Shipp, S. & Friston, K. J. Predictions not commands: active inference in the motor system. *Brain Struct. Funct.* **218**, 611–643 (2013).
42. Shipp, S., Adams, R. A. & Friston, K. J. Reflections on agranular architecture: predictive coding in the motor cortex. *Trends Neurosci.* **36**, 706–716 (2013).
43. Miall, R. C., Weir, D. J., Wolpert, D. M. & Stein, J. F. Is the cerebellum a smith predictor? *J. Mot. Behav.* **25**, 203–216 (1993).
44. Koziol, L. F. et al. Consensus paper: the cerebellum's role in movement and cognition. *Cerebellum* **13**, 151–177 (2014).
45. Ramnani, N. Automatic and controlled processing in the corticocerebellar system. *Prog. Brain Res.* **210**, 255–285 (2014).
46. Bizzi, E., Mussa-Ivaldi, F. A. & Giszter, S. Computations underlying the execution of movement: a biological perspective. *Science* **253**, 287–291 (1991).
47. Atkeson, C. G. et al. *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (IEEE, 2015).
48. Yuan, K. & Li, Z. Multi-expert synthesis for versatile locomotion and manipulation skills. *Front. Robot. AI* **9**, 970890 (2022).
49. Yang, C., Yuan, K., Zhu, Q., Yu, W. & Li, Z. Multi-expert learning of adaptive legged locomotion. *Sci. Robot.* **5**, eabb2174 (2020).
50. Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P. & Pezzulo, G. Active inference: a process theory. *Neural Comput.* **29**, 1–49 (2017).
51. Sajid, N., Ball, P. J., Parr, T. & Friston, K. J. Active inference: demystified and compared. *Neural Comput.* **33**, 674–712 (2021).
52. Da Costa, L. et al. Active inference on discrete state-spaces: A synthesis. *J. Math. Psychol.* **99**, 102447 (2020).
53. Sajid, N. et al. Simulating lesion-dependent functional recovery mechanisms. *Sci. Rep.* **11**, 7475 (2021).
54. Lang, C. J. G., Kneidl, O., Hielscher-Fastabend, M. & Heckmann, J. G. Voice recognition in aphasic and non-aphasic stroke patients. *J. Neurol.* **256**, 1303–1306 (2009).
55. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. *Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Association for Computing Machinery, 2019).
56. Yuan, K., Chatzinikolaidis, I. & Li, Z. Bayesian optimization for whole-body control of high-degree-of-freedom robots through reduction of dimensionality. *IEEE Robot. Autom. Lett.* **4**, 2268–2275 (2019).
57. Hasselt, H. Double Q-learning. In *Advances in Neural Information Processing Systems* (eds Lafferty, J. et al.) 2613–2621 (Curran Associates Inc., 2010).
58. Siciliano, B., Khatib, O. & Kröger, T. *Springer Handbook of Robotics*, Vol. 200 (Springer, 2008).
59. Yuan, K., McGreavy, C., Yang, C., Wolfslag, W. & Li, Z. Decoding motor skills of artificial intelligence and human policies: a study on humanoid and human balance control. *IEEE Robot. Autom. Mag.* **27**, 87–101 (2020).
60. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) 1861–1870 (PMLR, 2018).
61. Yang, C., Yuan, K., Heng, S., Komura, T. & Li, Z. Learning natural locomotion behaviors for humanoid robots using human bias. *IEEE Rob. Autom. Lett.* **5**, 2610–2617 (2020).
62. McGreavy, C. et al. *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020).

Acknowledgements

We would like to thank the Theoretical Neurobiology Group at University College London and the Advanced Intelligent Robotics Lab for their insightful feedback. K.Y. was supported by the Engineering and Physical Sciences Research Council Center for Doctoral Training in Robotics and Autonomous Systems (EP/L016834/1). N.S. is funded by the Medical Research Council (MR/S502522/1) and a 2021–2022 Microsoft PhD Research Fellowship. K.F. is supported by funding for the Wellcome Centre for Human Neuroimaging (Ref: 205103/Z/16/Z), a Canada-UK Artificial Intelligence Initiative.

Author contributions

K.Y. and Z.L. conceptualized the robot control architecture. K.Y., N.S. and K.F. designed and formulated the hierarchical generative model. K.Y. implemented the model and performed the robotic experiments. K.Y. and N.S. wrote the manuscript. All authors contributed to and edited the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-023-00752-z.

Correspondence and requests for materials should be addressed to Zhibin Li.

Peer review information *Nature Machine Intelligence* thanks Pablo Lanillos, Wouter Kouw and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editor: Jacob Huth, in collaboration with the *Nature Machine Intelligence* team.