MOTION PLANNING AND CONTROL

Demonstrate once, execute on many: Kinematic intelligence for cross-robot skill transfer

Sthithpragya Gupta*†, Durgesh Haribhau Salunkhe†, Aude Billard

Teaching robots new skills should be as natural as showing rather than programming. Learning from demonstration (LfD) moves toward this goal by allowing users to guide a robot or sketch a desired motion, enabling learning without writing a line of code. However, most LfD methods remain tied to the robot they were trained on. Changes in morphology, different link lengths, joint orientations, or limits often break the learned behavior, making retraining unavoidable. Here, we introduce a framework that endows robots with kinematic intelligence: an internal understanding of their own joint limits, singularities, and connectivity. Instead of correcting for these constraints after learning, we embedded them directly into the control policy from the outset. The approach takes one or multiple demonstrations, extracts a globally stable dynamical system, and produces behaviors that remain valid across robots with different kinematic structures. Our method is grounded in a comprehensive analytical classification of noncuspidal three-revolute (3R) robots, which form the building blocks of many commercial robots. This classification enables a joint space policy that preserves user intent and adapts to robot-specific constraints. We validated the framework on diverse simulated and real robots, both redundant and nonredundant, with varied link geometries and joint configurations. The demonstrated skill executes safely and consistently across robots without retuning, thereby achieving cross-robot skill transfer.

INTRODUCTION

As robots become more common in our daily lives, from homes and hospitals to warehouses and factories, the ability to teach them new skills quickly and safely is becoming increasingly important. Instead of relying solely on expert programming, a more natural and intuitive alternative is to let users show robots what to do. Learning from demonstration (LfD) provides this convenience (*1*–*3*). It allows users, expert or not, to teach a robot by simply demonstrating the desired behavior. Demonstrations can take many forms: physically guiding the robot's joints (kinesthetic teaching) (*4*), using a remote control (teleoperation) (*5*), or even drawing a desired path or motion in the workspace (as shown in Fig. 1A). This removes the need for coding and lowers the barrier for human-robot interaction. Although LfD simplifies how we teach robots, there is a critical limitation: Most of today's LfD systems are tied to the specific robot they were trained on. If a user upgrades to a new robot, perhaps one with longer arms or different motion capabilities, the same taught skill may no longer work. Ideally, transferring skills between robots should be as seamless as syncing your preferences and apps when switching to a new phone or laptop. Unfortunately, that is not the case today. The problem becomes even more difficult because new robots offer greater motion articulation, more degrees of freedom, and more complex joint configurations, each of which changes how the robot moves and what motions are feasible.

To address this challenge, we argue that robots need more than the ability to mimic demonstrations. They must have what we call kinematic intelligence: an internal awareness of their kinematic constraints, joint limits, and feasible motion paths. Rather than relying on extensive demonstrations that implicitly avoid failure, our approach explicitly incorporates this structural awareness into the control policy. Kinematic intelligence enables robots to generalize behavior across different body structures. This is important because the human body rarely matches the mechanical structure of robots, and even among robots, link lengths and joint orientations vary widely, leading to different unstable configurations (singularities). However, the task itself (such as wiping a surface, picking up an object, or following a curved path) remains the same. To ensure that a behavior learned on one system can be reused by others, we would typically need to reprogram or retrain the task with full knowledge of each robot's constraints. Transfer learning offers a potential solution by reusing previously learned behaviors. However, it also faces challenges related to explainability, abstraction of transfer, and transfer metrics (*6*). Kinematic intelligence addresses this gap, allowing behaviors learned on one robot, or even from human-guided input, to execute safely and effectively on another robot without expert intervention.

In the past, mechanical intelligence was introduced to build part of a robot's capability directly into its physical design, material properties, and passive mechanics (*7*). In this sense, it is conceptually related to kinematic intelligence, given that both leverage the robot's inherent structure to support task execution. The key difference is that mechanical intelligence relies on physical embodiment, whereas kinematic intelligence relies on analytical properties, making it well suited to transferring demonstrated behaviors across different robot designs.

Challenges set forth by kinematic constraints in transfer learning

Previous work in LfD has made notable progress in learning smooth and repeatable behaviors from demonstrations, particularly in simple or well-controlled environments. The methods focused on directly imitating motion trajectories in the workspace, allowing robots to reproduce tasks like reaching, drawing, or object manipulation (*8*, *9*). These approaches worked well when the target robot had physical characteristics similar to those of the robot used during training. Some probabilistic models offer flexibility but still require separate handling of kinematic feasibility (*10*, *11*). Other techniques,

Learning Algorithms and Systems Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland.
*Corresponding author. Email: sthithpragya.gupta@epfl.ch
†These authors contributed equally to this work.

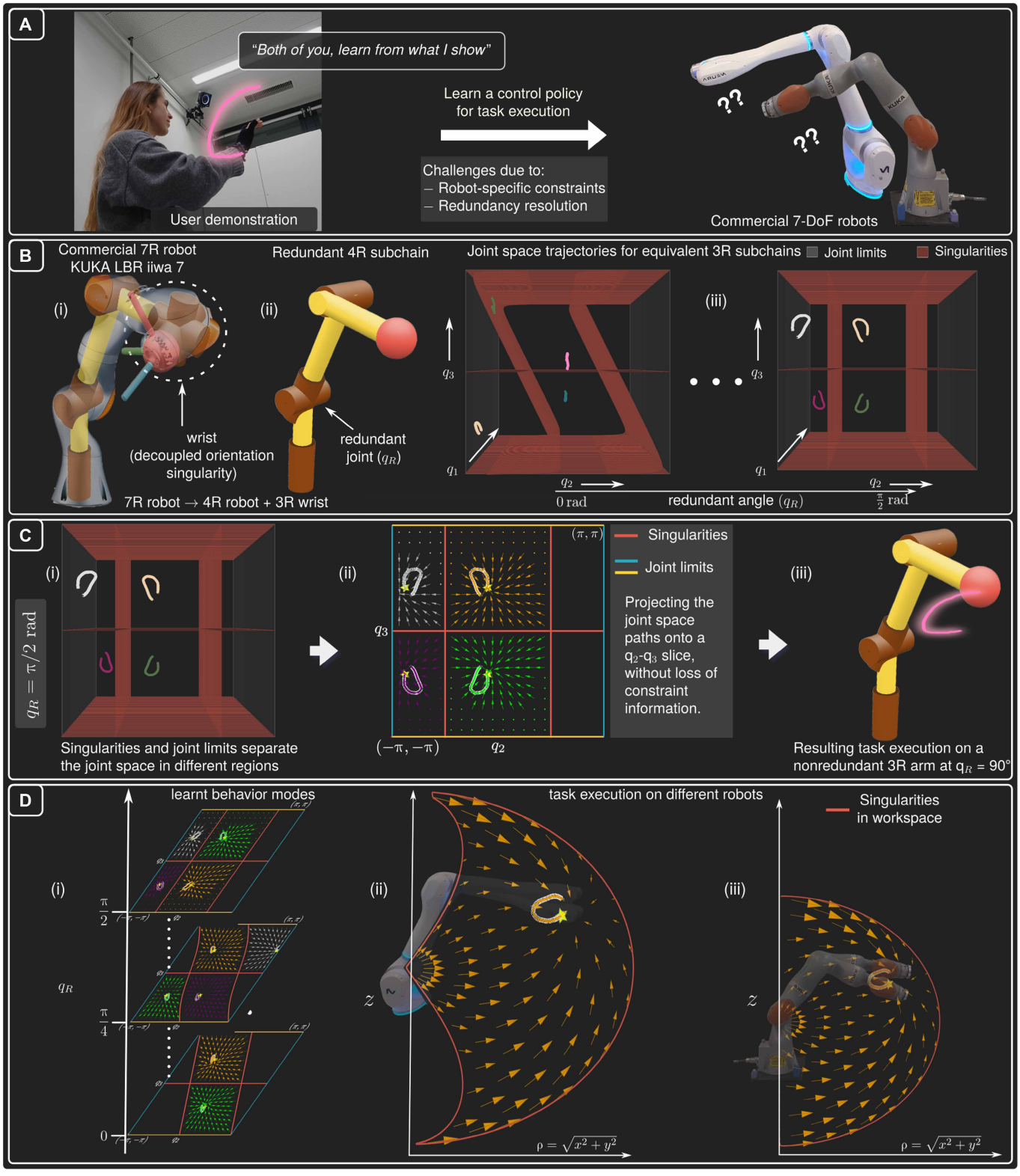


Fig. 1. Embedding kinematic intelligence in the human-demonstrated behavior for transferable skill policies. (**A**) A human demonstration defines the task, the policy for which must be learned and transferred across two commercially available 7-DoF robots. (**B**) Transfer reduction via kinematic structure: (i) wrist-partitioned 7-DoF robot, (ii) simplification to a redundant 4-DoF subchain, and (iii) redundancy parameterized by joint q_R, establishing equivalence across nonredundant 3R robots. (**C**) Kinematic constraints of an equivalent 3R robot (at $q_R = \pi/2$): (i and ii) Singularities and joint limits partition the joint space, and (ii and iii) a global control policy generates stable subspace dynamics for constraint-respecting execution. (**D**) Cross-robot transfer: (i) policies parameterized by q_R and (ii and iii) successful robot execution.

although more integrated, depend on detailed tuning or assumptions about the robot's workspace (*12*, *13*). However, when the robot changes, or when its physical structure leads to limitations near joint limits or singular configurations, the learned behavior may become infeasible or unsafe. Some approaches attempted to solve this by abstracting the task into a higher-level representation that is independent of the robot's embodiment (*14*–*17*). This abstract version is then adapted for each robot individually. Although this approach shows promise, it often requires multiple demonstrations, extensive parameter tuning, or manual safety corrections near the robot's constraints (*18*, *19*). Other research has explored learning mappings from workspace goals to joint configurations, commonly known as inverse models. These mappings work well in predictable areas of the robot's motion space, but they tend to fail near boundaries where movement becomes unpredictable or unstable. As a result, many methods include additional safety filters or online corrections to patch these issues in real time (*20*). Existing approaches typically emphasize task feasibility and constraints but often treat robot-specific constraints like singularities and joint limits as post hoc considerations rather than integral components of the learning process. Although useful, these strategies treat the robot's motion constraints as a separate concern rather than as part of the learning process itself. However, because of a lack of global understanding of the robot's constraints, using numerical tools in post hoc can make the robot follow paths that may contain unstable configuration(s) or hit a motion limit, risking task failure or even endangering human safety (*21*–*25*).

More recently, some data-driven methods aim to build shared task spaces or embeddings that allow robots to learn behaviors that can be reused across different platforms. These methods have demonstrated generalization capabilities but often rely on large datasets, robot-specific fine-tuning, or access to every target robot during training (*8*–*10*). This makes them less practical in real-world settings where only a few demonstrations are often available, given that robot teaching is a time-consuming endeavor for a nonexpert user. Overall, a gap remains in unifying robot-specific kinematic characteristics with generalizable learning frameworks that can function reliably with minimal demonstrations.

Many of these recent approaches are built around seven–degree-of-freedom (7-DoF) wrist-partitioned robots, which have become standard in collaborative and humanoid robotics. Their popularity stems from their structural resemblance to the human arm and their enhanced flexibility, offering redundancy that allows for more natural and adaptive motion in complex environments by offering more DoFs (seven) than strictly necessary (six). However, redundancy also poses a challenge for learning and reproducing behavior. Unlike nonredundant robots, redundant robots (see Fig. 1A) admit infinitely many joint space solutions for a given task, making it difficult to learn and generalize violation-free behaviors consistently across different configurations.

Kinematic analysis to facilitate transfer across robotic manipulators

To analyze the behavior of 7-DoF noncuspidal robots, it is useful to decompose them into smaller kinematic units, or subchains, that isolate the functional components of motion. Most modern 7-DoF robots use a spherical wrist composed of three intersecting rotational joints that control the end-effector orientation. Wrist-partitioned robots, widely adopted in research and certain industrial applications such as the Kuka LWR iiwa and Motoman SIA series, exhibit noncuspidal kinematic architectures (*26*). For such robots, inverse kinematic solutions (IKSs) are separated by singularity boundaries in joint space, leading to well-defined, predictable transitions between configurations. The spherical wrist not only decouples orientation singularities from the positioning subchain but also is noncuspidal, exhibiting two IKSs separated by $\sin(\theta_w) = 0$, where θ_w denotes the second wrist joint. Hence, the analytical framework proposed for noncuspidal 3R (three-revolute joint) positioning chains can be directly applied to the wrist subchain, providing certified, singularity-free orientation control. The remaining four joints form the positional subchain, redundant for three-dimensional (3D) positioning [see Fig. 1B (i to ii)]. To manage the redundancy-induced ambiguity, the 4R positional chain can be treated as a family of 3R noncuspidal subchains. By setting the angle of a redundant joint in the 4R subchain to a specific value, we effectively eliminate one DoF, reducing the 4R subchain to a nonredundant 3R subchain that is just sufficient for spatial positioning. Thus, the behavior of the 4R chain can be studied by analyzing the properties of its constituent 3R subchains, where the redundant joint serves as a parameter indexing different members of the family. In this way, the entire 4R positional chain can be interpreted as a family of 3R subchains, each corresponding to a different value of the redundant joint angle. This creates a parameterized representation, where the value of the redundant joint angle acts as a parameter that indexes a continuum of 3R subchains embedded in the original 4R subchain. Each of these 3R subchains has the same joint limit constraints but exhibits different kinematic properties, such as singularities, number of IKSs, and the feasible joint space regions.

This decomposition provides a systematic means to analyze the core positional capabilities of 7-DoF wrist-partitioned robots through a family of noncuspidal 3R subchains. These 3R units constitute the kinematic foundation of the robot and form the basis for our global, constraint-aware analysis. Because the wrist joint angles depend functionally on the configuration of the positional subchain and can be computed reactively without ambiguity, the positional feasibility results obtained from these subchains can be directly extended to full end-effector pose control.

Kinematic-aware learning of control policies

Following the classification proposed by Burdick (*27*), 3R robots can be grouped as generic, nongeneric, or degenerate according to the nature of their singularities. Generic robots exhibit well-behaved singular surfaces whose topology remains stable under small variations of link parameters. Nongeneric robots correspond to special geometries where these surfaces intersect or bifurcate, leading to multiple branches or self-intersections. Degenerate cases arise typically in robots with symmetric link lengths or right-angle joint arrangements. In such configurations, two joint axes may align or intersect, reducing the robot's effective mobility and leading to indeterminate motions if not explicitly modeled. Recognizing these degeneracies is therefore crucial: When known, they can be avoided for safety or exploited to improve task dexterity. Our framework systematically incorporates this awareness, embedding kinematic intelligence across all categories of noncuspidal robots—generic, nongeneric, and degenerate—thus ensuring constraint-aware and predictable behavior regardless of design variations.

We designed safe and generalizable control strategies that are not tailored to a specific robot. Safety refers here to control strategies that guarantee violation-free task execution by embedding certified

redirection mechanisms near kinematic constraints. To this end, instead of trying to remove or ignore a robot's physical constraints as done in many LfD approaches, we embed these constraints analytically into the policy architecture. By doing so, we ensure that the learned behavior is automatically adapted to each robot's structure in a safe and stable way [see Fig. 1, C (i to iii) and D (i to iii)]. Specifically, we used a topological and differential classification of singularities to understand how different parts of the robot's configuration space are separated by constraints (see Fig. 2). This classification allows us to identify singularity-free zones, distinguish between reachable and unreachable regions, and design joint space control policies that avoid constraint violations. With such kinematic-aware embedding of the learned policies, even a single demonstration is enough to create a behavior that is safe, robust, and executable on many different robots. We demonstrate this framework across a range of robots, including both redundant and nonredundant robots, each with different kinematic structures and singularity patterns. This is achieved without any retraining or robot-specific reprogramming.

RESULTS

We demonstrated transfer of skills in simulation with a series of nonredundant 3R robots. For redundant robots, we considered the KUKA IIWA LWR 7 and the Neura Maira M robot, both 7-DoF commercially available robots. The Maira robot had a peculiarity in that it had a degeneracy, but the degenerate configurations remained outside the feasible joint space. We observed across all robot configurations that the system successfully acquired a generalized policy of the user behavior, resulting in smooth, stable, and physically feasible robot execution (see Figs. 3 and 4). Even with varying robot structures and joint constraints, the embedded kinematic intelligence ensured that the task execution remained safe and accurate, demonstrating the framework's ability to generalize a skill across different robot bodies even from a single user demonstration. We further demonstrated redundancy parametrization and how it facilitated behavior transfer to redundant robots (see Fig. 5).

Kinematic intelligence

Kinematic intelligence refers to the embedding of analytical kinematic properties into the control framework using deterministic, fixed-compute algorithms. This enabled certified, constraint-compliant path planning and policy synthesis that generalized across robot embodiments without retraining. A central result of this work was a theoretically grounded classification of all noncuspidal 3R robots into six distinct categories, each defined by the global structure of their kinematic constraints. This classification formed the backbone of our kinematic intelligence framework, enabling the synthesis of robot-specific control strategies that were inherently safe and generalizable. The classification is based on the algebraic and topological analysis of the robot's Jacobian determinant. The determinant's factors and its root structure are determined by the robot's design parameters, such as link lengths and joint offsets, which define the location and shape of the locus of singular configurations, singularities, in the joint space. These singularities, together with joint limits, partition the joint space into distinct feasible regions, referred to in the remainder of this paper as aspects (*28*). Each category captured a specific class of kinematic behaviors, characterized by how singularities folded, looped, and intersected to segment the joint space. Crucially, the structure of each class also dictated a corresponding near-boundary control strategy. Rather than treating constraint handling as an afterthought, the categorization informed the embedding of corrective actions directly into the control policy. Once a robot's class was known, the corresponding strategy could be instantiated without further tuning or demonstration, yielding a scalable method for constraint-compliant behavior transfer across robot types. Figure 2 provides an overview of the topological structure of the singularities and joint limits in each class. This structure was not just descriptive: It was operational. It enabled one to build controllers that adapted dynamically to robot-specific constraints and preserved the fidelity of the demonstrated task.

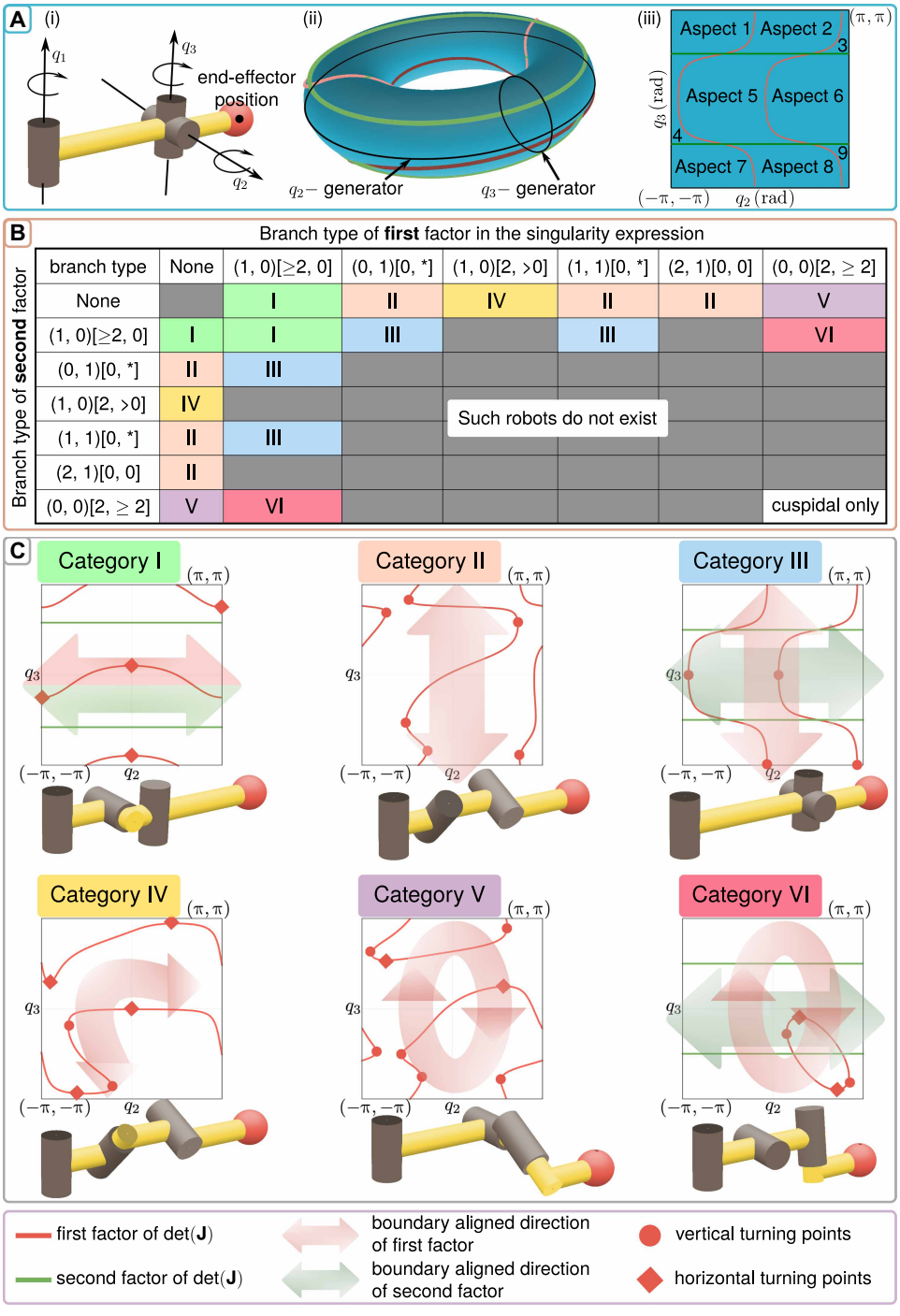
Robot categorization

We established a classification scheme for noncuspidal 3R robots through algebraic analysis of the Jacobian determinant, $\det \mathbf{J}(\mathbf{q})$, where $\mathbf{q}$ represents the joint configuration. Singularities depended only on the second and third joints (q_2 and q_3), allowing analysis in a reduced space defined by their angles, referred to as the q_2-q_3 slice of the joint space. $\det \mathbf{J}(\mathbf{q})$ can have up to three irreducible factors whose zero sets defined closed branches on the torus $\mathbb{T}^2$ generated by q_2-q_3. Each branch's winding numbers (n_1 and n_2) indicated how many times it wrapped around the torus generators.

Existing homotopy-based classifications (*29*, *30*) characterized singularities by their branches' winding numbers. Although exhaustive, this classification does not inherently support constrained motion planning. Consequently, robots in the same homotopy category may exhibit different differential properties affecting constraint-compliant motion planning. In addition, this classification is limited to generic robots only. To address these limitations, we introduced a symbolic notation $(n_1, n_2)[n_3, n_4]$ that retained the global winding behavior and captured the branch's differential structure. Here, n_3 and n_4 denote the number of horizontal and vertical turning points on a factor branch. By incorporating both topological and differential properties, this categorization ensured that robots in the same class shared identical constraint structures and allowed a unified path planning strategy near constraints.

Applying this framework to generic and nongeneric noncuspidal 3R robots, we identified six canonical categories describing the global structure of the joint space. The six categories shown in Fig. 2B were as follows: (i) nonloop, nonintersecting factors with only horizontal turning points—branches of type $(1, 0)[\geq 2, 0]$; (ii) nonloop, nonintersecting branches with no horizontal turning points—branches of type $(0, 1)[0, >0]$ or $(1, 1)[0, \geq 0]$; (iii) nonloop intersecting branches—two or more factors with intersecting nonloop branches; (iv) nonloop, nonintersecting branches with vertical turning points—branch type $(1, 0)[2, >0]$; (v) loop, nonintersecting branches—branch type $(0, 0)[2, \geq 2]$; and (vi) intersecting branches with a loop—intersecting factors where one branch formed a loop. Each class reflected a distinct constraint-induced partitioning of the joint space and corresponded to different boundary-following behaviors. This classification organized singularities across 3R robots and provided a basis for designing constraint-aware control policies that generalized across robot geometries. Examples of noncuspidal robots with varying Denavit-Hartenberg (D-H) parameters are presented in movie S1.

When a user demonstrated a task, the behavior was encoded as a task-level objective that each robot executed in a constraint-aware manner on the basis of its categorization. In contrast with traditional LfD methods that rely on post hoc corrections or multiple robot-specific demonstrations, the proposed framework generalized from



branch type	None	(1, 0)[≥2, 0]	(0, 1)[0, *]	(1, 0)[2, >0]	(1, 1)[0, *]	(2, 1)[0, 0]	(0, 0)[2, ≥ 2]
None		I	II	IV	II	II	V
(1, 0)[≥2, 0]	I	I	III		III		VI
(0, 1)[0, *]	II	III					
(1, 0)[2, >0]	IV						
(1, 1)[0, *]	II	III					
(2, 1)[0, 0]	II						
(0, 0)[2, ≥ 2]	V	VI					cuspidal only

Fig. 2. Topological-differential properties of singularities for categorizing noncuspidal 3R robots and defining boundary-aligned directions. (**A**) Representation of a 3R robot: (i) robot schematic, (ii) torus generated by (q_2-q_3), and (iii) aspects partitioning the q_2-q_3 slice. (**B**) Topological-differential analysis of the factors of the Jacobian determinant to assign any noncuspidal 3R robot to one of six categories (I to VI). (**C**) Representative robots from each category with their singularity curves and boundary-aligned directions (arrows) in the q_2-q_3 slice of joint space. Arrows indicate the control actions executed by the track-cycle control policy when the robot is near constraints. Category I: horizontal traversal; category II: vertical traversal; category III: switching between horizontal and vertical traversal; category IV: predominantly horizontal traversal with turning-point considerations; category V: clockwise or counterclockwise traversal along closed singular loops; category VI: loop traversal combined with horizontal drift along extended branches. The track-cycle policy, near constraints, guides the robot along the aspect boundary toward the goal maintaining motion within the same aspect.

a single demonstration by using an internally encoded model of joint space structure that was robot specific yet reusable across all robot categories. For example, if executing a demonstrated behavior on another robot would cause it to approach a singular configuration or violate a joint limit, the system automatically adjusted the trajectory to follow the boundary of the feasible region until it safely returned to the nominal path.

In essence, the structural classification functioned as a map that each robot used to safely interpret and execute a shared task. By integrating this into the learning process, we moved beyond treating kinematic constraints as external concerns and instead made them an intrinsic part of behavior generalization. Details of the robot categorization framework, its properties, joint space topology, constraints, and algorithms are provided in the "Robot categorization" section of Supplementary Materials and Methods.

Near-constraint strategy

A key outcome of this work was the establishment of a one-to-one correspondence between the robot's kinematic category and the control strategy that governs robot behavior along aspect boundaries. Each aspect is bounded on at least one side by a factor of det **J**(**q**) and on the remaining sides (if any) by the joint limits. Traversal along joint limits is straightforward because they form a linear variety in joint space, but motion along the factor boundary is nontrivial and varies markedly across categories. The core insight was that the traversal along the factor boundary in an aspect dictated the constraint-aware motion strategy and was distinctly determined by the robot's category. Each category is characterized by a key boundary-aligned direction(s), vector(s) intrinsic to the structure of the singularity factor, that governs motion near the constraint (see Fig. 2C). This direction is projected onto the tangent of the boundary to yield the feasible motion along it. This approach avoided motion stalling along the singularity branch, even in the presence of curvature reversals.

Robots from categories I, II, IV, and V exhibited a single boundary-aligned direction: horizontal (along q_2) in category I, vertical (along q_3) in category II, horizontal motion with intermittent vertical switching in category IV, and motion along a closed loop in category V

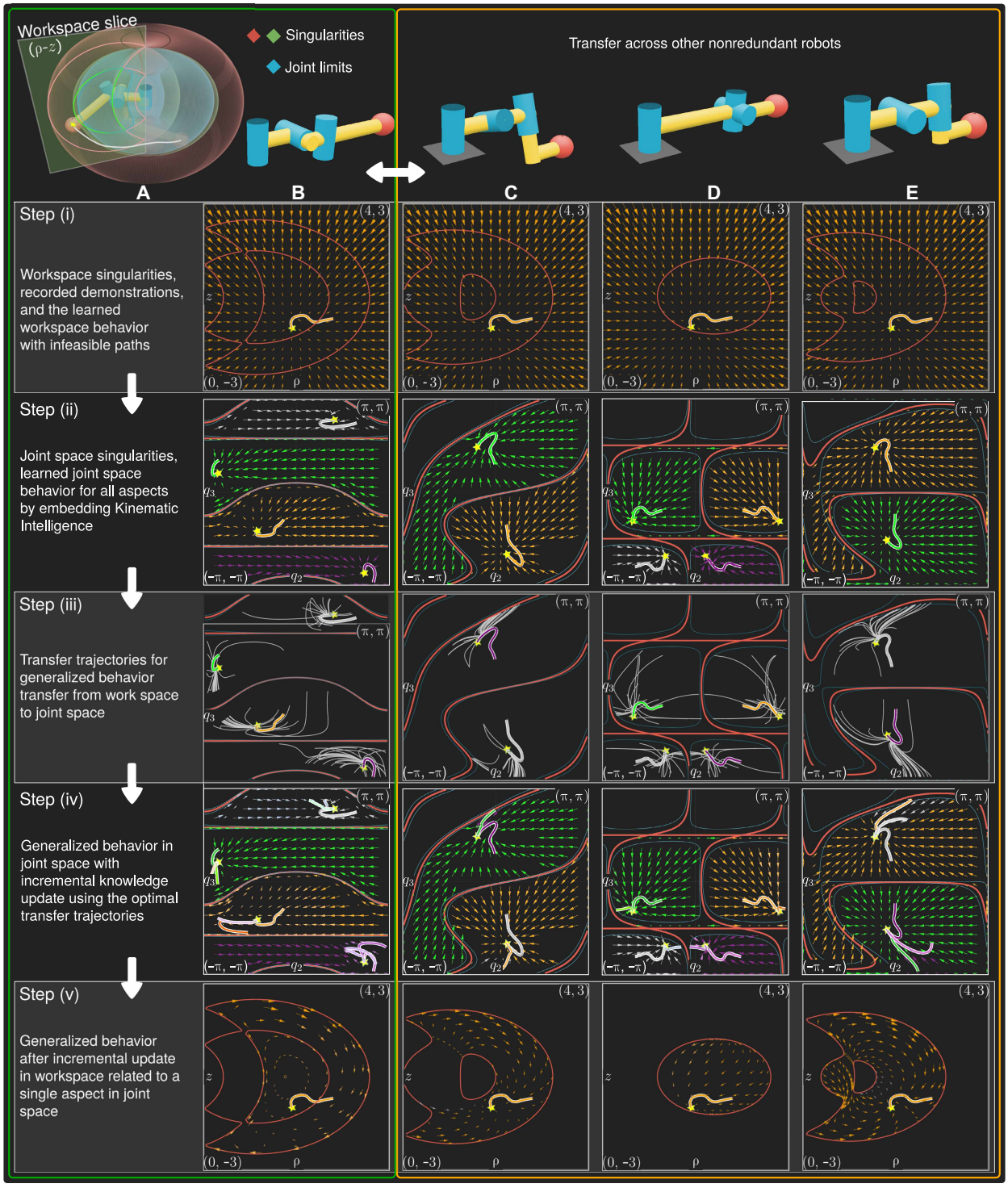


Fig. 3. Transfer of demonstrated behavior to multiple nonredundant robots. (**A**) Workspace of a 3R robot [schematic in (B)] with the learning framework steps (i to v). The demonstrated trajectory is visualized in a ρ-*z* workspace slice. Columns (**B** to **E**) correspond to representative robots from categories I to IV, and rows show the framework stages: (i) nominal workspace policy and singularities in the ρ-*z* slice, (ii) learned control policies for feasible joint space aspects, (iii) transfer trajectories for incremental policy updates, (iv) most novel transfer trajectory and updated behavior in each aspect, and (v) the resulting constraint-compliant workspace behavior.

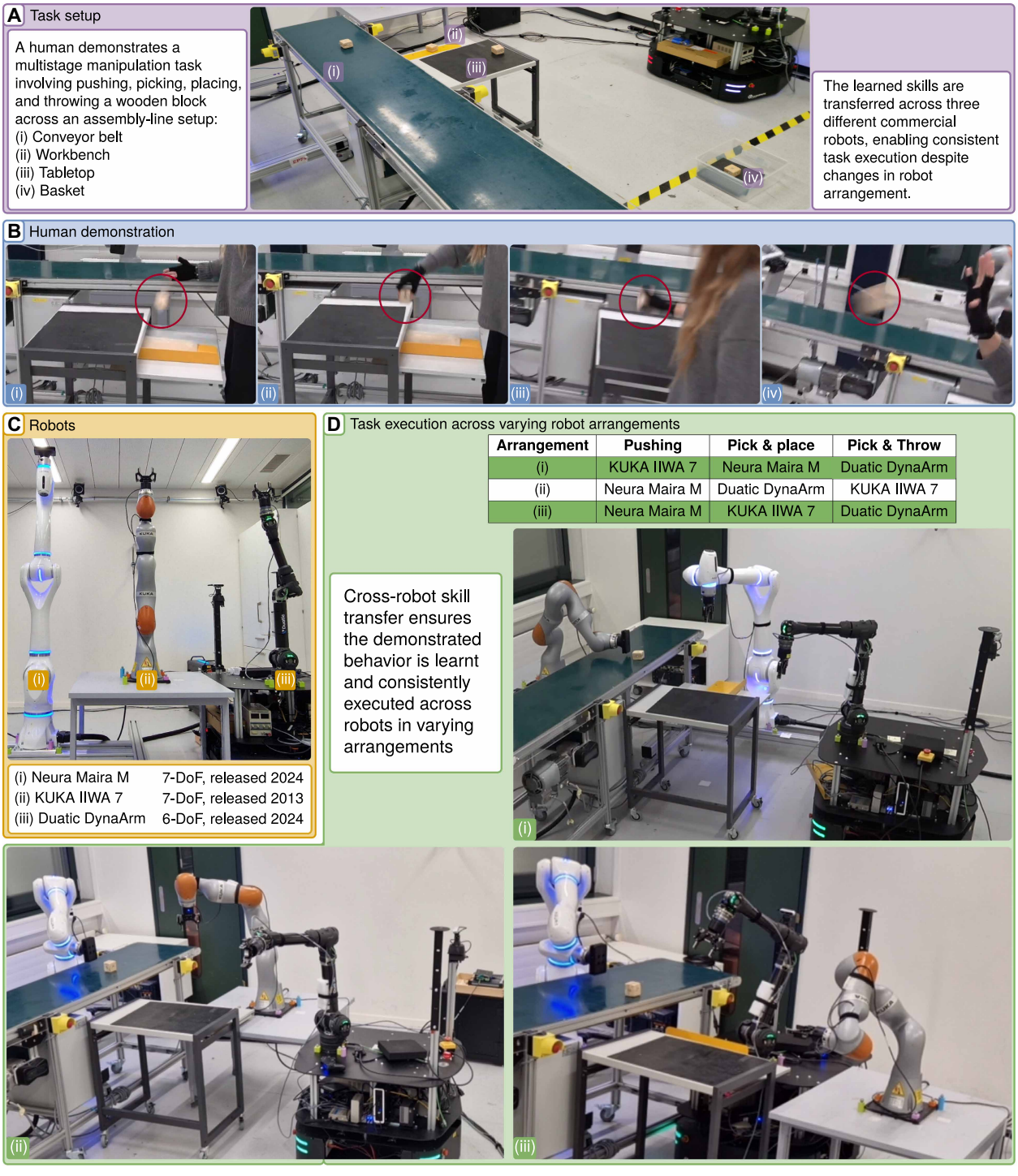


Fig. 4. Experimental validation in a mock multirobot automated assembly line. (**A**) Assembly-line setup consisting of a (i) conveyor belt, (ii) workbench, (iii) tabletop, and (iv) basket. (**B**) Human demonstration of the task: (i) pushing the wooden block off the conveyor belt on to the workbench, (ii) picking the block off of the workbench and placing it on the tabletop, (iii) lifting it into a throwing pose, and (iv) throwing the wooden block into the basket. (**C**) Three commercial robots to reproduce the demonstrated behavior: (i) Neura Maira M, (ii) KUKA IIWA 7, and (iii) Duatic DynaArm. (**D**) Varying robot arrangements (i to iii) in the assembly line for task execution.

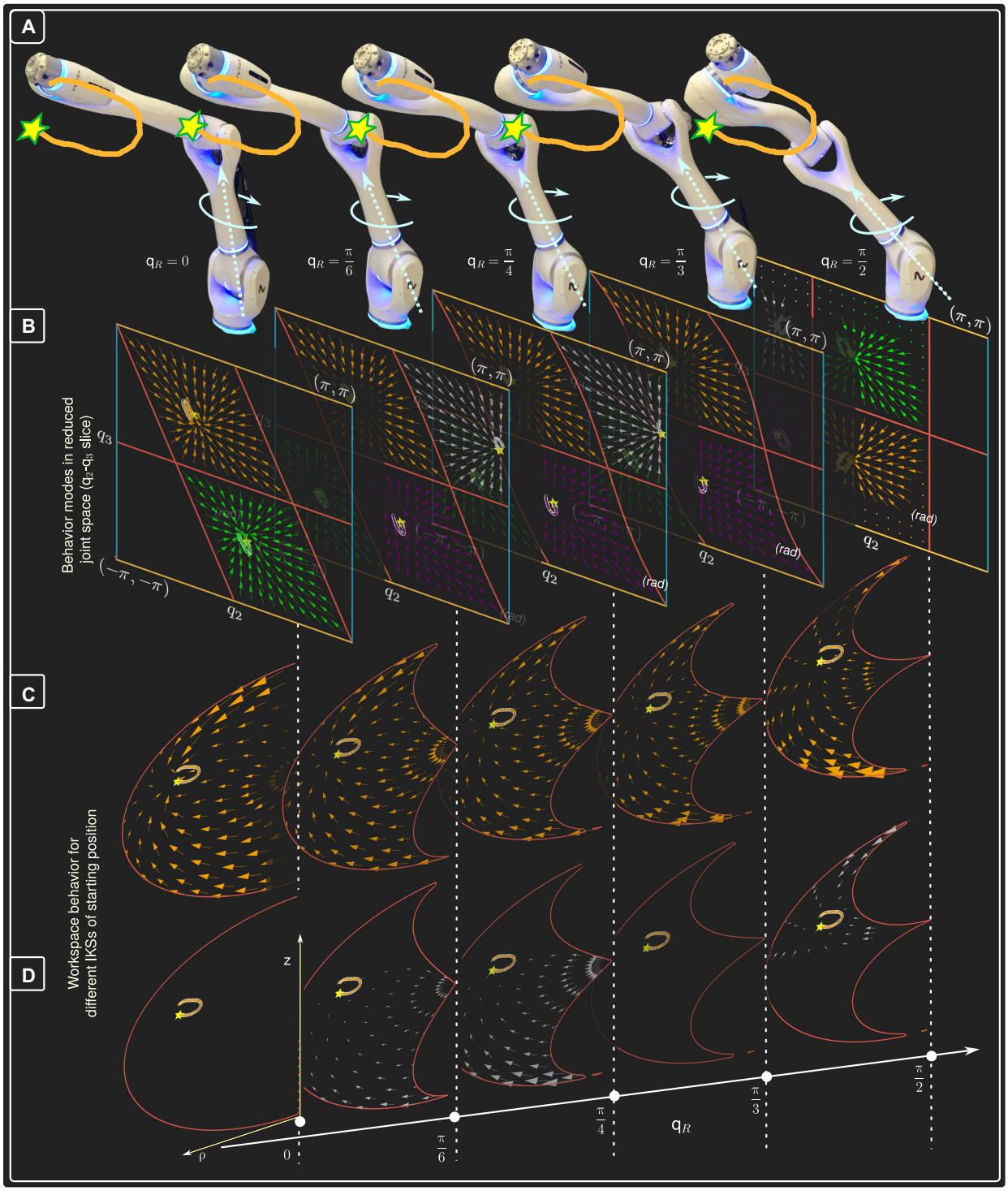


Fig. 5. Behavior transfer to redundant robots through redundancy parameterization. (**A**) Behavior of the redundant 4R subchain of a wrist-removed 7-DoF wrist-partitioned robot is characterized by discretizing the redundant angle q_R. Five representative values from 0 to $\pi/2$ radians are considered. (**B**) Control policies for the equivalent nonredundant 3R subchains at these q_R values in the reduced joint space. (**C**) Feasible workspace shown in a ρ-z slice for each q_R, with the robot configuration in a specific joint space aspect highlighted in orange. (**D**) Resulting workspace behavior when the robot starts from a different IKS, shown in silver.

(see Fig. 2C). In contrast, categories III and VI were defined by two boundary-aligned directions arising from two distinct factors in det $\mathbf{J}(\mathbf{q})$, corresponding to intersecting singularities. Each factor contributed its own traversal direction. This differed from category IV, where only one factor existed despite mixed motion, given that horizontal and vertical behaviors occurred along the same branch. Multiple factors also did not necessarily imply multiple directions: Category I robots could have two factors, but because both factors had the same branch type $(1, 0)[\geq 2, 0]$, the horizontal direction sufficed.

We refer to the structured control strategies incorporating these category-specific boundary-aligned directions for governing the robot behavior near constraints as track cycles. When the nominal dynamics, modeled from user demonstration(s), generated a trajectory $\mathbf{Q}$ that exited the current aspect, convergence to the goal required eventual reentry. The track cycle defined a stable control policy active between the first exit and last reentry, redirecting motion along the aspect boundaries. Overall, the track cycle captured the sequence of boundary transitions that guided the robot from any boundary configuration back to a safe configuration consistent with the original trajectory $\mathbf{Q}$. We adopted a hybrid control framework that overlaid the track-cycle policy onto the nominal dynamics. When the robot approached safety margins around singularities or joint limits, the corresponding corrective policies were activated promptly. Further details on factors, their properties, and track cycles are provided in the "Behavior near kinematic constraints" section in Supplementary Materials and Methods.

Transfer across robots

To evaluate cross-robot behavior transfer, we considered a scenario where a user demonstrated tasks such as drawing shapes or letters or hitting, rearranging, or throwing objects, which different robots attempted to reproduce. These robots varied in mechanical design: Some were nonredundant (with just enough degrees of freedom), and others were redundant (with more joints than necessary). The framework generated robot-specific joint-space control policies that preserved the demonstrated intent and ensured safety. The results validated the core hypothesis: Embedding structural constraints into the learning framework allowed a behavior demonstrated once to transfer safely across robots with different kinematics without expert tuning or additional data.

Transfer to nonredundant robots

To evaluate how a single demonstration could be extended across multiple nonredundant robots with different kinematic structures, we tested our framework on four 3R robots: one each from categories I through IV. For each robot, the goal was to generate a control policy that generalized user-demonstrated motion and respected robot-specific constraints like singularities and joint limits. As shown in Fig. 3, our method models joint space behavior aspect-wise, where each aspect is a region free of singularities, and incrementally builds a constraint-aware policy. The process is presented for the four aforementioned robots in Fig. 3 (B to E). The figure outlines each step of the framework: from modeling the workspace behavior [Fig. 3, step (i)], learning aspect-specific joint space behaviors [Fig. 3, step (ii)], identifying feasible transfer trajectories [Fig. 3, step (iii)], and workspace–to–joint space transfer and updating the policy on the basis of the most novel transfer trajectories [Fig. 3, step (iv)]. Last, Fig. 3, step (v) formulated the resulting workspace motion that was faithful to the original demonstration and stayed entirely in safe, constraint-compliant regions. Across all tested robot categories, the learned joint space policy remained free from constraint violations without requiring retraining, enabling robust and scalable behavior transfer to diverse nonredundant robots from only a single demonstration.

Transfer to redundant robots

To assess transfer to redundant robots, we considered 7-DoF wrist-partitioned robots. The user-demonstrated end-effector trajectory generated a control policy for position control of these redundant robots. Redundant robots can be parameterized by an arm angle q_R, representing a family of nonredundant 6R robots whose architecture depends on this angle (*31*, *32*). A prior work showed that det $\mathbf{J}(\mathbf{q})$ of redundant robots can be expressed as a function of q_R (*26*), implying that the redundant robot can be treated as a nonredundant robot at any arbitrary value of q_R. In practice, we discretized candidate q_R values and performed classification-fixed q_R that realized the task in a single feasible aspect without joint-limit or singularity violation. The q_R value remained constant during execution such that, from the viewpoint of the positional chain, the robot behaved like a nonredundant 3R robot and no discontinuities due to online q_R switching were introduced.

Figure 5 outlines redundancy parametrization into five behavioral modes. The redundant 4R positional subchain was identified and parameterized using $q_R = 0, \pi/6, \pi/4, \pi/3$, and $\pi/2$ rad (Fig. 5A). All five of these 3R robots had nonloop intersecting singularity branches with types $(1, 0)$ $[\infty, 0]$ and $(1, 1)[0, \geq 0]$ when $q_R = 0$ and $(1, 0)$ $[\infty, 0]$ and $(0, 1)[0, >0]$ otherwise. Figure 5B shows the resulting joint-space constraints. At each q_R, multiple IKSs existed for the starting configuration. Some of these configurations were feasible for task completion, as shown in Fig. 5B, and produced the workspace behavior shown in Fig. 5C. However, some of the other IKSs (silver) were not feasible. For $q_R = 0$ and $\pi/3$, feasible behaviors were not detected because the goal configuration lay on the joint-limit boundary. The feasible workspace bounded by singularities varied with q_R (Fig. 5, C and D). For different values of q_R, the procedure yielded a set of behavioral modes, each encoding the demonstrated behavior and enabling selective retrieval depending on the task constraints. This modular arrangement preserved the safety and robustness properties of each individual behavior mode, thereby providing a violation-free control policy for wrist-partitioned redundant robots.

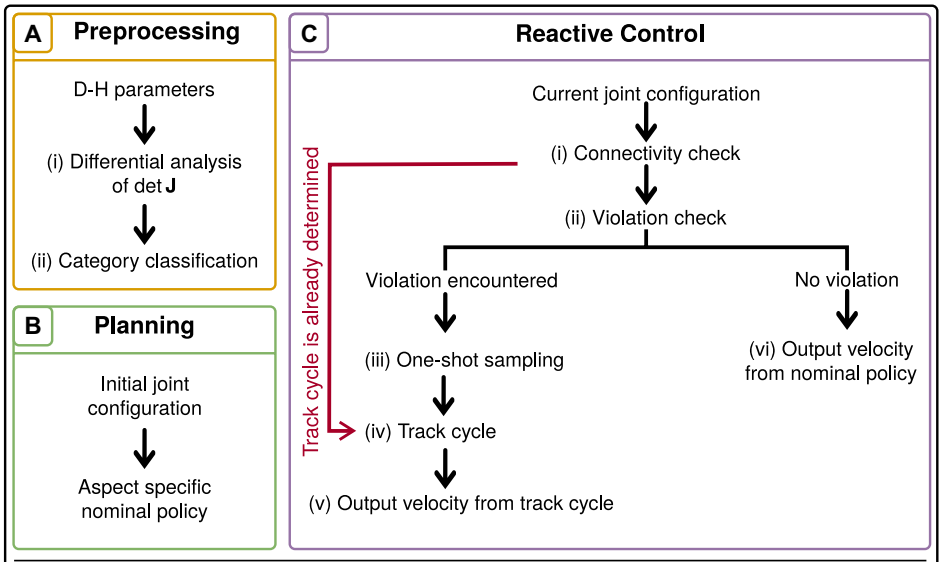
Certified execution

To model the robot control policy from joint trajectories, we represented each trajectory as a globally stable dynamical system in a one-shot manner (*33*). The resulting policy, composed of such systems, was resilient to temporal and spatial perturbations. Small deviations were corrected passively, and the space region triggered a topology-aware feasibility analysis to determine whether the robot's current aspect admitted a constraint-compliant trajectory to the demonstrated goal. Because not all aspects admitted a valid goal configuration, we developed certified factor-specific connectivity. Five factor types arose from their branch types: $(1, 0)[\geq 2, 0]$, $(0, 1)$ $[0, >0]$, $(1, 0)[2, >0]$, $(0, 0)[2, \geq 2]$, and $(1, 1)[0, \geq 0]$ (see the "Robot categorization" section in Supplementary Materials and Methods). Each irreducible factor of det $\mathbf{J}$ partitioned the joint space into multiple regions, and feasibility required the start and goal configurations to lie in the same partition for every factor. This incorporated

the global topological-differential structure of the robot's kinematic space into the control logic and enabled fast, deterministic connectivity analysis.

To leverage the certified kinematic properties in learning the demonstrated behavior and formulating a control policy, the framework undertook a multistep process summarized in Fig. 6, which depicts the implementation pipeline for executing a learned policy on a single robot. For each 3R robot at a fixed value of q_R, we first grouped the offline, one-time preprocessing steps used to determine the robot category as shown in Fig. 6A (see the "Robot categorization" section of Supplementary Materials and Methods). Figure 6B corresponds to the planning stage, where we learned the aspect-specific nominal joint behavior (see the "Modeling the user behavior" section of Supplementary Materials and Methods). The resulting policy governed the reactive control loop shown in Fig. 6C, which combined connectivity checks with the attractor, constraint-violation monitoring, nominal policy execution, and track-cycle computations to ensure faithful replication of the demonstrated behavior and respect kinematic constraints. The wall-clock timings and latency induced by each step are reported in Fig. 6D; these timings showed that preprocessing was negligible relative to demonstration time and that both planning and reactive control incurred only modest computational overhead, making the certified execution layer compatible with real-time control on the tested platforms.

Figure 7 illustrates certified execution and its aspect-based failure modes on a robot from category I with regions of two and four IKSs in the workspace. From any chosen IKS, the controller executed the demonstration only if a feasible aspect connected the start and goal configurations; otherwise, it halted safely. For nonredundant 6R robots, failures occurred when no admissible path existed or when perturbations drove the state into joint-limit or singularity margins, triggering track cycles or a safe stop. The same logic extended to wrist-partitioned 7R robots (Fig. 5), where tasks were declared infeasible if no redundant angle q_R yielded a feasible aspect. A detailed discussion of the failure modes is presented in the "Robustness" section of Supplementary Materials and Methods.



Task	Mean (μ)	Std. Dev. (σ)	95th Pctl.	Min	Max
A. Preprocessing					
(i) Differential analysis of det(**J**)	407.3 ms	466.3 ms	935.7 ms	12.0 ms	7755.0 ms
(ii) Category classification	253.3 μs	744.4 μs	1143.8 μs	5.9 μs	70637.4 μs
B. Planning					
Aspect specific behavior*	2.2 ms	0.94 ms	2.8 ms	1.7 ms	17.6 ms
C. Reactive Control					
(i) Connectivity checks	434.8 μs	7125.0 μs	562.4 μs	57.6 μs	679353.8 μs
(ii) Violation check	80.7 μs	38.1 μs	100.0 μs	19.8 μs	3525.0 μs
(iii) One-shot sampling*	5.6 ms	1.3 ms	8.2 ms	3.0 ms	11.6 ms
(iv) Track-cycle calculation	701.5 μs	15320.2 μs	966.7 μs	1.0 μs	1980828.8 μs
(v) Track-cycle velocity output	345.4 μs	640.3 μs	585.1 μs	1.0 μs	64676.5 μs
(vi) Nominal policy velocity*	2.2 ms	0.94 ms	2.8 ms	1.7 ms	17.6 ms
D. Learning Statistics					
(i) Model Training Duration*	250.3 s	123.2 s	457.7 s	130.0 s	1131.9 s
(ii) Trajectory Error (DTW)*	1.7	15.9	0.0	0.0	210.7

*: Data are from a smaller subset of 330 robots. All other data are from 21,787 robots.

Fig. 6. End-to-end policy execution on a single robot. (**A**) Preprocessing: an offline one-time step to determine the robot's kinematic structure and perform categorization using topological-differential properties. (**B**) Planning: Given an initial joint configuration, the framework identifies the corresponding aspect and the suitable joint space control policy. (**C**) Reactive control: an online loop combining connectivity checks, constraint-violation monitoring, nominal policy execution, and track-cycle computations to keep the robot in feasible regions. Reported wall-clock timings summarize the computational cost of each component, highlighting that the one-time offline preprocessing is very fast and that the online planning and reactive control steps are compatible with real-time execution.

Experimental evaluation of the "demonstrate once, execute on many" principle

We experimentally validated the "demonstrate once, execute on many" principle on three industrial robots. These included a compact 6-DoF Duatic DynaArm with tighter joint limits, a 7-DoF KUKA LWR IIWA 7 with a wrist-partitioned architecture and moderate limits, and a 7-DoF Neura Robotics Maira M with longer links and more relaxed limits, yielding markedly different workspaces, proximity to limits, and feasible aspect layouts for the same end-effector task.

In experiment 1, a human demonstrated a path resembling the letters of "SCIENCE" once per letter using a motion-capture glove and OptiTrack. The recorded trajectories were encoded as dynamical-system–based policies and replayed on the two 7-DoF robots, which reproduced the characters without joint limit or singularity violations across multiple aspects and redundant angles (see movies S2 and S3). In experiment 2, we constructed a mock multirobot assembly line with three skills—pushing, pick-and-place, and throwing. Each skill was demonstrated once, encoded once as a joint space policy, and executed on three robots without retraining. Pushing and throwing drove the joints toward workspace boundaries and induced large excursions, whereas

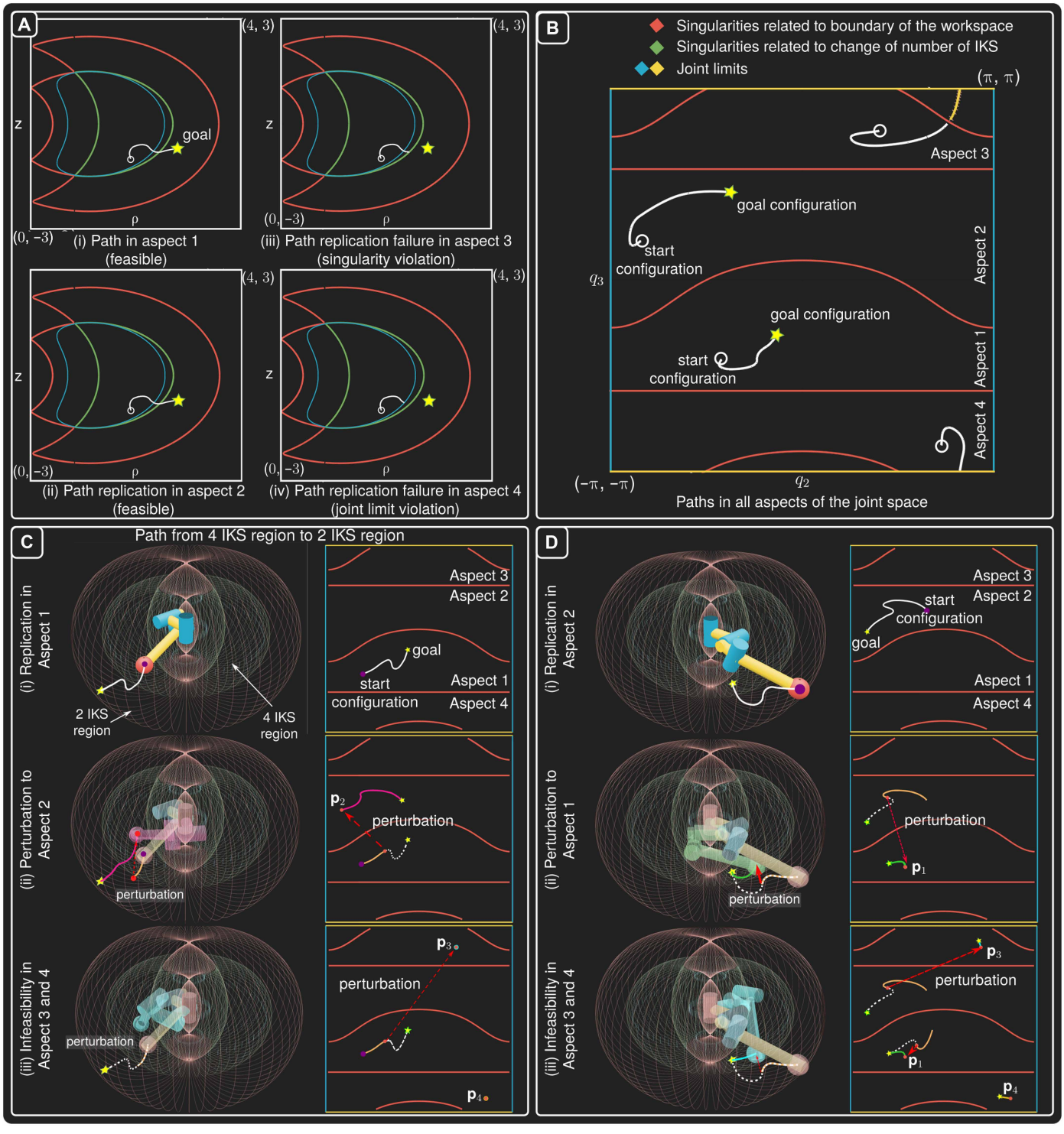


Fig. 7. Certified execution for nonredundant robots. The figure analyzes the workspace from Fig. 3A (i) on a category I 3R robot. The start position admits four IKSs, and the goal admits two. (**A**) Workspace trajectories for each aspect in a ρ-z slice of the workspace: (i and ii) Aspects 1 and 2 reach the goal without violations, (iii) aspect 3 encounters a singular configuration, and (iv) aspect 4 violates the third-joint limit. (**B**) Corresponding joint space trajectories for the four aspects, highlighting constraint violations in aspects 3 and 4. (**C**) Robot behavior after learning the control policy and embedding kinematic intelligence: (i and ii) Starting from a feasible aspect or being perturbed to a feasible configuration $\mathbf{p}_2$, the policy achieves violation-free convergence to the goal; (iii) if the robot is perturbed to or starts from an infeasible configuration ($\mathbf{p}_3$ and $\mathbf{p}_4$, respectively), the motion halts to avoid constraint violations. (**D**) Behavior when the policy is learned from the reversed demonstration, where the start has two IKSs and the goal has four: (i to iii) The learned policy reaches the goal from each configuration, including perturbations to configurations $\mathbf{p}_1$ and $\mathbf{p}_3$ and starting from $\mathbf{p}_4$.

pick-and-place required aspect queries combined with connectivity checks. The same policy was reused; only the kinematic embedding and certified execution layer changed, keeping trajectories within feasible aspects and redirecting them along analytic track cycles near constraints. Figure 4 shows the three robots performing the three skills; full executions appear in movie S4.

DISCUSSION

Teaching robots through demonstration is an effective way for humans to communicate complex skills. As robotic platforms diversify, sharing learned behaviors across systems without reprogramming becomes increasingly important. This is the goal of transfer LfD: to teach once and generalize broadly. We propose integrating behavior learning with robot-specific constraint enforcement by embedding analytic kinematic knowledge directly into the control policy. This enables proactive avoidance of infeasible configurations without post hoc checks. Our framework advances LfD by enabling safe and generalizable skill transfer across robot architectures through kinematic intelligence, separating user intent from robot morphology. Each demonstrated trajectory is modeled as a globally asymptotically stable dynamical system, ensuring convergence to its target configuration. This stability at the most fundamental level yields robustness to temporal and spatial perturbations, without requiring reactive correction. Because the control policy is incrementally built by adding more demonstrations, the stability of each dynamical system ensures that the entire policy remains stable at every stage of learning.

Although behavior is demonstrated in workspace, it is ultimately executed in joint space. Reliable replication therefore requires preserving user intent in joint space accounting for robot-specific constraints. Our framework achieves this through a topological-differential classification of singularities to categorize robot types, followed by connectivity analysis to assess feasibility and, last, a constraint-aware policy synthesis. Control policies are encoded in joint-space aspects, ensuring that trajectories remain within feasible boundaries. If a trajectory approaches a constraint, a near-boundary strategy redirects motion along a safe path called the track cycle and rejoins the original trajectory once clear. These category-specific strategies are embedded into the control logic, enabling certifiable behavior through bounded-complexity feasibility checks. All connectivity and constraint handling methods are derived analytically from the robot's structure, ensuring that motion is always grounded in the robot's capabilities. This analysis also extends to redundant robots: It executes successfully for all redundant angles in feasible aspects and correctly halts when a configuration becomes infeasible. The framework also supports incremental generalization of demonstrated behavior. At any instance, the already-learned behavior is used to generate feasible joint space trajectories, which are then ranked by a novelty metric based on how much new information they contribute, incrementally aligning the generalized behavior more closely with the user intent. Experiments validate the robustness of this approach on robots with different kinematics. Our framework synthesizes robot-specific joint space controllers in a transparent and explainable manner, establishing user trust by making robot behavior predictable and legible (*34*, *35*).

Although many recent collaborative and industrial robots are cuspidal (such as ABB GoFa, Fanuc CRX, and Kinova Link 6) or use nonspherical wrists (for instance, UR5), the wrist-partitioned architecture remains widely adopted (including Kuka LWR, Neura Maira, and DynaArm) (*26*). Studying wrist-partitioned noncuspidal cases therefore provides a foundation for certified, singularity-aware control policies because it provides a separation of IKSs allowing rigorous analysis. In contrast, cuspidal robots have multiple IKSs in an aspect, introducing ambiguity in control policies due to nonsingular changes of solutions (*23*, *36*). We therefore focused on noncuspidal architectures to establish safety and predictability, with the extension to cuspidal architectures as the next step. Recent results show that all 3R robots admit reduced aspects (*37*), which subdivide joint space even when multiple IKSs exist in an aspect. These reduced aspects, together with admissible path types in cuspidal robots (*22*), could support generalized connectivity checks and track-cycle strategies. In principle, collision handling could also be incorporated by treating collision manifolds as additional joint-space boundaries in the track-cycle framework, although developing and validating such collision-aware, cuspidal-ready extensions remains future work.

In summary, our framework provides a constraint-aware foundation for transferable robot learning. Kinematic intelligence in the learning pipeline supports scalable and safe deployment across diverse platforms, bringing us closer to seamless integration of robotics in everyday life.

MATERIALS AND METHODS

Overview

This research aimed to develop a framework for transferring demonstrated behavior across a class of 3R robots and redundant 4R robots, laying the foundation for behavior transfer across wrist-partitioned 6R and 7R robots. The framework first modeled the user behavior agnostic to the robot and then autonomously embedded the behavior model with kinematic intelligence, resulting in a control policy that oversaw safe and violation-free execution on a nonredundant robot. A modular arrangement of such nonredundant robot control policies, when composed together, enabled behavior execution on redundant robots. An overview of the challenges, a brief motivation to kinematic analysis, and behavior transfer are presented in movie S5.

Behavior modeling from demonstration

We began by modeling the user-demonstrated behavior in both the workspace $\mathcal{X}$ and joint space $\mathcal{Q}$ of the robot (see Fig. 8, A to C). Each user demonstration (trajectories in $\mathcal{X}$) and its inverse-kinematics mapping in $\mathcal{Q}$ were individually embedded as globally asymptotically stable dynamical systems: $\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x})$ for the workspace trajectory, where $\mathbf{x}$ is the position of the robot in the workspace, $\dot{\mathbf{x}}$ is the velocity of the robot in the workspace, and $\mathbf{g}(\mathbf{x})$ is the robot control policy in the workspace, and $\dot{\mathbf{q}} = \mathbf{g}(\mathbf{q})$ for the joint trajectory, where $\mathbf{q}$ is the joint configuration of the robot, $\dot{\mathbf{q}}$ is the joint velocity of the robot, and $\mathbf{g}(\mathbf{q})$ is the robot control policy in the joint space. The dynamical system formulation ensured that a trajectory from any starting point in the workspace or joint space converged to the demonstrated goal over time. Each dynamical system was constructed in a latent space $\mathcal{U}$, a transformed coordinate system obtained via a bijective mapping $\psi: \mathcal{U} \rightarrow \mathcal{Z}$, where $\mathcal{Z}$ was either $\mathcal{X}$ or $\mathcal{Q}$. In the latent space, the dynamics became (quasi)linear, simplifying stability guarantees. The learned dynamics in $\mathcal{U}$ were then pulled back to the original space via the mapping, ψ, producing nonlinear but stable dynamical systems in both spaces, $\mathcal{X}$ and $\mathcal{Q}$.

To generalize across multiple demonstrations, we grouped together all joint space trajectories that terminated in the same aspect $\mathcal{A}_i$ (see Fig. 8C), where i is the aspect index. Each resulting

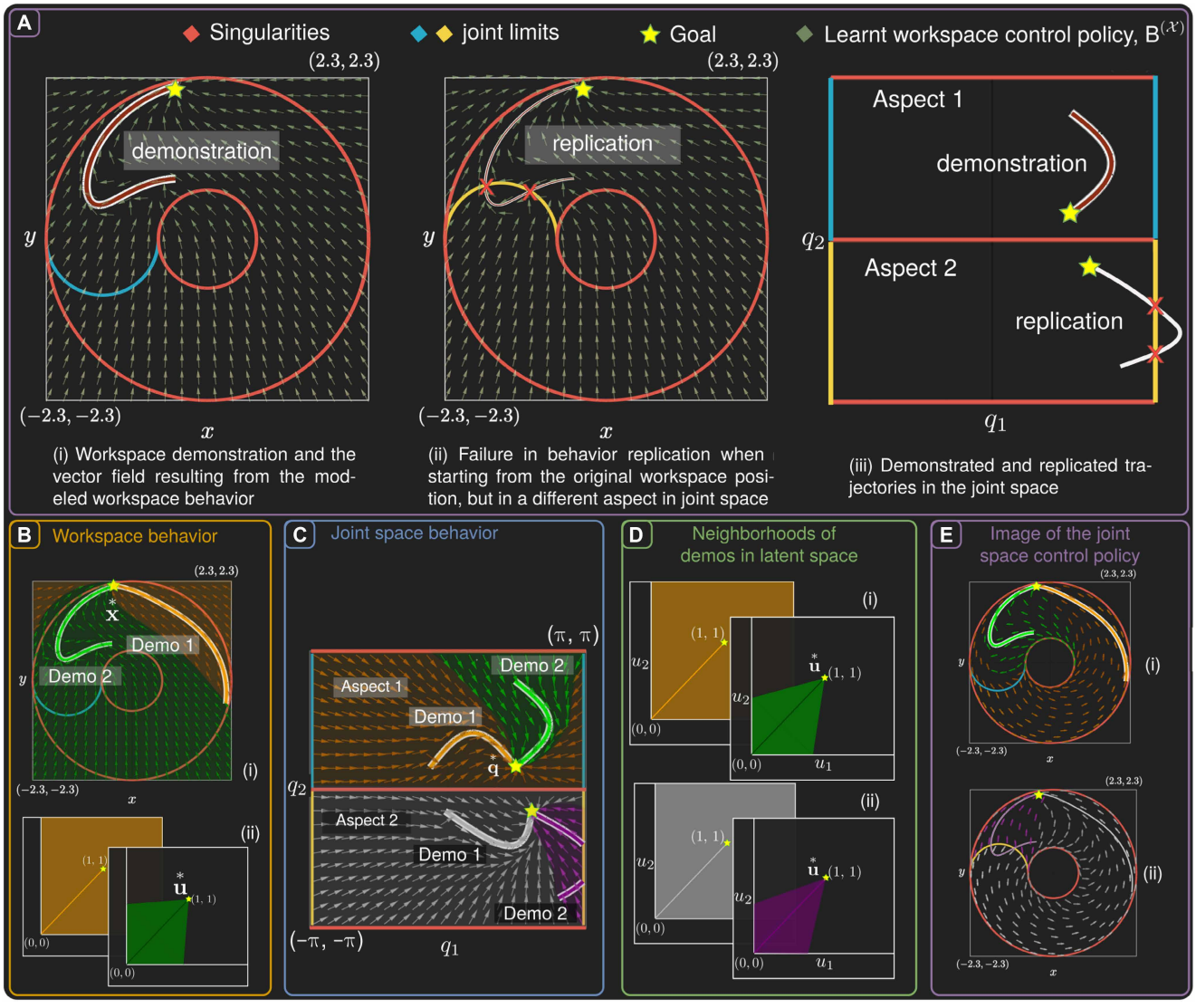


Fig. 8. Learning and replicating the demonstrated behavior for a 2R robot. (**A**) Modeling the workspace behavior and replication from different IKSs: (i) workspace trajectory when robot is in aspect 1, (ii) workspace replication when robot is in aspect 2 leading to joint-limit violation, and (iii) joint space representation of replications. (**B**) Learning workspace behavior from multiple demonstrations: (i) dynamical-system embeddings and the resulting workspace behavior and (ii) latent-space representations with their cones of influence. (**C**) Corresponding joint space trajectories, trimmed for constraint compliance, and the resulting aspect-specific behavior. (**D**) Latent embeddings of the dynamical systems in the aspect-specific behaviors, with their respective cones of influence (i and ii). (**E**) Resulting workspace behavior when executing the joint space control policy from the initial configuration: (i) in aspect 1 and (ii) aspect 2.

dynamical-system embedding $\mathbf{g}_j^{(\mathcal{Q})}$ in this aspect was associated with a localized region of influence in aspect $\mathcal{A}_i$ (see Fig. 8C), defined as the set of joint configurations where $\mathbf{g}_j^{(\mathcal{Q})}$ governed the motion. That is, for any $\mathbf{q}$ in this region, the robot's behavior was given by $\dot{\mathbf{q}} = \mathbf{g}_j^{(\mathcal{Q})}(\mathbf{q})$. An analogous partitioning was performed in the workspace when multiple demonstrations were available. The workspace partitioning spanned the entire domain, in contrast with the joint space, where the partitioning was aspect specific.

These regions were defined via cones of influence (see Fig. 8D) in the latent space

$$\mathcal{L}_j = \{\mathbf{u} \in \mathcal{U} \mid \angle\left(\overset{*}{\mathbf{u}} - \mathbf{u}, \mathbf{v}_j\right) \leq \lambda_j\},\ \mathcal{R}_j = \psi_j(\mathcal{L}_j) \tag{1}$$

Here, $\mathcal{L}_j$ is the cone of influence of control policy $\mathbf{g}_j^{(\mathcal{Q})}(\mathbf{q})$ in the latent space indexed by j. $\mathbf{u}$ refers to a general point in the latent space $\mathcal{U}$, and $\overset{*}{\mathbf{u}} = \mathbf{1}_n^T$ is the latent embedding of the goal, where $\mathbf{1}_n^T$ represents an n-dimensional column vector denoting a latent space point with all coordinates = 1 and T states transpose to indicate that this is a column vector. $\mathbf{v}_j$ is the vector from the start to goal state. $\mathcal{R}_j$ is the region of the joint space corresponding to $\mathcal{L}_j$. The angle λ_j controls the size of the neighborhood influenced by the dynamical system. At inference time, the control law is given by

$$\dot{\mathbf{q}} = \mathbf{g}_j^{(\mathcal{Q})}(\mathbf{q}), \text{where } j = \text{argmax}\{P(\mathcal{R}_j) \mid \mathbf{q} \in \mathcal{R}_j\} \tag{2}$$

Priority $P(\mathcal{R}_j)$ ensured disambiguation when regions overlapped. To prevent high-priority demonstrations from overriding lower-priority behaviors, we adjusted the opening angle λ_j using projection checks

$$\lambda_j = \min\left\{\angle\left(\psi_j^{-1}(\mathbf{q}_l) - \mathbf{1}_n^T, \mathbf{v}_j\right)\right\}_{\mathbf{q}_l \in \mathbf{Q}_l} \quad (3)$$

Here, $\mathbf{Q}_l$ is the trajectory associated with $\mathbf{g}_l^{(Q)}(\mathbf{q})$, $1 \leq l < j$. By combining such aspect-specific behaviors, we constructed the full joint space policy. The process of modeling the joint space policy from multiple workspace demonstrations for the case in Fig. 8 is presented in movie S6. For full derivations and algorithmic procedures, see the "Modeling the robot control policy" section in Supplementary Materials and Methods.

Embedding kinematic intelligence

To generalize user-demonstrated behaviors and ensure safe execution, we embedded kinematic intelligence into the robot's control policy. Up to this point, we had modeled the behavior in both workspace and joint space. Either of these models can act as the control policy. Directly embedding constraints in the workspace model is challenging because of complex singularity features such as nodes and cusps as well as nonlinear boundaries formed by joint limits. Moreover, the workspace lacks dimensional homogeneity in position and orientation, complicating violation-free control. Conversely, joint space singularities form continuous, differentiable C^∞ functions, and joint limits define a linear subspace, making joint space more suitable for embedding constraints and preserving user behavior. We thus adopted the joint space model as the primary control policy, enriched via singularity-based robot categorization and incremental generalization that transfers behavioral variability from workspace to joint space.

We introduced an exhaustive categorization of generic and nongeneric noncuspidal 3R robots into six categories. This revealed how motion becomes constrained, enabling targeted strategies for stable, feasible behavior around constraints. We analyzed the joint space singularities, classifying them by loop structure and turning points to design locally stable, constraint-aware strategies. Categories differed by whether branches looped, intersected, or had folds, dictating directional constraints. Further details are in the "Robot categorization" section of Supplementary Materials and Methods. If the joint space trajectory risked violating limits or singularities, we activated a near-boundary control policy to keep motion in the same aspect, bounded by joint limits and singularities, tailored per robot on the basis of its singularity layout.

We defined safety margins near aspect boundaries and continuously monitored motion. If a violation was imminent, we switched to the near-boundary policy, which temporarily deviated to follow along constraint-safe paths, termed the track cycle, before resuming nominal policy. Near-constraint strategies for all six categories are in the "Behavior near kinematic restraints" section of Supplementary Materials and Methods. The framework integrated these strategies with deterministic connectivity checks of fixed computational complexity, ensuring stable and safe motion near constraints. These checks verified that the configuration was in a feasible aspect and could reach the goal without violating it; otherwise, execution halted, preventing unsafe behavior. Details for all singularities in noncuspidal 3R robots are in the "Certified execution" section of Supplementary Materials and Methods. Kinematic intelligence made the framework robot agnostic during learning but robot specific during execution, allowing robots to safely and faithfully replicate user intent, even near constraints.

Workspace to joint space transfer for enhanced generalization

To improve generalization of the joint space control policy, we introduced an incremental workspace–to–joint space transfer mechanism. This began by uniformly sampling the joint space $\mathcal{Q}$ to generate a grid of candidate joint configurations $\{\mathbf{Q}\}^{\text{sample}}$, which were then mapped to corresponding workspace positions $\{\mathbf{x}\}^{\text{sample}}$ using forward kinematics. For each $\mathbf{x}_i \in \{\mathbf{x}\}^{\text{sample}}$, a transfer trajectory $\mathbf{X}_i^{\text{transfer}}$ was generated using the workspace dynamical system model $B^{(\mathcal{X})}$ such that

$$\mathbf{X}_i^{\text{transfer}} = \{\mathbf{x}_t | \mathbf{x}_{t=0} = \mathbf{x}_i, \lim_{t\to\infty} \mathbf{x}_t = \overset{*}{\mathbf{x}}\} \quad (4)$$

Here, t refers to the time index along the transfer trajectory $\mathbf{X}_i^{\text{transfer}}$ and $\overset{*}{\mathbf{x}}$ is the robot's desired goal position in the workspace. These transfer trajectories were then mapped back into joint space, yielding a set $\{\mathbf{Q}\}^{\text{transfer}}$ of candidate joint space trajectories. Each trajectory was validated and trimmed to remove unsafe or constraint-violating segments using certified constraint checks. To ensure efficient expansion of the joint space model $B^{(\mathcal{Q})}$, we introduced a novelty metric that prioritized trajectories contributing the most new information. The novelty of a trajectory was evaluated by computing its angular deviation in the latent space from existing demonstrations. Let $\psi_j : \mathcal{U} \to \mathcal{Q}$ denote the embedding function of the j^{th} learned dynamical system, and let $\mathbf{q}_t$ be a point on a transfer trajectory $\mathbf{Q}^{\text{transfer}}$. The angular deviation was given by

$$\Theta_j(\mathbf{Q}^{\text{transfer}}) = \left\{\angle\left(\psi_j^{-1}(\mathbf{q}_t) - \mathbf{1}^\top, \mathbf{1}^\top - \mathbf{0}^\top\right)\right\}_{t=1}^{|\mathbf{Q}^{\text{transfer}}|} \quad (5)$$

For an aspect $\mathcal{A}_i$, let $B^{(\mathcal{A}_i)}$ be the corresponding aspect-specific behavior. The novelty score of a trajectory is defined as

$$\text{novelty}(\mathbf{Q}^{\text{transfer}}) = \min_j \{\Theta_j(\mathbf{Q}^{\text{transfer}}) |\ \psi_j \in B^{(\mathcal{A}_i)}\} \quad (6)$$

The most novel trajectory was selected as

$$\mathbf{Q}_{\text{novel}}^{\text{transfer}} = \underset{\mathbf{Q}_i^{\text{transfer}} \in \{\mathbf{Q}\}^{\text{transfer}}}{\text{argmax}}\ \text{novelty}(\mathbf{Q}_i^{\text{transfer}}) \quad (7)$$

This most novel trajectory was then used to update the aspect-specific model $B^{(\mathcal{A}_i)}$, and by extension, the overall joint space behavior $B^{(\mathcal{Q})}$. This ensured that the control policy became richer over time without compromising constraint adherence, stability, or model compactness. Further technical details, including latent space construction and trajectory validation steps, are provided in the Supplementary Materials and Methods section "Incremental update by workspace–to–joint space transfer."

Controlling redundant robots

In robots with redundant degrees of freedom, those exceeding the minimal requirement to define end-effector poses, internal configurations can vary without affecting the external task. This inherent redundancy, although advantageous for optimizing secondary

criteria (such as obstacle avoidance, joint limits, or ergonomics), introduces challenges for characterizing and generalizing robot behavior. Using kinematic intelligence, we addressed this by parameterizing the redundant joint angle, denoted q_R, and reducing the complete 7R robot to a continuum of simpler, equivalent 3R chains, similar to one reported in (*26*). Each value of q_R yielded a distinct robot architecture with well-defined kinematic properties, particularly concerning singularities and solution multiplicity.

This dimensionality reduction enabled a structured encoding of robot behavior given that the redundant angle served as an intrinsic coordinate along which motion solutions could be organized. For instance, motion trajectories executed at different q_R values corresponded to distinct internal configurations achieving the same external task. These configurations could be interpreted as behavioral modes, with q_R acting as a modulating variable. In noncuspidal robots, where IKSs remained separated by singularities for each q_R, this parameterization ensured that the operation mode was consistent and predictable.

This modular arrangement of behaviors enabled redundancy resolution and generalization of behavior across platforms. Crucially, it provided an amodular, explainable framework for managing the solution null space of redundant robots, integrating geometric insight with practical utility in control and learning. For applications demanding safety, repeatability, or human interaction, such an interpretable structure was essential for ensuring trustworthy robot behavior. A detailed discussion of why classical numerical null-space strategies are insufficient in this context is provided in the "Issues with numerical path-planning strategies" section of Supplementary Materials and Methods.

Statistical analysis

Quantitative evaluation focused on the computational performance of the execution pipeline and the learning statistics of the demonstrated behavior (Fig. 6). Wall-clock execution times were recorded for each stage and aggregated across robot instances. Unless otherwise stated, statistics were computed over 21,787 noncuspidal 3R robots. The per-robot performance metrics (Fig. 6) were measured and summarized using the mean (μ), SD (σ), 95th percentile, minimum, and maximum.

For preprocessing steps [Fig. 6A (i and ii)], execution time was measured once per robot. Reactive-control components were evaluated on a grid of 100 joint configurations per robot, obtained by uniformly discretizing the q_2 and q_3 axes. Connectivity checks [Fig. 6C (i)] measured the time to determine connectivity between each grid point and a fixed query configuration [0, 0.5, 0.1] rad, with the mean across queries defining per-robot performance. Violation checks [Fig. 6C (ii)] measured the time to detect kinematic constraint violations. Track-cycle computation [Fig. 6C (iv)] and velocity output [Fig. 6C (v)] were evaluated using synthetic piecewise trajectories (that violated aspect boundaries) generated from the sampled configurations, with the mean across 100 instances defining per-robot performance.

Learning-related metrics for the planning stage (Fig. 6B), one-shot sampling [Fig. 6C (iii)], nominal velocity output [Fig. 6C (vi)], model training duration [Fig. 6D (i)], and trajectory error [Fig. 6D (ii)] were evaluated on a subset of 330 robots. Seven demonstrations of the Multi Models 1 character from the LASA Handwritten Dataset (*38*) were used. Model training duration [Fig. 6D (i)] was recorded during control-policy learning. For the other metrics, IKSs for the demonstration start points were identified, yielding up to 28 starting configurations across feasible aspects. Each configuration generated a joint-space trajectory using the learned policy. Per-robot performance was computed as the mean execution time and trajectory error across these queries. Trajectory error was measured using dynamic time warping (DTW) between reproduced and demonstration joint-space trajectories.

Supplementary Materials

The PDF file includes:
Materials and Methods
Figs. S1 to S22
Tables S1 to S3
Legends for movies S1 to S6

Other Supplementary Material for this manuscript includes the following:
Movies S1 to S6
Data file S1

REFERENCES AND NOTES

1. H. Ravichandar, A. S. Polydoros, S. Chernova, A. Billard, Recent advances in robot learning from demonstration. *Annu. Rev. Control Robot. Auton. Syst.* **3**, 297–330 (2020).
2. E. Gribovskaya, A. Billard, "Combining dynamical systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot" in *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction* (*HRI*) (Association for Computing Machinery, 2008), pp. 33–40.
3. K. Kronander, A. Billard, Learning compliant manipulation through kinesthetic and tactile human-robot interaction. *IEEE Trans. Haptic* **7**, 367–380 (2014).
4. A. L. P. Ureche, K. Umezawa, Y. Nakamura, A. Billard, Task parameterization using continuous constraints extracted from human demonstrations. *IEEE Trans. Robot.* **31**, 1458–1471 (2015).
5. A. Pervez, A. Ali, J.-H. Ryu, D. Lee, "Novel learning from demonstration approach for repetitive teleoperation tasks" in *2017 IEEE World Haptics Conference* (*WHC*) (IEEE, 2017), pp. 60–65.
6. N. Jaquier, M. C. Welle, A. Gams, K. Yao, B. Fichera, A. Billard, A. Ude, T. Asfour, D. Kragic, Transfer learning in robotics: An upcoming breakthrough?: A review of promises and challenges. *Int. J. Robot. Res.* **44**, 465–485 (2024).
7. R. R. Ma, A. Spiers, A. M. Dollar, "M2 gripper: Extending the dexterity of a simple, underactuated gripper" in *Advances in Reconfigurable Mechanisms and Robots II*, X. Ding, X. Kong, J. S. Dai, Eds. (Springer International Publishing, 2016), pp. 795–805.
8. B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration. *Robot. Auton. Syst.* **57**, 469–483 (2009).
9. S. Calinon, F. Guenter, A. Billard, On learning, representing, and generalizing a task in a humanoid robot. *IEEE Trans. Syst. Man Cybern. B Cybern.* **37**, 286–298 (2007).
10. A. Shaw, J. Lee, J. Park, "Constrained dynamic movement primitives for collision avoidance in novel environments" in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2023), pp. 3672–3679.
11. L. Yang, M. Fischer, D. Kragic, "Enhancing learning from demonstration with DLS-IK and ProMPs" in *2024 29th International Conference on Automation and Computing* (*ICAC*) (IEEE, 2024), pp. 1–6.
12. E. Pignat, S. Calinon, Learning from demonstration using products of experts: Applications to manipulation and task prioritization. *Int. J. Robot. Res.* **41**, 163–188 (2022).
13. L. Bakker, V. Koltun, M. Toussaint, "TamedPUMA: Safe and stable imitation learning with geometric fabrics" in *Proceedings of the 7th Annual Learning for Dynamics and Control* (*L4DC*) *Conference* (Proceedings of Machine Learning Research, 2025).
14. B. Trabucco, M. Phielipp, G. Berseth, "AnyMorph: Learning transferable polices by inferring agent morphology" in *Proceedings of the 39th International Conference on Machine Learning* (Proceedings of Machine Learning Research, 2022), pp. 21677–21691.
15. C. Sferrazza, D.-M. Huang, F. Liu, J. Lee, P. Abbeel, "Body transformer: Leveraging robot embodiment for policy learning" in *Proceedings of the 8th Annual Conference on Robot Learning* (Proceedings of Machine Learning Research, 2024), pp. 3407–3424.
16. H. Klein, N. Jaquier, A. Meixner, T. Asfour, "A Riemannian take on human motion analysis and retargeting" in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) (IEEE, 2022), pp. 5210–5217.
17. N. Makondo, B. Rosman, O. Hasegawa, "Knowledge transfer for learning robot models via local procrustes analysis" in *2015 IEEE-RAS 15th International Conference on Humanoid Robots* (*Humanoids*) (IEEE, 2015), pp. 1075–1082.

18. P. Pastor, H. Hoffmann, T. Asfour, S. Schaal, "Learning and generalization of motor skills by learning from demonstration" in *2009 IEEE International Conference on Robotics and Automation* (IEEE, 2009), pp. 763–768.
19. S. Schaal, J. Peters, J. Nakanishi, A. Ijspeert, "Learning movement primitives" in *Robotics Research. The Eleventh International Symposium*, P. Dario, R. Chatila, Eds. (Springer, 2005), pp. 561–572.
20. M. Ewerton, G. Maeda, G. Kollegger, J. Wiemeyer, J. Peters, "Incremental imitation learning of context-dependent motor skills" in *2016 IEEE-RAS 16th International Conference on Humanoid Robots* (*Humanoids*) (IEEE, 2016), pp. 351–358.
21. D. H. Salunkhe, D. Chablat, P. Wenger, "Trajectory planning issues in cuspidal commercial robots" in *2023 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2023), pp. 7426–7432.
22. D. H. Salunkhe, T. Marauli, A. Müller, D. Chablat, P. Wenger, Kinematic issues in 6R cuspidal robots, guidelines for path planning and deciding cuspidality. *Int. J. Robot. Res.* **44**, 1035–1054 (2025).
23. P. Wenger, El J. Omri, "Changing posture for cuspidal robot manipulators" in *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE, 1996), pp. 3173–3178.
24. P. Wenger, "Design of cuspidal and non-cuspidal robot manipulators" in *Proceedings of International Conference on Robotics and Automation* (IEEE, 1997), pp. 2172–2177.
25. P. Wenger, Uniqueness domains and regions of feasible paths for cuspidal manipulators. *IEEE Trans. Robot.* **20**, 745–750 (2004).
26. D. H. Salunkhe, S. Gupta, A. Billard, Cuspidal redundant robots: Classification of infinitely many IKS of special classes of 7R robots. *IEEE Rob. Autom. Lett.* **10**, 12509–12516 (2025).
27. J. W. Burdick, A classification of 3R regional manipulator singularities and geometries. *Mech. Mach. Theory* **30**, 71–89 (1995).
28. P. Borrel, A. Liegeois, "A study of multiple manipulator inverse kinematic solutions with applications to trajectory planning and workspace determination" in *Proceedings of the 1986 IEEE International Conference on Robotics and Automation* (IEEE, 1986), pp. 1180–1185.
29. P. Wenger, Classification of 3R positioning manipulators. *J. Mech. Des.* **120**, 327–332 (1998).
30. D. Paganelli, "Topological analysis of singularity loci for serial and parallel manipulators," thesis, Universita di Bologna, Bologna, Italy (2008).
31. M. Asgari, I. Bonev, C. Gosselin, Singularities of ABB's YuMi 7-DOF robot arm. *Mech. Mach. Theory* **205**, 105884 (2025).
32. A. J. Elias, J. T. Wen, Redundancy parameterization and inverse kinematics of 7-DOF revolute manipulators. *Mech. Mach. Theory* **204**, 105824 (2024).
33. S. Gupta, A. Nayak, A. Billard, Compact oneshot modeling of high dimensional demonstrations using Laplacian eigenmaps. *IEEE Trans. Robot.* **42**, 1468–1484 (2026).
34. A. Dragan, K. Lee, S. Srinivasa, "Legibility and predictability of robot motion" in *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*, *HRI '13* (IEEE, 2013), pp. 301–308.
35. J. Geldenbott, K. Leung, "Legible and proactive robot planning for prosocial human-robot interactions" in *2024 IEEE International Conference on Robotics and Automation* (*ICRA*) (IEEE, 2024), pp.13397–13403.
36. P. Wenger, D. Chablat, A review of cuspidal serial and parallel manipulators. *J. Mech. Robot.* **15**, 040801 (2022).
37. D. H. Salunkhe, C. Spartalis, J. Capco, D. Chablat, P. Wenger, Necessary and sufficient condition for a generic 3R serial manipulator to be cuspidal. *Mech. Mach. Theory* **171**, 104729 (2022).
38. S. M. Khansari-Zadeh, A. Billard, Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Trans. Robot.* **27**, 943–957 (2011).

Acknowledgments: We thank D. Jacquemont for invaluable assistance in setting up and executing the robot experiments and N. Jeannot for providing the demonstration data used in this work. We are also grateful to I. Derivaz-Rabii for the essential administrative support during the project. We also used ChatGPT to assist with revising and refining the language in this paper. **Funding:** This work was supported by the EU project DARKO under grant H2020 ICT-46-2020 and Horizon Europe under grant 101070596, euROBIN. **Author contributions:** Conceptualization, simulation, investigation, analysis, visualization, and writing: S.G. and D.H.S. Resources, supervision, writing, and revision: A.B. **Competing interests:** The authors declare that they have no competing interests. **Data, code, and materials availability:** All data needed to support the conclusions of this manuscript are included in the main text or Supplementary Materials. All code is available online at the following DOI: https://zenodo.org/records/18992244. All other materials, electronics, and hardware used in this work are commercially available.

Submitted 30 June 2025
Accepted 17 March 2026
Published 15 April 2026
10.1126/scirobotics.aea1995

Science Robotics

Demonstrate once, execute on many: Kinematic intelligence for cross-robot skill transfer

Sthithpragya Gupta, Durgesh Haribhau Salunkhe, and Aude Billard

Sci. Robot. **11** (113), eaea1995. DOI: 10.1126/scirobotics.aea1995

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.