Demonstrate Once, Execute on Many: Kinematic Intelligence for Cross-robot Skill Transfer

Sthithpragya Gupta[1*†], Durgesh Haribhau Salunkhe[1†], Aude Billard[1]

[1]Learning Algorithms and Systems Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne & 1015, Switzerland

[†]These authors contributed equally to this work.

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. Yet 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 behaviour, 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 embed them directly into the control policy from the outset. The approach takes one or multiple demonstrations, extracts a globally stable dynamical system, and produces behaviours that remain valid across robots with different kinematic structures. Our method is grounded in a comprehensive analytical classification of non-cuspidal 3-revolute (3R) arms, which form the building blocks of many industrial manipulators. This classification enables a joint-space policy that preserves user intent while automatically adapting to robot-specific constraints. We validate the framework on a diverse set of simulated and real robots, redundant and non-redundant, with varied link geometries and joint configurations, and show that the demonstrated

skill can be executed safely, smoothly, and consistently on all of them. By showing how analytical kinematic properties can be leveraged for cross-robot skill transfer, our work moves LfD closer to scalable, intuitive robot teaching for non-experts, enabling safe and reliable deployment without retraining.

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 behaviour. 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 Figure 1). This removes the need for coding and lowers the barrier for human-robot interaction. But while LfD simplifies how we teach robots, there's 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's not the case today. The problem becomes even more difficult as 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 overcome this challenge, we argue that robots need more than just the ability to mimic task demonstrations. They need to be equipped with what we call kinematic intelligence. This means that the robot must internally encode and account for its own kinematic constraints, joint limits, and feasible movement paths. Rather than relying on extensive demonstrations that implicitly avoid failures, our approach explicitly incorporates this structural awareness into the robot's control policy. Kinematic intelligence plays a key role in enabling robots to generalize behaviour across different body structures. This is important because the human body rarely resembles the mechanical

structure of robots. Even among robots, different models often have very different link lengths and joint orientations, resulting in different types of unstable configurations (singularities). Yet the task being demonstrated, such as wiping a surface, picking up an object, or following a curved path, remains the same. To ensure that a behaviour 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 is a promising solution where a task is transferred by reusing a previously learned behaviour. However, transfer learning also faces challenges related to explainability, abstraction of transfer, and transfer metrics (*6*). Kinematic intelligence bridges this gap, enabling a system trained on one robot (or even from human-guided input) to function safely and effectively on a different robot without needing expert intervention.

Challenges set forth by kinematic constraints in transfer learning: Previous work in LfD has made significant progress in learning smooth and repeatable behaviours from demonstrations, particularly in simple or well-controlled environments. Early methods focused on directly imitating motion trajectories in the workspace, allowing robots to reproduce tasks like reaching, drawing, or object manipulation (*7, 8*). These approaches worked well when the target robot had similar physical characteristics to the one used during training. Some probabilistic models offer flexibility but still require separate handling of kinematic feasibility (*9, 10*). Other techniques, though more integrated, depend on detailed tuning or assumptions about the robot's workspace (*11, 12*). However, when the robot changes, or when its physical structure leads to limitations near joint limits or singular configurations, the learned behaviour 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 (*13–16*). This abstract version is then adapted for each robot individually. While this approach shows promise, it often requires multiple demonstrations, extensive parameter tuning, or manual safety corrections near the robot's constraints (*17, 18*) 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 (*19*). 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. While useful, these strategies treat the robot's motion constraints as a separate concern rather than as part of the learning process itself. However, due to 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 (*20–24*).

More recently, some data-driven methods aim to build shared task spaces or embeddings that allow robots to learn behaviours 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, as robot teaching is a time-consuming endeavour for a non-expert 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) anthropomorphic 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 degrees of freedom (seven) than strictly necessary (six). However, redundancy also poses a challenge for learning and reproducing behaviour. Unlike non-redundant robots, redundant robots admit infinitely many joint-space solutions for a given task, see Figure 1(Top row), making it difficult to learn and generalize violation-free behaviours consistently across different configurations.

Kinematic analysis to facilitate transfer across robotic manipulators To analyze the behaviour of 7-DoF non-cuspidal manipulators, it is useful to decompose them into smaller kinematic units, or subchains, that isolate the functional components of motion. Most modern 7-DoF arms employ a spherical wrist, composed of three intersecting rotational joints that control the end-effector orientation. Anthropomorphic manipulators, widely adopted in research and certain industrial

applications such as the Kuka LWR iiwa and Motoman SIA series, exhibit non-cuspidal kinematic architectures (*25*). For such robots, inverse kinematic solutions (IKS) 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 is also non-cuspidal, exhibiting two inverse kinematic solutions separated by $\sin(\theta_w) = 0$, where θ_w denotes the second wrist joint. Hence, the analytical framework proposed for non-cuspidal $3R$ (three revolute joints) 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 3D positioning, as shown in Figure 1(second row).

To manage the redundancy-induced ambiguity, the $4R$ positional chain can be treated as a family of $3R$ non-cuspidal subchains. By setting the angle of a redundant joint in the $4R$ subchain to a specific value, we effectively eliminate one degree of freedom, reducing the $4R$ subchain to a non-redundant $3R$ subchain that is just sufficient for spatial positioning. Thus, the behaviour 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 within the original $4R$ subchain. While each of these $3R$ subchains abide by the same joint limit constraints, they can exhibit different kinematic properties, such as singularities, number of IK solutions, and the feasible joint space regions.

This decomposition provides a systematic means to analyze the core positional capabilities of 7-DoF anthropomorphic manipulators through a family of non-cuspidal $3R$ subchains. These $3R$ units constitute the kinematic foundation of the arm and form the basis for our global, constraint-aware analysis. Since 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 (*26*), $3R$ manipulators can be grouped as generic, non-generic, 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. Non-generic robots correspond to special geometries where these surfaces intersect or bifurcate, leading to multiple branches or self-intersections. Degenerate cases arise typically in manipulators with symmetric link lengths or right-angle joint arrangements. In such configurations, two joint axes may align or intersect, reducing the arm'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 non-cuspidal robots, generic, non-generic, and degenerate, thus ensuring constraint-aware and predictable behavior regardless of design variations.

We design 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 behaviour is automatically adapted to each robot's structure in a safe and stable way, see Figure 1(3r and 4th rows). Specifically, we use a topological and differential classification of singularities to understand how different parts of the robot's configuration space are separated by constraints (refer to Figure 2). This classification allows us to identify singularity-free zones, distinguish between reachable and unreachable regions, and 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 behaviour that is safe, robust, and executable on many different robots. We demonstrate this framework across a range of robots, including both redundant and non-redundant arms, each with different kinematic structures and singularity patterns. This is achieved without any retraining or architecture-specific reprogramming.

RESULTS

We demonstrate transfer of skills in simulation with a series of non-redundant $3R$ robots. For redundant robots, we consider the KUKA IIWA LWR 7 and the Maira robot, both seven-degree-of-freedom commercially available robot arms. The Maira robot has a peculiarity in that it has a degeneracy, but the degenerate configurations remain outside the feasible joint space. We observe across all robot configurations that the system successfully acquires a generalised policy of the user behaviour, resulting in smooth, stable, and physically feasible robot execution (refer to Figure 3 and Figure 8). 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 demonstrate redundancy parametrisation and how it facilitates behaviour transfer to redundant robots (refer to Figure 4). An overview of the challenges, a brief motivation to kinematic analysis, and behaviour transfer is presented in Movie S1.

Kinematic intelligence

Kinematic intelligence refers to the embedding of analytical kinematic properties into the control framework using deterministic, fixed-compute algorithms. This enables certified, constraint-compliant path planning and policy synthesis that generalizes across robot embodiments without retraining.

A central result of this work is a theoretically grounded classification of all non-cuspidal $3R$ manipulators into six distinct categories, each defined by the global structure of their kinematic constraints. This classification forms the backbone of our kinematic intelligence framework, enabling the synthesis of robot-specific control strategies that are inherently safe and generalizable.

The classification arises from an algebraic and topological analysis of the robot's Jacobian determinant. The factorization and root structure of this determinant are determined by the robot's design parameters, such as link lengths and joint offsets, which define the location and shape of singular configurations in the joint space. These singularities, along with joint limits, partition the joint space into distinct feasible regions: referred to in the remainder of this paper as **aspects** (*27*).

Each category captures a specific class of kinematic behaviours, characterized by how singular-

ity curves fold, loop, intersect to segment the joint space. Crucially, the structure of each class also dictates a corresponding near-boundary control strategy. Rather than treating constraint handling as an afterthought, the categorization informs how corrective actions should be embedded directly into the control policy. Once a robot's class is known, the corresponding strategy can be instantiated without further tuning or demonstration, yielding a scalable method for constraint-compliant behaviour transfer across robot types.

Figure 2 provides an overview of the topological structure of the singularities and joint limits in each class. This structure is not just descriptive, it is operational. It allows us to build controllers that adapt dynamically to robot-specific constraints while preserving the fidelity of the demonstrated task.

Robot Categorization

Through a detailed algebraic analysis of the Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ where $\mathbf{q}$ represents the robot's joint configuration, we establish a structured classification scheme for noncuspidal $3R$ robots.

The expression of singularities for $3R$ robots is only affected by the actuation of the second and third joints (q_2 and q_3 respectively). Consequently, the kinematic constraints imposed by singular configurations can be analysed in a reduced space spanned by the second and third joint's angle values, hereby referred to as the $q_2 - q_3$ slice of the joint space. The expression for $\det \mathbf{J}(\mathbf{q})$ can be decomposed into upto three irreducible factors, whose zero sets define closed branches on the configuration torus $\mathbb{T}^2$. Each factor exhibits distinct winding numbers (n_1, n_2), denoting how many times they wrap around the torus generators associated with joints q_2 and q_3. Each individual encirclement by the factor is a unqiue branch of that factor. Building on this topological insight, we introduce a novel symbolic notation of the form $(n_1, n_2)[n_3, n_4]$, which describes both the global winding behavior and the differential structure, where n_3 and n_4 capture the number of horizontal and vertical turning points, respectively that can occur on a branch of the factor. This notation extends the existing homotopy classification (*28, 29*), which relied only on the winding numbers. While this classification was exhaustive, it remained insufficient for planning motions along constraints, as robots within the same category could exhibit distinct differential properties-necessitating case-by-case treatment during execution. In contrast, our proposed categorization

incorporates both topological and differential properties, ensuring that robots within the same class exhibit identical constraint structures and thus permit a unified path planning strategy near singularities.

Applying this framework across both generic and non-generic noncuspidal $3R$ robots, we identify six canonical robot categories that comprehensively describe the global structure of the joint space. To arrive at this categorisation, we inspect the factor(s) of the expression of $\det \mathbf{J}(\mathbf{q})$ for the given robot and generate a description of their accompanying branches. The six categories as presented in Figure 2(B) are: (I) non-loop, non-intersecting factors with ONLY horizontal turning points - only factors with branches of type $(1,0)[\geq 2,0]$; (II) non-loop, non-intersecting branches with NO horizontal turning points - factors with branch types $(0,1)[0,*]$ where $*$ can assume any non-negative value; (III) non-loop intersecting branches - two or more factors with non-loop type branches that intersect; (IV) non-loop, non-intersecting branches with non-zero vertical turning points - factors with branch type $(1,0)[2,>0]$; (V) loop non-intersecting branches - factor with branch type $(0,0)[2,\geq 2]$; and (VI) intersecting branches with a loop - intersecting factors where one of the factors has branch that makes a loop. Each class reflects a distinct mode of constraint-induced partitioning in joint space and corresponds to qualitatively different boundary-following behaviours. This classification organizes the singularities across $3R$ manipulators, and provides a predictive basis for designing constraint-aware control policies that generalize across robot geometries. Multiple non-cuspidal robots with varying kinematic properties with their respective DH parameters are presented in Movie S2.

When a user demonstrates a task, the behaviour is encoded as a task-level objective, which each robot then interprets and executes in a constraint-aware manner based on its own categorisation. This approach contrasts with traditional LfD methods that either rely on post-hoc corrections or require multiple demonstrations tailored to each robot. Instead, our framework generalizes from a single demonstration by leveraging an internally encoded model of joint-space structure that is both robot-specific and reusable across the entire category. For example, if a demonstrated behaviour would cause a new robot to approach a singular configuration or violate a joint limit, the system automatically adjusts the trajectory to follow the boundary of the feasible region until it can safely return to the nominal path. This ensures continuity and stability in the robot's motion while faithfully reproducing the demonstrated behaviour.

In essence, the structural classification functions as a map that each robot uses to safely interpret and execute a shared task. By integrating this into the learning process, we move beyond treating kinematic constraints as external concerns and instead make them an intrinsic part of behaviour generalization. A comprehensive exposition of the mathematical framework of the categorization, defining properties of each robot class, their topological implications in joint space, the formalization of kinematic constraints and accompanying algorithms , is presented in the Supplementary Methods section 'Robot Categorisation'.

Near-constraint strategy

A key outcome of this work is the establishment of a one-to-one correspondence between the robot's kinematic category and the control strategy that governs robot behaviour along **aspect** boundaries. Each joint-space aspect is bounded on at least one side by a factor of $\det \mathbf{J}(\mathbf{q})$, and on the remaining sides (if any) by the joint limits. While traversal along joint limits is straightforward since they form a linear variety in joint space, motion along the factor boundary is non-trivial and varies significantly across categories.

The core insight is that the traversal along the factor boundary within an aspect dictates the constraint-aware motion strategy, and this traversal is uniquely determined by the kinematic category of the robot. Each category is characterised by a key boundary-aligned direction(s), vector(s) intrinsic to the structure of the singularity factor, that governs motion near the constraint (see Figure 2(C)). This direction is projected onto the tangent of the boundary to yield the feasible motion along it. Crucially, this approach remains robust even in the presence of turning points or curvature reversals along the singularity branch: the projected motion does not stall or violate constraints.

Robots from Categories I, II, IV, and V exhibit a single key boundary-aligned direction viz-horizontal (along q_2 axis) in Category I, vertical (along q_3 axis) in Category II, a horizontal direction with intermittent vertical switching in Category IV, and along a closed-loop in Category V (see Figure 2(C1) through (C6)). In contrast, Categories III and VI are defined by a pair of distinct boundary-aligned directions. These arise due to the presence of two distinct factors in $\det \mathbf{J}(\mathbf{q})$, corresponding to intersecting singularities. Each factor contributes its own characteristic direction for boundary traversal. This is notably different from Category IV, where only a single

factor exists, despite the presence of mixed motion, meaning that both horizontal and vertical behaviours emerge along the same singularity branch, rather than from multiple factors. Note that the occurrence of multiple factors does not necessitate multiple directions. Robots from Category I can have upto two factors, yet since the branch types of these factors is identical ($(1,0)[\geq 2,0]$), the horizontal direction suffices. We refer to the structured control strategies incorporating these category-specific boundary-aligned directions for governing the robot behaviour near constraints as *track cycles*. When the nominal dynamics modelled from the user behaviour generates a trajectory $\mathbf{Q}$ that temporarily exits the current aspect, convergence to the goal necessitates eventual reentry. The track cycle defines a stable control policy that takes effect between the first exit and last reentry configurations. Within this segment, motion is redirected along the boundaries of the aspect. Overall, the track cycle encapsulates the correct sequence of transitions across boundary regions, driving the robot from any boundary configuration back to a safe configuration, once again consistent with the original trajectory $\mathbf{Q}$. We adopt a hybrid control framework that overlays track cycle policy onto the nominal joint-space dynamics learned from demonstration. These corrective policies activate preemptively as the robot approaches predefined safety margins around singular regions or joint limits, while following the trajectory generated by the nominal dynamics. The readers are encouraged to refer to section titled ”Behaviour near Kinematic Constraints” in Supplementary Methods for further details on factors, their properties and track cycles.

Transfer Across Robots

To evaluate cross-robot behaviour transfer, we considered a scenario where a user demonstrates a task, such as drawing shapes or letters, hitting an object, rearranging an object and throwing, and different robots attempt to reproduce the behaviour. These robots vary in mechanical design: some are non-redundant (with just enough degrees of freedom), while others are redundant (with more joints than necessary). From this, our system generates robot-specific joint-space control policies that preserve the demonstrated intent while ensuring safety. These results validate the core hypothesis: by embedding structural constraints directly into the learning framework, a behaviour demonstrated just once can be safely and reliably transferred to robots with different kinematics, all without needing expert tuning or additional data.

Transfer to Non-redundant robots

To evaluate how a single demonstration can be extended across multiple non-redundant robots with different kinematic structures, we tested our framework on four $3R$ robots - one each from Categories 1 through 4. For each robot, the goal was to generate a control policy that generalised user-demonstrated motion but also respected robot-specific constraints like singularities and joint limits. As shown in Figure 3, our method models joint-space behaviours 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 Figure 3 (A) through Figure 3 (D) respectively. The figure outlines each step of the framework: from modelling the workspace behaviour (Figure 3 Step 1), learning aspect-specific joint-space behaviours (Figure 3 Step 2), identifying feasible transfer trajectories (Figure 3 Step 3), and workspace-to-joint space transfer and updating the policy based on most novel transfer trajectories (Figure 3 Step 4). Finally, Figure 3 Step 5 formulates the resulting workspace motion that is faithful to the original demonstration while staying entirely within safe, constraint-compliant regions. Across all tested robot categories, the learned joint-space policy remains free from constraint violations without requiring retraining, enabling robust and scalable behaviour transfer to diverse non-redundant robots from just a single demonstration.

Transfer to Redundant robots

To assess transfer to redundant robots, we consider a 7 DoF anthropomorphic arm. The user demonstrated that the end-effector trajectory is used to generate a control policy for the position control of these redundant arms.

The redundant robots can be parameterized by a chosen angle (q_R) to represent a family of non-redundant $6R$ robots. The architecture of the $6R$ robot depends on the *arm angle* or *self-motion angle*. The arm angle is not always easy to define and interpret (*30*), and different methods have been proposed earlier to parameterize the redundant motion of $7R$ robots (*31*), such as the *shoulder-elbow-wrist* (SEW) angle. It was shown in (*25*) that $\det \mathbf{J}(\mathbf{q})$ of the redundant robots can be obtained as a function of q_R, implying that the redundant robot can be treated as a non-redundant robot at any arbitrary value of q_R. In practice, we discretise a set of candidate q_R values, run the full classification and connectivity analysis for the induced $3R$ subchain at each candidate, and select a *fixed* q_R that

realises the task within a single feasible aspect with joint limits and singularity violation. This q_R is then kept constant during execution, so that, from the viewpoint of the positional chain, the robot behaves like a non-redundant $3R$ manipulator and no discontinuities due to online q_R switching are introduced.

The Figure 4 outlines the process of redundancy parametrisation into five behavioural modes for brevity

. The redundant $4R$ positional subchain is identified for each robot in Figure 4, and parametrised using different values for q_R - 0°, 30°, 45°, 60°, and 90° in Figure 4(A). This yields the five behavioural modes, each characterised by its equivalent $3R$ subchains. All of the $3R$ subchains for the robot belong to Category III. The branch types for $q_R = 0$ are $(1,0)[\infty,0]$, and $(1,1)[0,*]$ while for the rest are $(1,0)[\infty,0]$, and $(0,1)[0,*]$. Figure 4(B) presents the varying kinematic constraints across these behavioural modes in joint space. At each q_R value, multiple IK solutions for the starting position of the $3R$ subchain will exist. Some of these configurations are feasible for task completion, Figure 4(B) and produce workspace behaviour Figure 4(C). However, some of the other IK solutions (silver) may not be feasible. For $q_R = 0°$, and 60∘, feasible behaviours are not detected as the goal joint configuration in the respective

aspect of joint space lies in the joint limit constraint boundary and hence safe task execution is not feasible. Notice that the feasible workspace Figure 4(C) and (D) varies significantly with the q_R value. The feasible workspace is bounded by the workspace singularities (red). For different values of redundant angle, one builds a stack of behavioural modes, each encoding the demonstrated motion pattern, allowing smooth interpolation or selective retrieval depending on task constraints. As a result of this modular arrangement, the safety and robustness properties of every individual behaviour mode guarantees a violation-free control policy for anthropomorphic redundant robots.

Certified Execution

To model the robot control policy from joint trajectories, each trajectory is modelled a globally stable dynamical system (DS) in a one-shot manner (*32*). The proposed control policy, constructed as a composition of such stable dynamical systems, is resilient to both temporal and spatial perturbations. While small deviations are passively corrected, larger displacements, such as those moving the robot to a different joint space region, trigger a topology-aware feasibility analysis to assess whether the robot's current aspect permits a constraint-compliant trajectory to the demonstrated goal.

Crucially, not all aspects of the joint space admit a valid inverse kinematic solution (IKS) to the demonstrated goal position. To address this, we develop certified factor-specific connectivity checks grounded in the analytical structure of $\det \mathbf{J}(\mathbf{q})$. There exist five factor types based on their branch types, $(1,0)[\geq 2, 0]$, $(0,1)[0, *]$, $(1,0)[2, > 0]$, $(0,0)[2, \geq 2]$, and $(1,1)[0, *]$ (see Supplementary Methods section *Robot Categorisation*). Each irreducible factor of $\det \mathbf{J}$ partitions the joint space into multiple regions, and feasibility is established only if the start and goal configurations belong to the same partition with respect to every factor. We leverage the global topological structure of the robot's kinematic space into the control logic, a reasoning capability missing from standard learning-from-demonstration frameworks. Our method enables us to find connectivity in a deterministic and fast manner.

To leverage the certified kinematics properties while learning the demonstrated behaviour and formulating a control policy, the framework undertakes a multi-step process summarised in Figure 5, which depicts the implementation pipeline for executing a learned DS policy on a single robot. For each $3R$ robot at a fixed value of q_R, Panel (A) groups the offline, one-time pre-processing steps to determine the robot category (details in the Supplementary Material, section "Robot Categorisation"). Panel (B) then corresponds to the planning stage, where we learn the aspect-specific nominal joint behaviour(see the Supplementary Material, section "Modelling the Robot Control Policy"). The resulting policy governs the reactive control loop shown in Panel (C), which combines connectivity checks with the attractor, constraint-violation monitoring, nominal DS policy execution, and track-cycle computations to ensure faithful replication of the demonstrated behaviour while respecting kinematic constraints (further details in "Modelling the Robot Control Policy"). The detailed pipeline, together with the wall-clock timings and time latency induced by

each step, is reported in Figure 5; these timings show that pre-processing is negligible compared to demonstration time and that both planning and reactive control incur only modest computational overhead, making the certified execution layer compatible with real-time control on the tested platforms.

Figure 6 illustrates certified execution and its aspect-based failure modes on a Category 1 arm whose workspace mixes regions with four and two IKS. From any chosen IKS, the controller executes the demonstration only if a feasible aspect connects start and goal; otherwise, it halts safely. For non-redundant 6R arms, failures occur when no admissible path exists or perturbations drive the state into joint-limit or singularity margins, triggering track cycles or a safe stop. The same logic extends to wrist-partitioned 7R arms (Figure 4), where tasks are declared infeasible if no redundant angle q_R yields a feasible aspect. A detailed discussion of the failure modes is presented in the Supplementary Materials, section 'Robustness'.

Experimental evaluation of "Demonstrate once, execute on many"

We experimentally validate the "demonstrate once, execute on many" principle on three industrial manipulators: a compact 6-DoF Duatic Dyna arm with tighter joint limits, a 7-DoF KUKA LWR iiwa7 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 demonstrates a path resembling the letters of "SCIENCE" once per letter using a motion-capture glove and OptiTrack. The recorded trajectories are encoded as DS policies and replayed on the two 7-DoF arms, which reproduce the characters without joint-limit or singularity violations across multiple aspects and redundant angles (refer to Movies S3–S4).

In Experiment 2, we construct a mock multi-robot assembly line with three skills—pushing, pick-and-place, and throwing. Each skill is demonstrated once, encoded once as a DS in joint space, and executed on three arms without retraining. Pushing and throwing drive joints close to workspace boundaries and induce large excursions, while pick-and-place requires aspect queries

enforced by connectivity checks. The same policy is reused; only the kinematic embedding and certified execution layer change, keeping trajectories within feasible aspects and redirecting them along analytic track cycles near constraints. Figure 8 shows the three robots performing the three skills; full executions appear in Movie S6.

DISCUSSION

Teaching robots through demonstration is becoming a natural and effective way for humans to communicate complex skills. As robotic platforms diversify, the ability to share learned behaviours across different systems, without reprogramming each one, becomes increasingly vital. This is the central goal of transfer learning from demonstration: to teach once and generalize broadly.

We propose that integrating behaviour learning with robot-specific constraint enforcement, by embedding analytic kinematic knowledge directly into the control policy, enables proactive avoidance of infeasible configurations without relying on post-hoc checks. Our framework advances Learning from Demonstration (LfD) by enabling generalizable and safe skill transfer across a wide class of robot architectures, through a principle we term kinematic intelligence. It separates user intent from robot morphology, making intent transferable across different robots.

Each demonstrated trajectory is modelled as a globally asymptotically stable dynamical system (DS), ensuring convergence to its target configuration. This stability at the most fundamental level yields robustness to temporal and spatial perturbations, without requiring reactive correction. As the control policy is incrementally built by adding more demonstrations, the stability of each DS ensures that the entire policy remains stable at every stage of learning.

Although behaviour is demonstrated in workspace, execution occurs in joint space. Therefore, reliable task replication requires embedding user intent in the joint space while accounting for robot-specific constraints. Our framework achieves this through: (1) a topological classification of singularities to categorise robot types, (2) connectivity analysis for assessing feasibility, and (3) constraint-aware policy synthesis. By encoding control policies within joint space aspects, regions bounded by singularity surfaces and joint limits, trajectories are guaranteed to remain within feasible boundaries, avoiding violations.

If a predicted trajectory nears a constraint, a near-boundary strategy redirects motion along a

safe path called the track cycle, rejoining the original trajectory once clear. These strategies are category-specific and embedded into the control logic, making the behaviour certifiable, guaranteed to be safe through bounded-complexity feasibility checks. All connectivity and constraint handling methods are derived analytically from the robot's structure, ensuring that motion is grounded in the robot's true capabilities.

The framework supports generalization through a transfer mechanism: joint configurations are sampled and mapped to the workspace, where the workspace model generates transfer trajectories. These are inverted into joint space, validated, and trimmed to retain only feasible segments. A novelty metric ranks these trajectories based on how much new information they contribute, those that expand the joint space coverage are incorporated into the control policy. This allows the system to generalize from limited examples without compromising safety.

Experiments validate the robustness of this approach on anthropomorphic robots with differing kinematics. A common demonstration is used to synthesize robot-specific joint-space controllers, enabling safe execution without retraining. As shown in Figure 4, the framework adapts to redundancy: it executes successfully for all redundant angles in feasible aspects, and correctly halts when a configuration becomes infeasible. This transparency in decision-making enhances user trust by making robot behaviour predictable and legible (*33, 34*).

While recent collaborative and industrial robots are cuspidal (e.g., ABB YuMi/GoFa, Fanuc CRX, Kinova Link 6) or incorporate non-spherical wrists (e.g., UR5), the conventional wrist-partitioned architecture is still widely adopted (e.g., Kuka LWR, Neura Maira, Dyna Arm) (*25*). Hence, studying wrist-partitioned non-cuspidal cases provides the essential foundation for developing certified, singularity-aware control policies, as it guarantees a clear separation of feasible regions and enables rigorous analytical treatment. In contrast, cuspidal manipulators permit continuous transitions between distinct IKS regions without crossing a singularity, which complicates connectivity analysis and can lead to nonsingular solution transitions (*22, 35*). Our focus was therefore on non-cuspidal architectures to establish provable safety and predictability before extending these principles to the more intricate cuspidal cases.

Extending the framework to cuspidal architectures is an important next step. Recent results show that all $3R$ manipulators admit *reduced aspects* (*36*), which provide an intrinsic subdivision of joint

space even in the presence of multiple IKS in an aspect. We envision leveraging these reduced aspects, together with the different types of admissible paths in cuspidal robots (*21*), to generalise our connectivity checks and track-cycle strategies to such architectures. This is particularly relevant for newer cuspidal designs such as ABB GoFa (*37*) and Fanuc CRX, where kinematic awareness is crucial for safe skill transfer. Explicit planning over redundant dimensions and smoother blending between nominal and corrective velocities could further improve path feasibility, collision avoidance, and motion fluidity near constraints. In our current implementation, the violation-check step monitors proximity to joint limits, singularities, and voids in workspace and triggers the corresponding track-cycle behaviour; in principle, self- and scene-collisions could be handled in exactly the same way by treating collision manifolds as additional joint-space boundaries and updating the track-cycle logic accordingly. Developing and validating this collision-aware, cuspidal-ready extension remains an important avenue for future work.

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

MATERIALS AND METHODS

Overview

This research aimed at developing a framework to transfer the demonstrated behaviour across a class of $3R$ robots and redundant $4R$ robots. The framework first models the user behaviour as is, agnostic to the robot and autonomously embeds the behaviour model with robot-specific kinematic intelligence, resulting in a control policy overseeing the safe and violation-free execution on a non-redundant robot. A modular arrangement of such non-redundant robot control policies, when composed together, enables behaviour execution on redundant robots.

Behaviour modelling from demonstration

We begin by modelling the user-demonstrated behaviour in both the workspace $\mathcal{X}$ and joint space $\mathcal{Q}$ of the robot (see Figure 7(A1), (B), and (C)). Each user demonstration (trajectories in $\mathcal{X}$) and its inverse-mapped versions in $\mathcal{Q}$, are individually embedded as globally asymptotically stable dynamical systems (DSs): $\dot{\mathbf{x}} = \mathbf{g}(\mathbf{x})$ for workspace trajectory, and $\dot{\mathbf{q}} = \mathbf{g}(\mathbf{q})$ for joint trajectory. The DS formulation ensures that a trajectory from any starting point in the workspace or joint space converges to the demonstrated goal over time. Each DS is constructed in a *latent space* $\mathcal{U}$, a transformed coordinate system obtained via a bijective mapping $\psi : \mathcal{Z} \rightarrow \mathcal{U}$, where $\mathcal{Z}$ is either $\mathcal{X}$ or $\mathcal{Q}$. In latent space, the dynamics become (quasi)linear, simplifying stability guarantees. The learned dynamics in $\mathcal{U}$ are then pulled back to the original space via inverse mapping, ψ^{-1}, producing non-linear but stable DSs in both spaces - $\mathcal{X}$ and $\mathcal{Q}$

To generalize across multiple demonstrations, all joint space trajectories that terminate in the same *aspect* $\mathcal{A}_i$ are grouped together (see Figure 7(C)). Each resulting DS embedding $\mathbf{g}_j^{(Q)}$ within this aspect is associated with a localized *region of influence* in aspect $\mathcal{A}_i$ (see Figure 7(C)), defined as the set of joint configurations where $\mathbf{g}_j^{(Q)}$ governs the motion. That is, for any $\mathbf{q}$ in this region, the robot's behaviour is given by $\dot{\mathbf{q}} = \mathbf{g}_j^{(Q)}(\mathbf{q})$. An analogous partitioning is performed in the workspace when multiple demonstrations are available. The workspace partitioning spans the entire domain, in contrast to the joint space, where the partitioning is aspect-specific.

These regions are defined via *cones of influence* (see Figure 7(D1), and (D2)) in the latent space:

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

Here, $\overset{*}{\mathbf{u}} = \mathbf{1}_n^T$ is the latent embedding of the goal, and $\mathbf{v}_j$ is the vector from the start to goal. The angle λ_j controls the size of the neighbourhood influenced by the DS.

At inference time, the control law is given by:

$$\dot{\mathbf{q}} = \mathbf{g}_j^{(Q)}(\mathbf{q}), \quad \text{where } j = \arg\max \left\{ P(\mathcal{R}_j) \,\middle|\, \mathbf{q} \in \mathcal{R}_j \right\}$$

Priority $P(\mathcal{R}_j)$ ensures disambiguation when regions overlap.

To prevent high-priority demonstrations from overriding lower-priority behaviours, the opening angle λ_j is adjusted using projection checks:

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

By combining such aspect-specific behaviours, we construct the full joint space policy. The process of modelling the joint space policy from multiple workspace demonstrations for the case in Figure 7 is presented in Movie S5.

For full derivations and algorithmic procedures, see Supplementary Methods: *Modelling the Robot Control Policy*.

Embedding kinematic intelligence

To generalise user-demonstrated behaviours and ensure safe execution, we embed kinematic intelligence into the robot's control policy. So far, we have modelled the behaviour in both workspace and joint space. While either model can act as a control policy, directly embedding constraints in the workspace model is challenging due to complex singularities (e.g., nodes, cusps) and non-linear boundaries formed by joint limits. Moreover, the workspace lacks 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 while preserving user behaviour. We thus adopt the joint space model as the primary control policy, enriched via: (1) singularity-based robot categorisation and (2) incremental generalisation that transfers behavioural variability from workspace to joint space.

We introduce a novel, exhaustive categorisation of generic and non-generic non-cuspidal $3R$ robots into six categories. This reveals how motion becomes constrained, enabling targeted strategies for stable, feasible behaviour around constraints. We analyse the topology of joint space singularities, classifying them by loop structure and turning points to design locally stable, constraint-aware strategies.

Categories differ by whether branches loop, intersect, or fold, dictating directional constraints. Further details are in Supplementary Methods: *Robot Categorisation*.

If the joint space trajectory risks violating limits or singularities, we activate a near-boundary control policy to keep motion within the same aspect, bounded by joint limits and singularities, tailored per robot based on its singularity layout.

We define safety margins near aspect boundaries and monitor motion continuously. If a violation

is imminent, we switch to the near-boundary policy, which temporarily deviates along constraint-safe paths, termed the track cycle, before resuming nominal trajectory. Near-constraint strategies for all six categories are in Supplementary Methods: *Behaviour near Kinematic Constraints*. The framework integrates these strategies with deterministic connectivity checks of fixed computational complexity, ensuring stable and safe motion near constraints. These checks verify whether the configuration lies within a feasible aspect and can reach the goal without violating it; otherwise, execution halts, preventing unsafe behaviour. Details for all singularities in non-cuspidal $3R$ robots are in Supplementary Methods: *Reliable Execution*.

Embedding kinematic intelligence keeps 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 jointspace transfer for enhanced generalisation

To improve generalization of the joint-space control policy, we introduce an incremental *workspace-to-joint space transfer* mechanism. This begins by uniformly sampling the joint space $\mathcal{Q}$ to generate a grid of candidate joint configurations $\{\mathbf{Q}\}^{\text{sample}}$, which are 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* $\mathbf{X}_i^{\text{transfer}}$ is generated using the workspace DS model $\mathrm{B}^{(\mathcal{X})}$ such that:

$$\mathbf{X}_i^{\text{transfer}} = \left\{ \mathbf{x}_t \;\middle|\; \mathbf{x}_{t=0} = \mathbf{x}_i,\ \lim_{t\to\infty} \mathbf{x}_t = \overset{*}{\mathbf{x}} \right\}$$

These transfer trajectories are then inverted back into joint space, yielding a set $\{\mathbf{Q}\}^{\text{transfer}}$ of candidate joint-space trajectories. Each trajectory is validated and trimmed to remove unsafe or constraint-violating segments using certified constraint checks. To ensure efficient expansion of the joint space model $\mathrm{B}^{(\mathcal{Q})}$, we introduce a *novelty metric* that prioritizes trajectories contributing the most new information. The novelty of a trajectory is evaluated by computing its angular deviation in the latent space from existing demonstrations. Let $\psi_j : \mathcal{Q} \to \mathcal{U}$ denote the embedding function of the j^{th} learned DS, and let $\mathbf{q}_t$ be a point on a transfer trajectory $\mathbf{Q}^{\text{transfer}}$. The angular deviation is given by:

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

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

$$\text{novelty}(\mathbf{Q}^{\text{transfer}}) = \min_j \left\{ \Theta_j(\mathbf{Q}^{\text{transfer}}) \;\middle|\; \psi_j \in \mathrm{B}^{(\mathcal{A}_i)} \right\}$$

The most novel trajectory is selected as:

$$\mathbf{Q}^{\text{transfer}}_{\text{novel}} = \underset{\mathbf{Q}_i^{\text{transfer}} \in \{\mathbf{Q}\}^{\text{transfer}}}{\arg\max} \; \text{novelty}(\mathbf{Q}_i^{\text{transfer}})$$

This most novel trajectory is then used to update the aspect-specific model $\mathrm{B}^{(\mathcal{A}_i)}$, and by extension, the overall joint space behaviour $\mathrm{B}^{(Q)}$. This ensures that the control policy becomes 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 Methods section titled *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, while advantageous for optimising secondary criteria (e.g., obstacle avoidance, joint limits, or ergonomics), poses significant challenges for characterising and generalising robot behaviour. By leveraging Kinematic Intelligence, we address this by parameterising the redundant joint angle, denoted q_R, and reducing the complete $7R$ manipulator to a continuum of simpler, equivalent $3R$ chains, similar to one reported in (*25*). Each value of q_R yields a distinct robot architecture with well-defined kinematic properties, particularly concerning singularities and solution multiplicity.

This dimensionality reduction enables a structured encoding of robot behaviour as the redundant angle serves as an intrinsic coordinate along which motion solutions can be organised. For instance,

motion trajectories executed at different q_R values correspond to distinct internal configurations achieving the same external task. These configurations can be interpreted as behavioural modes, with q_R acting as a modulating variable. In non-cuspidal robots, where inverse kinematic solutions remain separated by singularities for each q_R, this parameterisation ensures that the operation mode is consistent and predictable.

This modular arrangement of behaviours allows redundancy resolution and behaviour generalisation across platforms. Crucially, it offers a modular, explainable framework for managing the solution null space of redundant manipulators, integrating geometric insight with practical applicability in control and learning. For applications demanding safety, repeatability, or human interaction, such an interpretable structure is essential for ensuring trustworthy robot behaviour. A detailed discussion of why classical numerical null-space strategies are insufficient in this context is provided in the Supplementary Materials, section *Issues with numerical path-planning strategies*.

Supplementary materials

Materials and Methods

Figs. S1 to S24

Tables S1 to S3

Movies S1 to S6

References and Notes

1. H. Ravichandar, A. S. Polydoros, S. Chernova, A. Billard, Recent Advances in Robot Learning from Demonstration. *Annual Review of Control, Robotics, and Autonomous Systems* **3** (1), 297–330 (2020), number: 1, doi:10.1146/annurev-control-100819-063206, https://www.annualreviews.org/doi/10.1146/annurev-control-100819-063206.

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)* (2008), pp. 33–40.

3. K. Kronander, A. Billard, Learning Compliant Manipulation through Kinesthetic and Tactile Human-Robot Interaction. *IEEE Transactions on Haptics* **7** (3), 367–380 (2014), doi:10.1109/TOH.2013.54.

4. A. L. P. Ureche, K. Umezawa, Y. Nakamura, A. Billard, Task Parameterization Using Continuous Constraints Extracted From Human Demonstrations. *IEEE Transactions on Robotics* **31** (6), 1458–1471 (2015), doi:10.1109/TRO.2015.2495003, http://ieeexplore.ieee.org/document/7339616/.

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)* (2017), pp. 60–65, doi:10.1109/WHC.2017.7989877.

6. N. Jaquier, *et al.*, Transfer learning in robotics: An upcoming breakthrough?: A review of promises and challenges. *The International Journal of Robotics Research* p. 02783649241273565 (2024), doi:10.1177/02783649241273565, https://doi.org/10.1177/02783649241273565.

7. B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration. *Robotics and Autonomous Systems* **57** (5), 469–483 (2009), number: 5, doi:10.1016/j.robot.2008.10.024, https://linkinghub.elsevier.com/retrieve/pii/S0921889008001772.

8. S. Calinon, F. Guenter, A. Billard, On Learning, Representing, and Generalizing a Task in a Humanoid Robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **37** (2), 286–298 (2007), doi:10.1109/TSMCB.2006.886952.

9. A. Shaw, J. Lee, J. Park, Constrained dynamic movement primitives for collision avoidance in novel environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2023).

10. L. Yang, M. Fischer, D. Kragic, Enhancing Learning from Demonstration with DLS-IK and ProMPs, in *International Conference on Automation and Computing (ICAC)* (2024).

11. E. Pignat, S. Calinon, Learning from demonstration using products of experts: Applications to manipulation and task prioritization. *The International Journal of Robotics Research* **41** (4), 373–394 (2022).

12. L. Bakker, V. Koltun, M. Toussaint, TamedPUMA: Safe and Stable Imitation Learning with Geometric Fabrics, in *Learning for Dynamics and Control (L4DC)* (2025).

13. B. Trabucco, M. Phielipp, G. Berseth, AnyMorph: Learning Transferable Polices By Inferring Agent Morphology, in *Proceedings of the 39th International Conference on Machine Learning*, K. Chaudhuri, *et al.*, Eds. (PMLR), vol. 162 of *Proceedings of Machine Learning Research* (2022), pp. 21677–21691.

14. C. Sferrazza, D.-M. Huang, F. Liu, J. Lee, P. Abbeel, Body Transformer: Leveraging Robot Embodiment for Policy Learning, in *8th Annual Conference on Robot Learning* (2024), `https://openreview.net/forum?id=0ce2215aJE`.

15. 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)* (2022), pp. 5210–5217, doi:10.1109/IROS47612.2022.9982127.

16. 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)* (2015), pp. 1075–1082, doi:10.1109/HUMANOIDS.2015.7363502.

17. 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, Kobe) (2009), pp. 763–768, doi:10.1109/ROBOT.2009.5152385, `http://ieeexplore.ieee.org/document/5152385/`.

18. S. Schaal, J. Peters, J. Nakanishi, A. Ijspeert, Learning Movement Primitives, in *Robotics Research. The Eleventh International Symposium*, P. Dario, R. Chatila, Eds. (Springer Berlin Heidelberg, Berlin, Heidelberg) (2005), pp. 561–572.

19. 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)* (2016), pp. 351–358, doi:10.1109/HUMANOIDS.2016.7803300.

20. D. H. Salunkhe, D. Chablat, P. Wenger, Trajectory planning issues in cuspidal commercial robots, in *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), pp. 7426–7432, doi:10.1109/ICRA48891.2023.10161444.

21. D. H. Salunkhe, T. Marauli, A. Müller, D. Chablat, P. Wenger, Kinematic issues in 6R cuspidal robots, guidelines for path planning and deciding cuspidality. *The International Journal of Robotics Research* p. 02783649241293481 (2024), doi:10.1177/02783649241293481.

22. P. Wenger, J. El Omri, Changing posture for cuspidal robot manipulators, in *Proceedings of IEEE International Conference on Robotics and Automation* (IEEE, Minneapolis, MN, USA), vol. 4 (1996), pp. 3173–3178.

23. P. Wenger, Design of cuspidal and non-cuspidal robot manipulators, in *Proceedings of International Conference on Robotics and Automation*, vol. 3 (1997), doi:10.1109/ROBOT.1997.619284.

24. P. Wenger, Uniqueness domains and regions of feasible paths for cuspidal manipulators. *IEEE Transactions on Robotics* **20** (4), 745–750 (2004), doi:10.1109/TRO.2004.829467.

25. D. H. Salunkhe, S. Gupta, A. Billard, Cuspidal Redundant Robots: Classification of Infinitely Many IKS of Special Classes of 7R Robots. *IEEE Robotics and Automation Letters* **10** (12), 12509–12516 (2025), doi:10.1109/LRA.2025.3623011.

26. J. W. Burdick, A classification of 3R regional manipulator singularities and geometries. *Mechanism and Machine Theory* **30** (1), 71–89 (1995), doi:https://doi.org/10.1016/0094-114X(94)00043-K, https://www.sciencedirect.com/science/article/pii/0094114X9400043K.

27. P. Borrel, A. Liegeois, A study of multiple manipulator inverse kinematic solutions with applications to trajectory planning and workspace determination, in *Proceedings. 1986 IEEE International Conference on Robotics and Automation* (Institute of Electrical and Electronics Engineers, San Francisco, CA, USA), vol. 3 (1986).

28. P. Wenger, Classification of 3R Positioning Manipulators. *Journal of Mechanical Design* **120** (2), 327–332 (1998), doi:10.1115/1.2826976.

29. D. Paganelli, *Topological Analysis of Singularity Loci for Serial and Parallel Manipulators*, Ph.D. thesis, Universita di Bologna, Bologna, Italy (2008).

30. M. Asgari, I. Bonev, C. Gosselin, Singularities of ABB's YuMi 7-DOF robot arm. *Mechanism and Machine Theory* **205** (2025), doi:https://doi.org/10.1016/j.mechmachtheory.2024.105884.

31. A. J. Elias, J. T. Wen, Redundancy parameterization and inverse kinematics of 7-DOF revolute manipulators. *Mechanism and Machine Theory* **204**, 105824 (2024), doi:https://doi.org/10.1016/j.mechmachtheory.2024.105824.

32. S. Gupta, A. Nayak, A. Billard, [Under Review]Compact Oneshot Modelling of High Dimensional Demonstrations using Laplacian Eigenmaps. *IEEE Transactions on Robotics* **40** (2024).

33. 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 Press) (2013), p. 301–308, doi:10.1109/HRI.2013.6483603.

34. J. Geldenbott, K. Leung, Legible and Proactive Robot Planning for Prosocial Human-Robot Interactions, in *2024 IEEE International Conference on Robotics and Automation (ICRA)* (2024), pp. 13397–13403, doi:10.1109/ICRA57147.2024.10611294.

35. P. Wenger, D. Chablat, A Review of Cuspidal Serial and Parallel Manipulators. *Journal of Mechanisms and Robotics* **15** (4), 040801 (2022), doi:10.1115/1.4055677.

36. D. H. Salunkhe, C. Spartalis, J. Capco, D. Chablat, P. Wenger, Necessary and sufficient condition for a generic 3R serial manipulator to be cuspidal. *Mechanism and Machine Theory* **171**, 104729 (2022), doi:10.1016/j.mechmachtheory.2022.104729.

37. A. J. Elias, J. T. Wen, Path Planning and Optimization for Cuspidal 6R Manipulators. *arxiv* (2025), https://arxiv.org/abs/2501.18505.

38. J. Denavit, R. S. Hartenberg, A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics* **22** (2), 215–221 (1955), doi:10.1115/1.4011045.

39. D. Kohli, J. Spanos, Workspace Analysis of Mechanical Manipulators Using Polynomial Discriminants. *Journal of Mechanisms, Transmissions, and Automation in Design* **107** (2), 209–215 (1985), doi:10.1115/1.3258710.

40. J. Burdick, A classification of 3R regional manipulator singularities and geometries, in *Proceedings. 1991 IEEE International Conference on Robotics and Automation* (1991), pp. 2670–2675 vol.3, doi:10.1109/ROBOT.1991.132033.

41. D. Pai, M. Leu, Genericity and singularities of robot manipulators. *IEEE Transactions on Robotics and Automation* **8** (5), 545–559 (1992).

42. J. W. Burdick, A classification of 3R regional manipulator singularities and geometries. *Mechanism and Machine Theory* **30** (1), 71–89 (1995), doi:https://doi.org/10.1016/0094-114X(94)00043-K, https://www.sciencedirect.com/science/article/pii/0094114X9400043K.

43. J. El Omri, P. Wenger, How to recognize simply a non-singular posture changing 3-DOF manipulator, in *Proc. 7th Int. Conf. on Advanced Robotics* (1995), pp. 215–222.

44. D. L. Pieper, *The Kinematics of Manipulators Under Computer Control*, Ph.D. thesis, Stanford University, USA (1968).

45. D. H. Salunkhe, *Robots cuspidaux : étude théorique, classification et applications aux robots commerciaux*, Ph.D. thesis, Ecole Centrale de Nantes, France (2023).

46. D. Chablat, R. Prébet, M. Safey El Din, D. H. Salunkhe, P. Wenger, Deciding Cuspidality of Manipulators through Computer Algebra and Algorithms in Real Algebraic Geometry, in *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, ISSAC '22 (Association for Computing Machinery, Villeneuve-d'Ascq, France) (2022), p. 439–448, doi:10.1145/3476446.3535477.

47. K. Makarychev, Y. Makarychev, Certified Algorithms: Worst-Case Analysis and Beyond, in *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, T. Vidick, Ed. (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany), vol. 151 of *Leibniz International Proceedings in Informatics (LIPIcs)* (2020), pp. 49:1–49:14, doi:10.4230/LIPIcs.ITCS.2020.49.

48. S. M. Khansari-Zadeh, A. Billard, Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. *IEEE Transactions on Robotics* **27** (5), 943–957 (2011), number: 5, doi:10.1109/TRO.2011.2159412, `http://ieeexplore.ieee.org/document/5953529/`.

49. S. Salvador, P. Chan, FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space p. 11.

Acknowledgments

We thank *Dimitri Jacquemont* for their invaluable assistance in setting up and executing the robot experiments, and *Natacha Jeannot* for generously providing the demonstration data used in this work. We are also grateful to *Isabelle 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 EU project DARKO under Grant H2020 ICT-46-2020, and Horizon Europe under Grant 101070596, euROBIN.

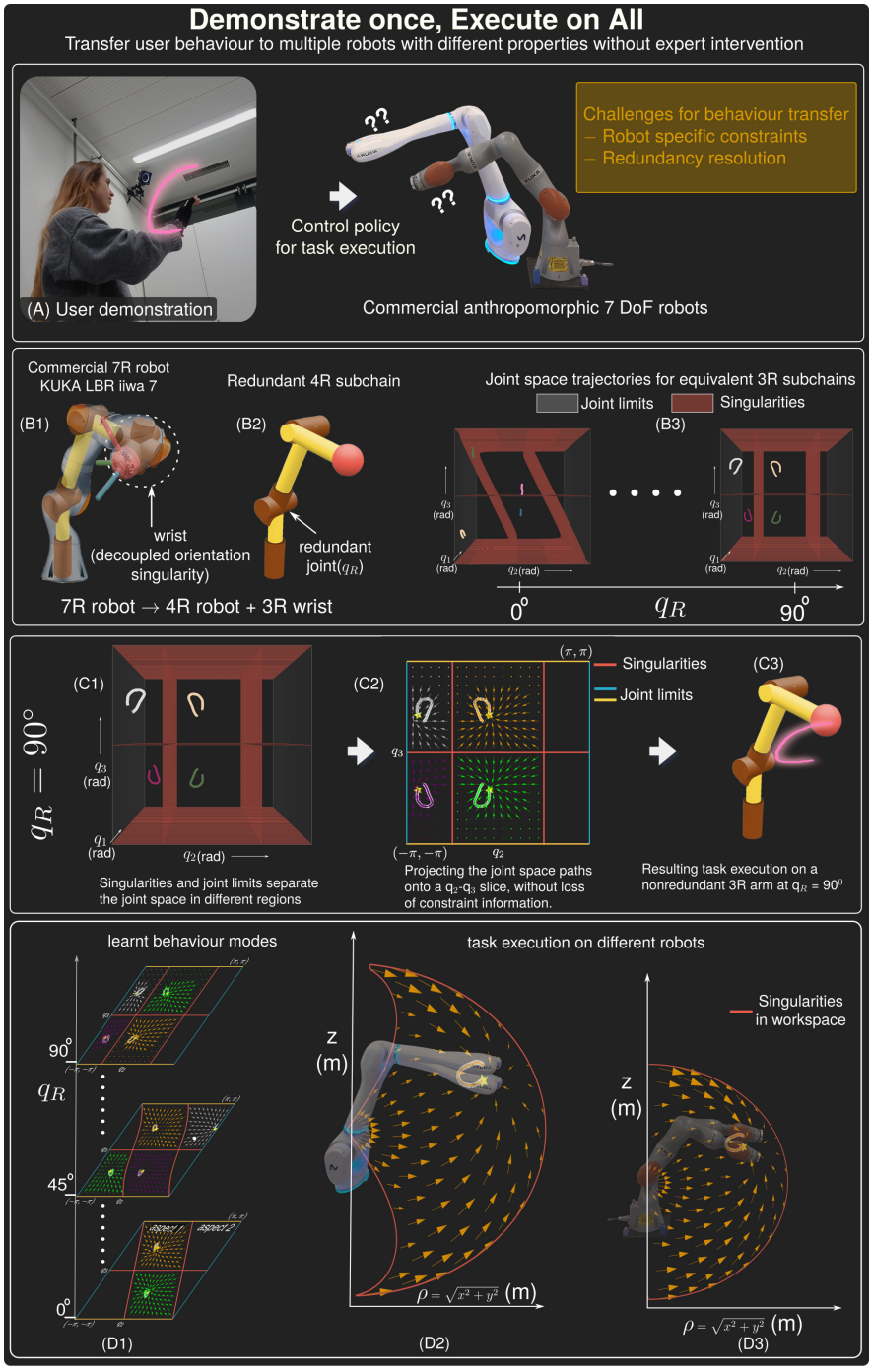


Figure 1: **Overview of kinematic-aware transfer of demonstrated behaviour to multiple robots** A user demonstrate a task (A) - challenge resides in embedding the control policy for easy transfer across multiple robots. We consider 7-DoF robots with wrist-partitioned architecture (B1), simplifying kinematics and reducing transfer to controlling the redundant 4-DoF subchain (B2). Redundancy, parameterized by the redundant joint angle q_R, identifies kinematic equivalence across 3-DoF non-redundant ($3R$) robots. (B3) Visualization of kinematic constraints (singularities and joint limits) for a specific kinematic, dividing joint space into four disjoint regions (C1). We learn a global control policy that generates stable subspace dynamics (C2), enabling violation-free task execution on a $3R$ robot (C3). Transfer across kinematics is achieved via policies parameterized by q_R (D1), with successful results on standard commercial robots (D2, D3).

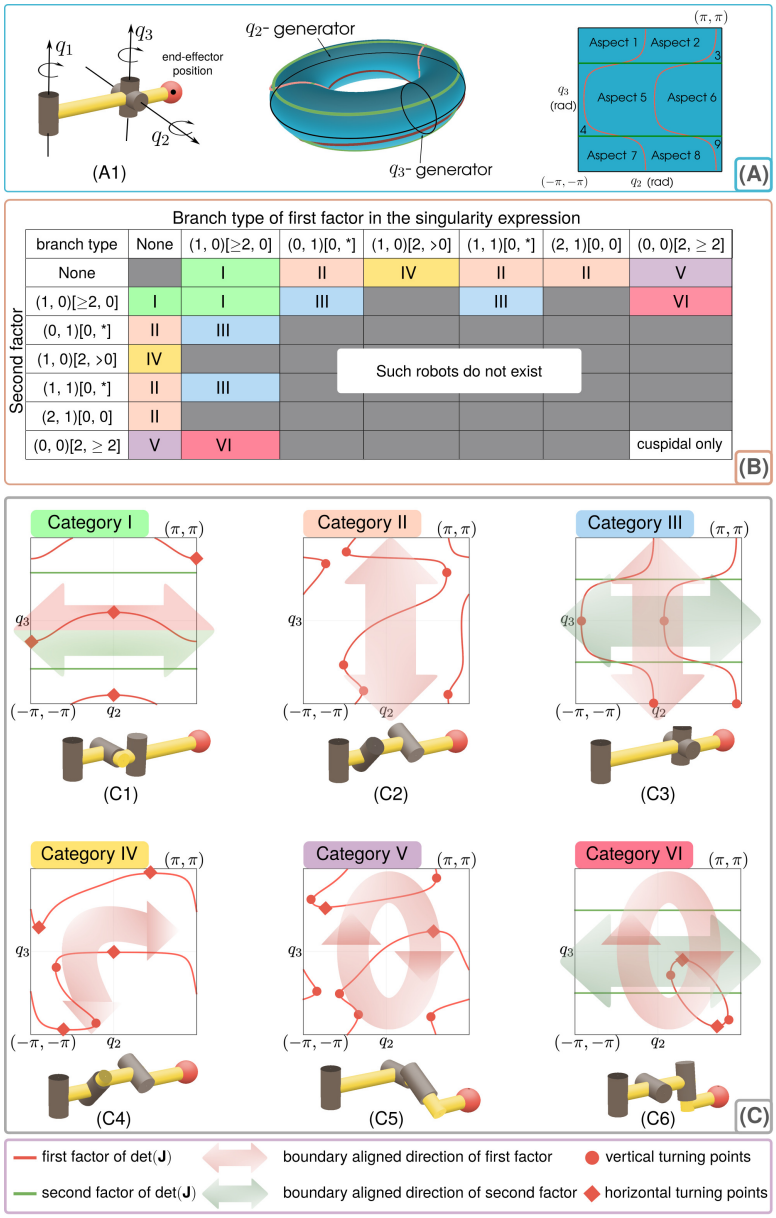


Figure 2: **Topological and differential properties of singularities as a basis for categorising 3R robots and defining boundary-aligned control actions.** To analyse the global properties of multiple robots, we study the different factors of the determinant of the Jacobian; their topological and differential properties give rise to six robot categories (B), and every non-cuspidal 3R robot falls into exactly one of these categories. Panels (C1)–(C6) show one representative robot per category, with its singularity curves and the associated *boundary-aligned directions* (arrows) drawn on the joint torus. Each set of arrows depicts the concrete control action used by the track-cycle controller when the state enters the safety band around that constraint: in Category I (C1), the track cycle traverses *horizontally* along the boundary; in Category II (C2), it traverses *vertically*; in Category III (C3), it switches between horizontal and vertical traversal depending on which singular branch is being violated; in Category IV (C4), it follows a primarily horizontal traversal but with additional turning-point considerations due to the more intricate branch geometry; in Category V (C5), the track cycle moves *clockwise or counterclockwise* around the torus along closed singular loops; and in Category VI (C6), it combines clockwise traversal around loops with horizontal drift along extended branches. In all cases, the nominal DS velocity is projected onto the depicted tangent direction so that motion remains within the same aspect and is steered along the boundary towards regions where re-entry into the interior is kinematically feasible.

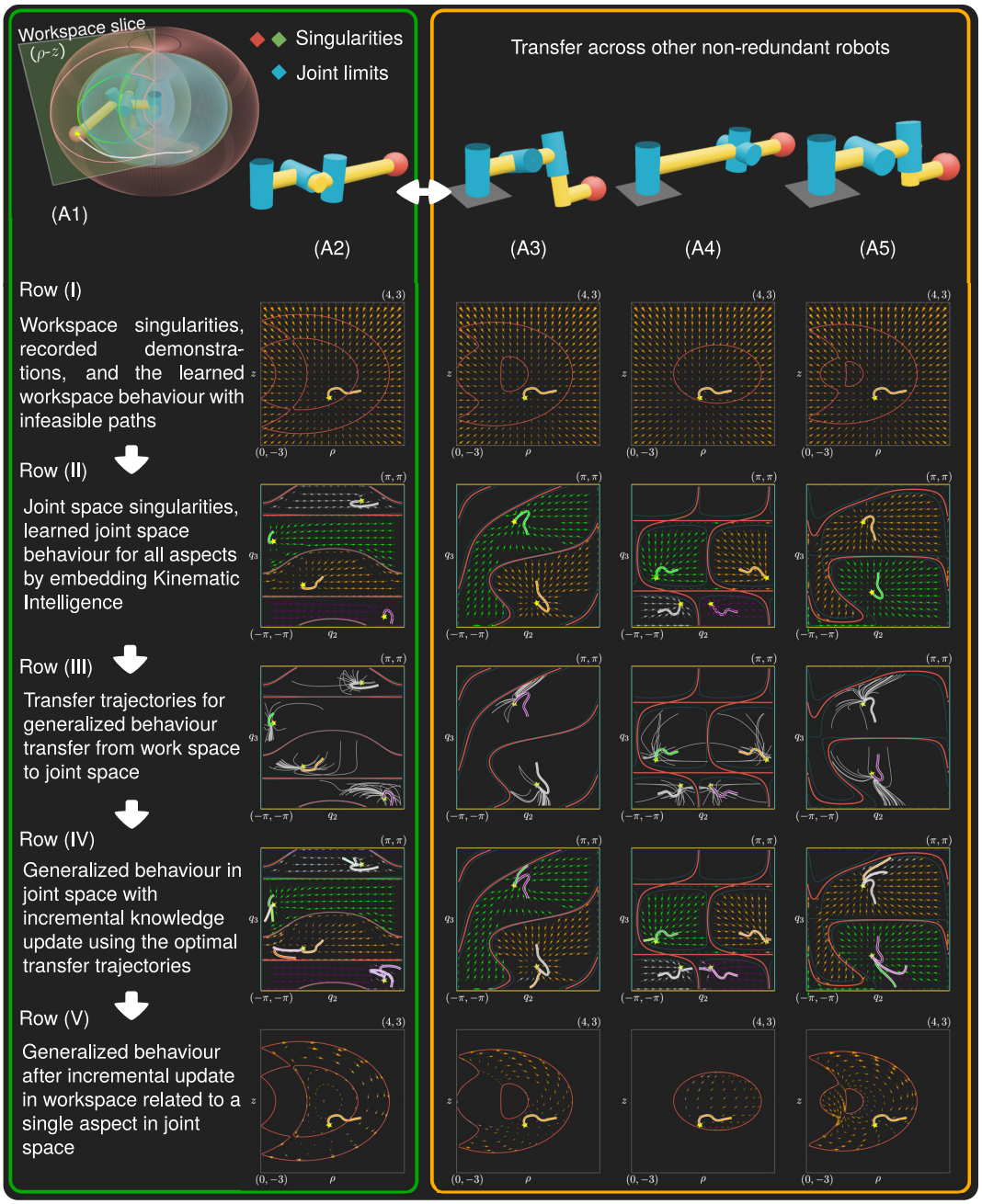


Figure 3: **Transfer of demonstrated behaviour to multiple non-redundant robots.** (A1) presents the Cartesian workspace of the $3R$ robot from (A2). The first column lists the steps from the learning framework, and each row reflects the status of these steps for four different robots - one each from Categories I, II, III, and IV - in columns (A2) through (A5), respectively. To visualise the user trajectory in the robot's workspace, we project it onto a $\rho - z$ slice of the workspace (A1). Row (I) depicts the different workspace singularities in the $\rho - z$ slice for the robots. Row (II) depicts the learnt control policy for every feasible aspect in the joint space of each robot. Row (III) depicts the set of transfer trajectories, used for incremental update of control policies. Row (IV) depicts the most novel transfer trajectory in each aspect and the updated behaviour. Row (V) depicts the resulting constraint-compliant workspace behaviour in a $\rho - z$ slice for each robot.

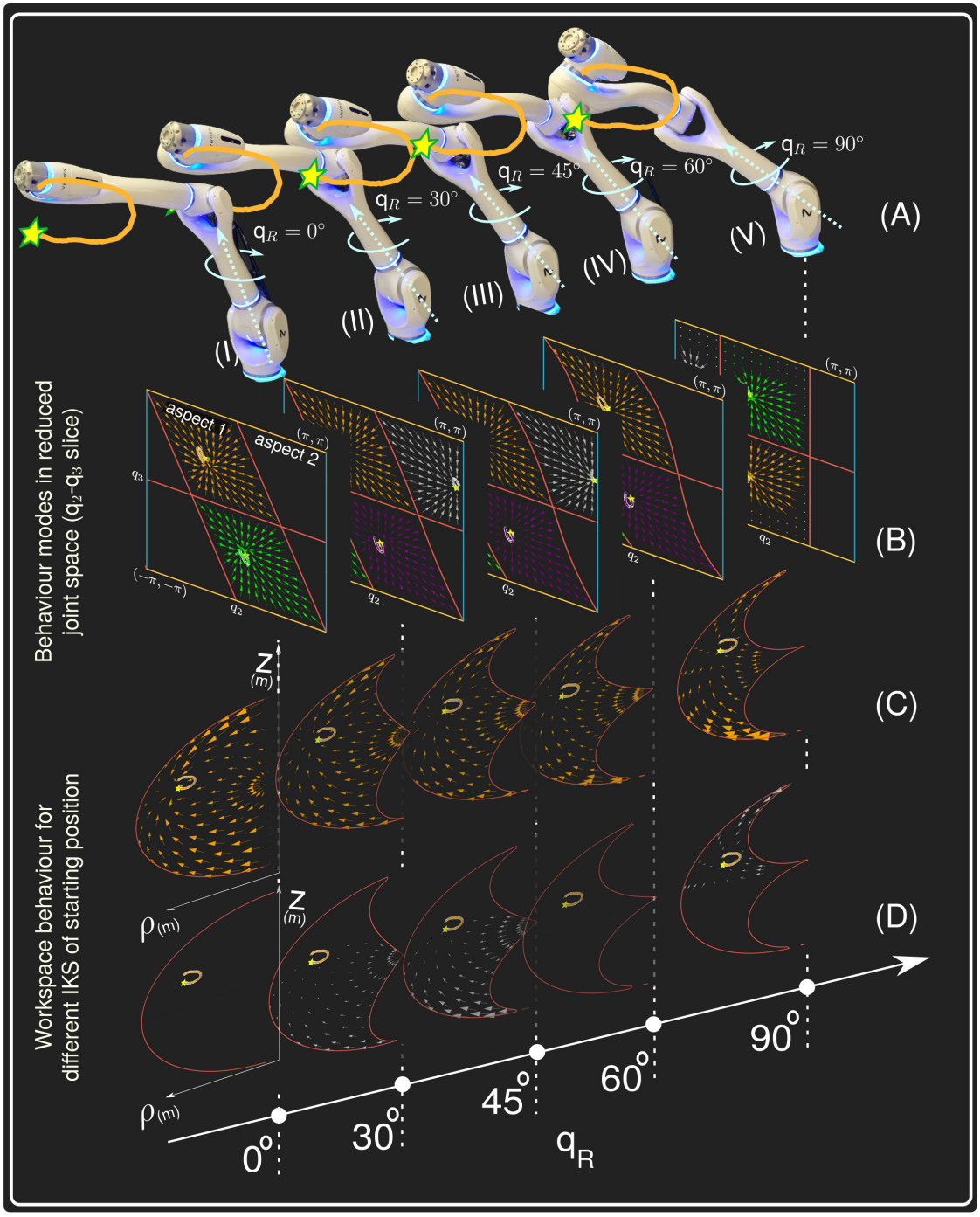
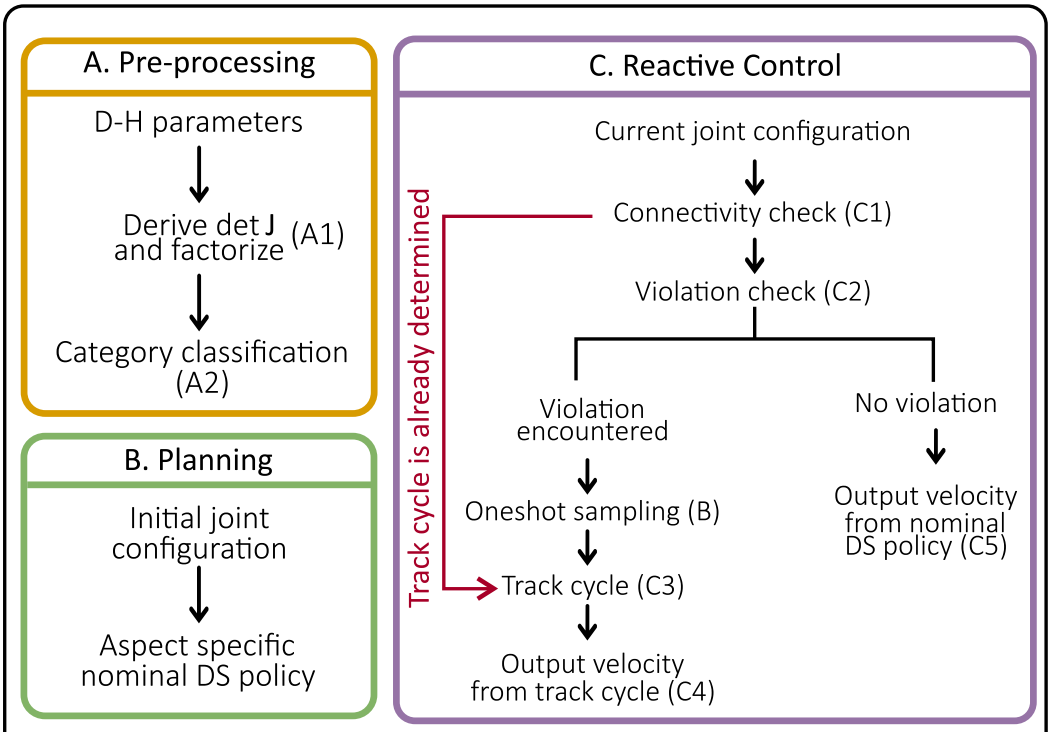


Figure 4: **Behaviour transfer to redundant robots through redundancy parametrisation** The behaviour of the redundant $4R$ subchcain of a wrist-removed 7 DoF anthropomorphic robot can be characterised by discretising the values of the redundant angle q_R, which in this case is the third joint. Five values are picked for q_R - 0°, 30°, 45°, 60°, and 90° (A). Control policies for the equivalent non-redundant $3R$ subchains at these q_R values in the reduced $q_2 - q_3$ slice of the joint space are presented in (B). (C) presents a $\rho - z$ slice of the feasible workspace at each q_R value, where the robot (from (A)) is in a specific aspect of the joint space (orange). (D) presents the resulting workspace behaviour when the robot has a joint configuration that is a different solution of IK (silver).



Task	Mean (μ)	Std. Dev. (σ)	95th Pctl.	Min	Max
A. Pre-processing					
Derive factors of det J (A1)	389.1 ms	779.1 ms	625.0 ms	0.1 ms	23930.0 ms
Category classification (A2)	253.3 μs	744.4 μs	1143.8 μs	5.9 μs	70637.4 μs
B. Planning					
Aspect specific behaviour*	15.5 ms	21.3 ms	67.2 ms	1.5 ms	84.1 ms
C. Reactive Control					
C1. Connectivity checks	434.8 μs	7125.0 μs	562.4 μs	57.6 μs	679353.8 μs
C2. Violation check	80.7 μs	38.1 μs	100.0 μs	19.8 μs	3525.0 μs
C3. Track-cycle calculation	701.5 μs	15320.2 μs	966.7 μs	1.0 μs	1980828.8 μs
C4. Track-cycle velocity output	345.4 μs	640.3 μs	585.1 μs	1.0 μs	64676.5 μs
C5. Nominal DS policy velocity output*	15.5 ms	21.3 ms	67.2 ms	1.5 ms	84.1 ms

$*$: Data is from a smaller subset of 1000 robots.
All other data is from $\sim 22,000$ robots.

Figure 5: **End-to-end DS-based policy execution on a single robot.** The figure illustrates the three main components of our framework for executing a learned dynamical system (DS) policy on a single robot and reports the wall-clock timings for each step of the process. (A) *Pre-processing*: an offline, one-time step that computes the kinematic structure, singularity layout, reduced aspects, and other quantities required for constraint-aware execution. (B) *Planning*: given an initial joint configuration, the framework selects the corresponding reduced aspect and synthesizes an aspect-specific reference behaviour compatible with the DS policy. (C) *Reactive control*: an online loop that combines connectivity checks, constraint-violation monitoring, nominal DS policy execution, and track-cycle computations to steer the robot while remaining within feasible regions safely. The accompanying wall-clock timings summarize the computational cost of each component, highlighting that the one-time offline pre-processing is very fast, and the online planning and reactive control steps are compatible with real-time execution.

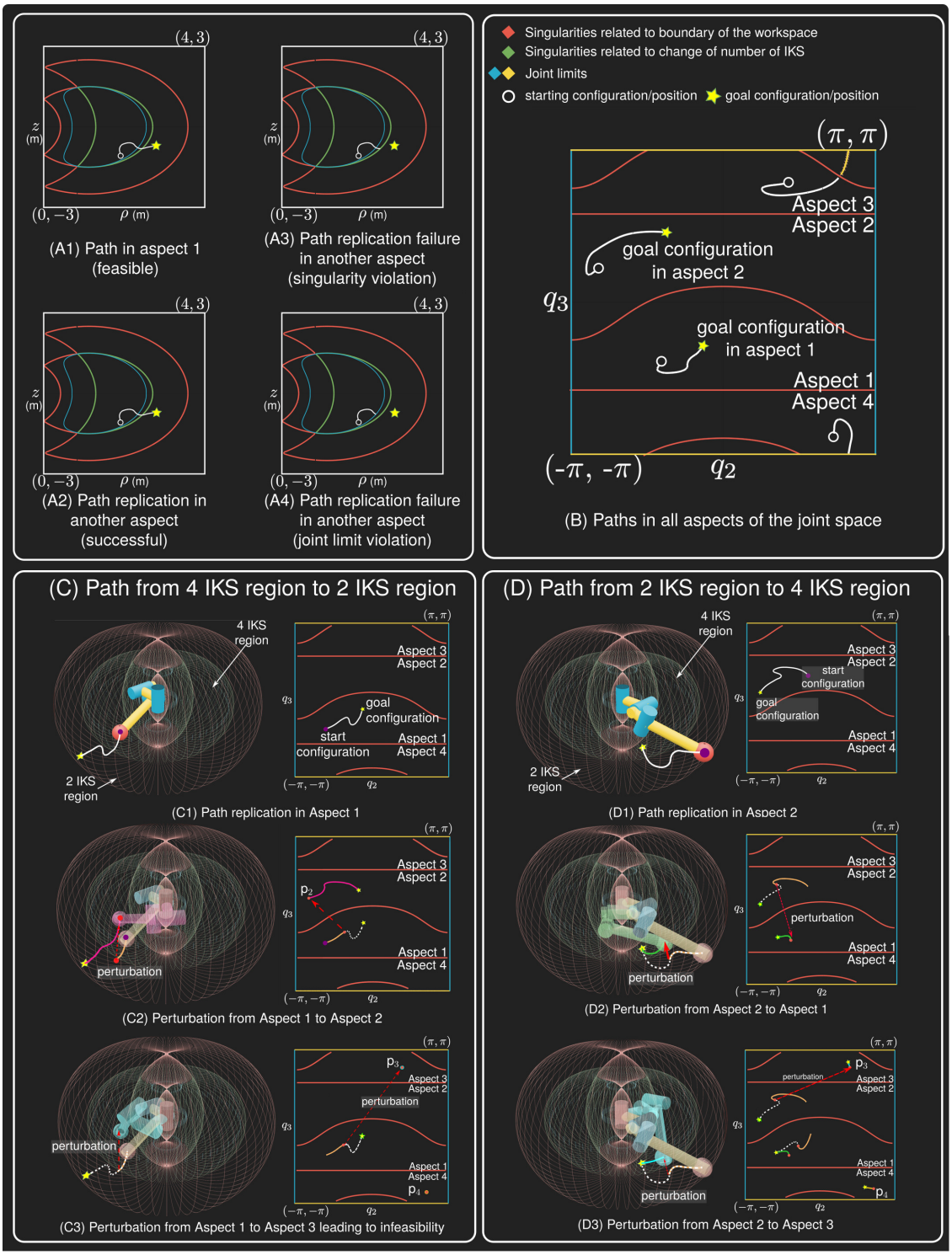


Figure 6: **Certified execution for non-redundant robots** The figure analyses the workspace demonstration provided in Figure 3 (A1) on a $3R$ robot from Category I. The starting position has four IKS, whereas the goal position has two. (A1 - A4) visualise the workspace trajectory corresponding to each aspect, in a $\rho - z$ slice of the workspace. Two of these trajectories - in Aspect 1 and Aspect 2 - reach the goal configuration without any violations (A1, A2). The trajectory in Aspect 3 encounters a singular configuration (A3), whereas that in Aspect 4 violates the third joint's limit (A4). (B) shows the joint space trajectories corresponding to the workspace demonstration, in each of the four aspects, and constraint violations encountered in Aspects 3 and 4. (C) presents the robot behaviour post learning the control policy and embedding kinematic intelligence. (C1) presents the replication when starting from Aspect 1. In (C2), the

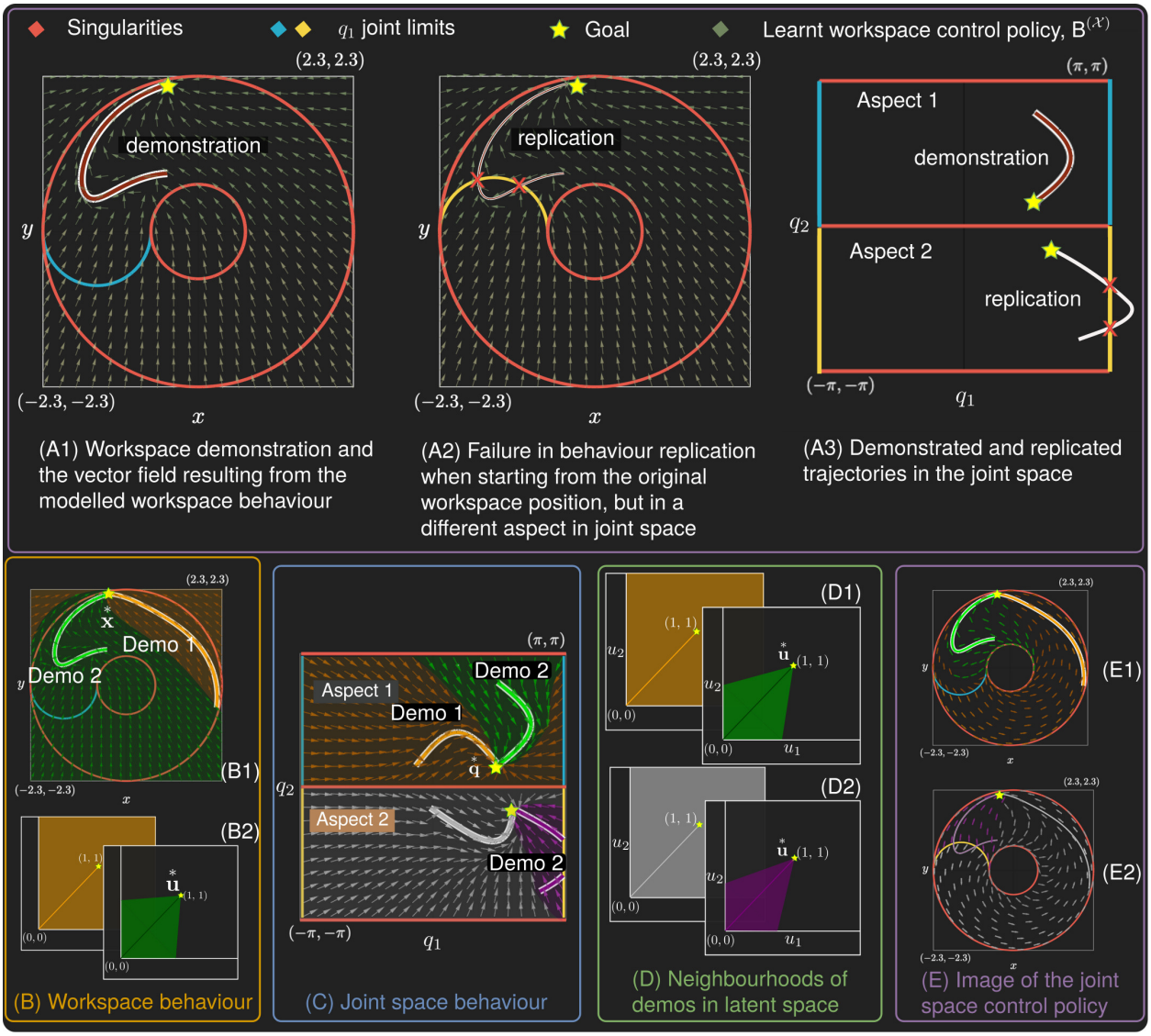


Figure 7: **An example of Learning and Replicating User-Intended Behaviour Across Workspace and Joint Space**. The top panel illustrates the modelling of a user-demonstrated trajectory in the workspace (A1) of a $2R$ robot, its corresponding joint space representation (A3), and the consequence of deploying the model from a different aspect, resulting in joint limit violation during replication (A2). The learned behaviour is encoded as a dynamical system (DS) in workspace and is aspect-agnostic unless explicitly constrained. The bottom panel demonstrates how multiple demonstrations are used to enrich the behaviour model. (B1) shows the DS embeddings and resulting workspace behaviour, while (B2) presents their latent space representations and cones of influence. (C) depicts the corresponding joint space trajectories, their trimming for constraint compliance, and the resulting aspect-specific joint space policies. (D1) and (D2) show the latent embeddings for joint space DSs in each aspect, with each DS's respective cone of influence. (E1) and (E2) display the resulting workspace behaviour when following the joint space control policy from initial configurations in Aspect 1 and Aspect 2, respectively.

Figure 8: **Experimental validation in a mock multi-robot automated assembly line:** (A) Overview of the mock assembly-line setup, consisting of a conveyor belt (A1), workbench (A2), tabletop (A3), and basket (A4). (B) Human demonstration of the task: pushing the wooden block from the conveyor belt onto the workbench (B1); placing it on the tabletop (B2); lifting it into a throwing pose (B3); and throwing it into the basket (B4). (C) Three robots — Neura Maira M (C1), KUKA LBR iiwa 7 (C2), and Duatic DynaArm (C3) — are deployed to reproduce the demonstrated behaviour. Manipulation skills are learned and transferred across these robots, enabling consistent execution of each stage regardless of the robot arrangement.

Author contributions: Conceptualization, simulation, investigation, analysis, visualization, and writing: S.G, D.H.S. Resources, supervision, writing, and revision: A.B.

Competing interests: There are no competing interests to declare.

Supplementary Materials for

Demonstrate Once, Execute on Many: Kinematic Intelligence for Cross-robot Skill Transfer

Sthithpragya Gupta[*†], Durgesh Salunkhe Scientist[†], Aude Billard

[*]Corresponding author. Email: sthithpragya.gupta@epfl.ch

[†]These authors contributed equally to this work.

This PDF file includes:

Materials and Methods

Figs. S1 to S24

Tables S1 to S3

Captions for Movies S1 to S6

Other Supplementary Materials for this manuscript:

Movies S1 to S6

Materials and Methods

Glossary

$\mathcal{Q}$ Joint space of the robot

$\mathcal{X}$ Workspace of the robot

EE-position End-effector position

$\mathbf{J}(\mathbf{q})$ Jacobian of the forward kinematic map at $\mathbf{q}$

$\det \mathbf{J}(\mathbf{q})$ Determinant of Jacobian at configuration $\mathbf{q}$

IKS Inverse kinematic solutions

$\mathcal{S}$ Singularities in joint space

t_i Rational parameterization of q_i, where $t_i = \tan \frac{q_i}{2}$

$\mathcal{B}$ Branch of a singularity, $\mathcal{B} \subset \mathcal{S}$

$\mathcal{U}$ Latent space that encodes the demonstration as a structured latent embedding

$\{\mathbf{X}\}$ Set of K expert demonstrations in the robot's workspace

$\{\mathbf{Q}\}$ Set of K_Q demonstrations in joint space

$\mathrm{B}^{(\mathcal{Q})}$ Behaviour learned in joint space

$B^{(\mathcal{A}_i)}$ Aspect-specific behaviour in joint space

$\mathrm{B}^{(\mathcal{X})}$ Behaviour learned in workspace

$\overset{*}{\mathbf{u}}$ Latent embedding of the attractor in latent space

$\overset{*}{\mathbf{x}}$ Demonstrated workspace goal position

$\{\overset{*}{\mathbf{q}}\}$ Attractors in $\mathcal{Q}$, i.e., IK solutions of $\overset{*}{\mathbf{x}}$

$\{A_i\}_{i=1}^{\mathcal{I}}$ Aspects forming a disjoint partition of $\mathcal{Q}$

Preliminaries

On the kinematics front, Section presents definitions relevant to the kinematics of serial robots, and their singularities. Section leverages these definitions and lays the foundation of homotopy-based classification of the robots on basis of their singularities. On the behaviour modelling front, Section presents the method to embed a demonstration as a stable control policy.

This paper uses original D-H parameters (*38*) to minimally represent a serial robot architecture. The subspace of $SE(3)$ formed by the reachable poses of a given nR robot is called the workspace and is denoted as workspace $\mathcal{X}$. The joint configuration of a robot is denoted as $\mathbf{q}$ and is a point in the joint space, joint space $\mathcal{Q}$, where $\mathbb{T}^n$ is an n-torus. Let $\mathbf{w}$ be the end-effector pose in $SE(3)$ corresponding to $\mathbf{q}$. The mapping between joint space $\mathcal{Q}$ and workspace $\mathcal{X}$, denoted by $\text{FK} : \mathcal{Q} \to \mathcal{X}$, defines the forward kinematics function $\text{FK} : \mathbf{q} \mapsto \mathbf{x}$.

Singularities in serial robots

The singularities $\mathcal{S}$ is the locus of critical points of FK in joint space $\mathcal{Q}$ that correspond to the configurations in the joint space where the Jacobian $\mathbf{J}(\mathbf{q})$ loses rank, i.e., when Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ is zero. The critical values are the images of the critical points in $\mathcal{W}$. (*39*) showed that the roots of the univariate polynomial used to solve the inverse kinematics of a 3R robot have a multiplicity greater than or equal to two. A similar conclusion can be drawn for nonredundant nR robots, as the robot either loses or gains multiple inverse kinematic solutions (IKS) upon crossing the locus of critical values.

Aspect: (*27*) defines an aspect as the largest singularity-free connected region in the joint space. Each aspect is bounded only by singularities in the joint space, and this definition does not incorporate the robot's joint limits. However, it has been noted that the kinematic properties of the robot are significantly affected by its joint limits (*35*). To facilitate real-world application of our framework, we extend the definition of aspect to be the largest singularity-free region in the joint space bounded by singularities, joint limits or a combination of both. The joint space is parameterized in q_2 and q_3 only as the singularities of the 3R robot are independent of q_1 (*40*). The pink region is bounded by only singularities, the green region by singularities and joint limits of q_2

only, the blue region by singularities and joint limits of q_3 only, and the yellow region is bounded by singularities as well as joint limits of both q_2 and q_3.

For a structured analysis of $3R$ robots, as proposed by (*41*), the properties of robot singularities can be leveraged to yield the following two categories:

Generic $3R$ *robot*: A robot whose singularities do not intersect.

Non-Generic robot: A $3R$ robot is nongeneric if it satisfies either of the following two conditions (*26*):

1. singularities $\mathcal{S}$ contains rank 1 critical points
2. singularities $\mathcal{S}$ contains rank 2 critical points, $\mathbf{q}_i^s$, satisfying:

$$\frac{\partial \det \mathbf{J}(\mathbf{q}^s)}{\partial \mathbf{q}_i^s} = 0, i = 2, 3$$

In this article, we will exclude the robots that satisfy the first condition of nongenericity as they have either reduced mobility or have singularities that map to an isolated critical value in the workspace (*42*).

Cusp: A cusp is a point in the locus of critical values of a 3R robot that satisfies the following conditions (*43*):

$$\begin{aligned} &M(t_3, R, z) = 0 \\ &\frac{\partial M}{\partial t_3}(t_3, R, z) = 0 \\ &\frac{\partial^2 M}{\partial t_3^2}(t_3, R, z) = 0 \\ &\frac{\partial^3 M}{\partial t_3^3}(t_3, R, z) \neq 0 \end{aligned} \tag{S1}$$

where, for an end effector position, $R = x^2 + y^2 + z^2$, and rational parameterization of q_i, first=$t_i = \tan\frac{q_i}{2}$, text=t_i. The function, $M(t_3, R, z)$, is a polynomial of degree four in t_3. This polynomial is obtained by eliminating the joint variables t_1 and t_2 from the forward kinematic function (refer to (*44*)).

Cuspidal robots: Robots whose joint space has at least one aspect with multiple IKS are defined as cuspidal robots (*43*).

Horizontal (resp. vertical) turning points: The joint configurations $\mathbf{q}$ that satisfy:

$$\begin{aligned} &\det \mathbf{J}(\mathbf{q}) = 0 \\ &\gamma \frac{\partial \det \mathbf{J}}{\partial q_2} = 0(resp.1), \text{ for } \gamma \neq 0 \\ &\gamma \frac{\partial \det \mathbf{J}}{\partial q_3} = 1(resp.0), \text{ for } \gamma \neq 0 \end{aligned} \tag{S2}$$

Homotopy classes

The classification of 3R serial robots based on the number and class of topology of the smooth manifolds of $\mathcal{S}$ was presented in (*40*). Later, the work was extended to classify 3R robots based on cuspidality by (*28*) and (*29*). As the Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ for a 3R robot depends on only q_2 and q_3, singularities are a union of 1-dimensional closed curves, *loops*. The joint space spanned by q_2 and q_3 is a $\mathbb{T}^2$. In the 2-dimensional Euclidean plane, $\mathbb{R}^2$, all loops are homeomorphic to a point, but this does not hold for loops on $\mathbb{T}^2$. The number of homotopy classes depends on the different ways of encircling the generators of the $\mathbb{T}^2$ (*28*). Let n_2 and n_3 be two non-negative integers, then a homotopy class (n_2, n_3) suggests that a loop in $\mathcal{S}$ encircles the q_2- (respectively q_3-) generator n_2 (respectively n_3) times. The most important results from (*28*) and (*29*) that will be used for the presented work are related to the analysis of loops on $\mathbb{T}^2$ by utilizing the degree of t_2 and t_3 in Jacobian determinant $\det \mathbf{J}(\mathbf{q})$. The rational parameterization of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ for a 3R robot in terms of t_2 and t_3 is given as

$$\det \mathbf{J}(q_2, q_3) \rightarrow p(t_2^2, t_3^4).$$

This suggests that for any given value of t_2 (resp. t_3), we can have at most four (resp. two) singular joint configurations. In a non-generic robot, we can also have singularities for all values of t_2 (resp. t_3) for a given value of t_3 (resp. t_2). This often appears as a factored term in Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ (e.g., $\det \mathbf{J} = \cos(q_3) \cdot \mathrm{j}(q_2, q_3)$). It is shown in (*28, 29*) that $n_2 \leq 2$ and $n_3 < 2$. So, a loop on $\mathbb{T}^2$ is homeomorphic to one of the five homotopy classes viz. $(0, 0), (0, 1), (1, 0), (1, 1), (2, 1)$. The previous homotopy-based classification does not consider *nongeneric* 3R robots. We extend the classification of singularities to *nongeneric* 3R robots by considering the conditions for non-

genericity. This is of practical importance as most of the industrial robots with wrist-partitioned architecture have *nongeneric* positional $3R$ chain.

Oneshot embedding of a demonstration

One principled approach to behavior modeling is to represent a demonstrated trajectory as the flow of a stable dynamical system (DS). Once learned, this DS functions as a control policy that inherently exhibits robustness to spatial and temporal perturbations. By incrementally building a library of stable control policies, the system gradually accumulates knowledge of the overall demonstrated task. Each policy captures a locally stable behavior derived from a demonstration, and as more demonstrations are added, the library grows to reflect a more complete understanding of the task, while maintaining stability throughout. As proposed in (*32*), we present here the oneshot mechanism to embed the demonstration as a globally asymptotically stable DS. In doing so, the demonstration is mapped to a stable control policy that ensures global convergence from any initial configuration in the task space.

We begin by considering a goal-directed human demonstration in the workspace $\mathbf{X} = \{\mathbf{x}_i \in \mathcal{X}\}_{i=1}^{M}$ consisting of a sequence of M demonstrated positions, achieving a zero-velocity final state. It is assumed that the demonstration is non self-intersecting. The objective is to model the workspace dynamical system $\dot{\mathbf{x}} = \mathrm{g}(\mathbf{x})$ that accurately models the demonstrated trajectory. The stable attractor of the dynamical system in the workspace $\overset{*}{\mathbf{x}}$ corresponds to the final configuration in the provided demonstration. Instead of modelling the dynamical system directly in the workspace, we learn a diffeomorphic transformation to an auxiliary space, referred to as the latent space $\sqcap$, where the dynamics are analytically defined and globally stable. This transformation allows us to induce an equivalent dynamical system in the workspace, while preserving the desired stability properties through the diffeomorphic mapping. (*32*) proposes a method to generate a (quasi)linear embedding $\mathbf{U} = \{\mathbf{u}_i \in \mathcal{U}\}_{i=1}^{M}$ of the workspace demonstration in a latent space of same dimensionality. The latent embedding depends only on the dimensionality of workspace $\mathcal{X}$, and the number of demonstrated points. Consequently, the method is applicable to generating latent embeddings for any non self-intersecting demonstrations without lack of generality. The globally asymptotically stable DS

governing the latent space, given by

$$\dot{\mathbf{u}} = \mathbf{k} \circ \sqrt{\mathbf{1_n^T} - \mathbf{u}^2} \tag{S3}$$

where $k \in \mathbb{R}^n_+$ is constant, uniquely determined while generating the latent embedding. The attractor for the latent space DS $\overset{*}{\mathbf{u}} = \mathbf{1}_n^T$ serves as the latent embedding of workspace goal $\overset{*}{\mathbf{x}}$. A diffeomorphic mapping $\psi : \mathcal{U} \rightarrow \mathcal{X}$ from the latent space to the workspace is learned to align the latent embedding $\mathbf{U}$ with the demonstrated trajectory $\mathbf{X}$, where $\mathbf{X} = \psi(\mathbf{U})$ Since ψ is a diffeomorphism, the latent space DS from Equation S3 and the workspace DS $\dot{\mathbf{x}} = \mathrm{g}(\mathbf{x})$ are topologically equivalent, preserving the global asymptotic stability of the equilibrium. The workspace DS can be reformulated as

$$\dot{\mathbf{x}} = \mathrm{g}(\mathbf{x}) = \frac{\partial \psi(\mathbf{u})}{\partial \mathbf{u}} \dot{\mathbf{u}} \Big|_{\mathbf{u}=\psi^{-1}(\mathbf{x})} \tag{S4}$$

wherein by substituting from Equation S3, we have the control policy $\mathrm{g}(\mathbf{x})$. The latent embedding is designed such that the resulting workspace DS $\mathrm{g}(\mathbf{x})$ ensures an accurate replication of the demonstrated positions at the demonstrated velocities when starting from the original starting position. Furthermore, the formulation in Equation S4 ensures global asymptotic convergence at the goal position $\mathrm{B}^{(\mathcal{X})}$ for all starting positions in workspace $\mathcal{X}$ as

$$\lim_{t\to\infty} \mathbf{x}_t = \lim_{t\to\infty} \psi(\mathbf{u}_t) = \psi\big(\lim_{t\to\infty} \mathbf{u}_t\big) = \psi(\mathbf{1_n^T}) = \overset{*}{\mathbf{x}} \tag{S5}$$

In doing so, we have embedded a demonstration $\mathbf{X}$ as a temporally invariant and robust policy $\mathrm{g}(\mathbf{x})$, fully specified by pair (ψ, k), where ψ defines the diffeomorphic transformation and $k \in \mathbb{R}^n_+$ is a scalar vector.

The same procedure can also be used to model the behaviour in joint space, from a demonstration $\mathbf{Q} = \{\mathbf{q}_i \in \mathcal{Q}\}$ with a goal joint configuration at $\overset{*}{\mathbf{q}}$, where $\overset{*}{\mathbf{x}} = \mathrm{FK}(\overset{*}{\mathbf{q}})$. The formulated joint space DS $\dot{\mathbf{q}} = \mathrm{g}(\mathbf{q})$ models the behaviour in joint space, and can be uniquely represented by the tuple $(\psi, \mathbf{k})$. The DS embedding of the demonstration is parameter-efficient and robust to perturbations, suitable for robot manipulation as demonstrated in (*32*). It must be noted that neither the workspace DS, nor the joint space DS intrinsically model the robot's kinematic constraints such as joint limits and singularities yet.

Learning the kinematics-aware control policy in the proposed framework (refer to Figure S1) is informed by a three-front analysis of the user-robot setup - (A) Modelling the user intent; (B) Identifying robot-specific kinematic constraints; (C) Modelling the robot control policy.

Modelling the User Behaviour

This section presents the method of modelling the user intent by modelling demonstrated workspace behaviour. We also discuss the challenges posed by the robot's kinematic constraints during replication of the demonstrated behaviour in the workspace.

We consider a workspace demonstration $\mathbf{X}$ provided for a $2R$ robot, that serves as an instance of the task specific behaviour to be modelled by the robot. We assume that the provided demonstration (Figure S2 (a)) does not encounter any singular configuration. The objective is to model the demonstrated behaviour across the workspace. Following the oneshot method presented in Section , we model the workspace behaviour as a globally asymptotically stable DS in the workspace $\dot{\mathbf{x}} = \mathrm{g}^{(\mathcal{X})}(\mathbf{x})$. It is also possible to have multiple workspace demonstrations $\{\mathbf{X_i}\}_{i=1}^{K}$, and use them model the workspace behaviour $\mathrm{B}^{(\mathcal{X})}$ as $\{\mathrm{g}_i^{(\mathcal{X})}\}_{i=1}^{K}$, and is presented in Section 12.

Figure S2 depicts the vector field in the workspace resulting from the control policy $\dot{\mathbf{x}} = \mathrm{g}^{(\mathcal{X})}(\mathbf{x})$, learnt from a single workspace demonstration. There exist two joint configurations for each workspace position. When starting from the original workspace start position, with the robot in the original joint configuration, the learnt behaviour $\mathrm{g}^{(\mathcal{X})}$ is able to generate an identical replication of the demonstration. However, the modelled workspace behaviour $\mathrm{B}^{(\mathcal{X})}$ does not inherently embed the kinematic constraints imposed due to joint limits, singularity partitions, and (or) holes (workspace regions with no IKS) in workspace $\mathcal{X}$. As a result, the learnt control policy is not robust to perturbations. When deviating significantly from the original positions, even though the joint space aspect may be retained, the learnt behaviour can result in singularity and (or) joint limit violations (Figure S2 (A)).

When starting from the original workspace position, with the joint configuration in another aspect, the learnt workspace behaviour leads to the joint limit violation for the first joint (Figure S2 (B)). In such a case, the demonstration itself cannot be replicated. Consequently, enforcing kinematic constraints is complicated by the multiplicity of IKS, their distribution across different

aspects, and the individual constraints bounding the aspect.

Such behavioural ambiguities are even more pronounced for $3R$ robots. Figure 6 depicts the workspace demonstration on a robot, where the starting position has four IKS, whereas the goal position workspace goal position $\overset{*}{\mathbf{x}}$ has only two IKS. In such cases, the original workspace demonstration cannot be replicated for two of the four possible joint configurations without encountering kinematic violations (Figure 6 (C1), (C4)). Thus, it is crucial to detect such configurations as there is no possibility of reaching the goal position without violation.

Replicating the user behaviour while being robust to perturbations, and respecting kinematic constraints cannot be directly addressed via $\mathrm{g}^{(\mathcal{X})}$ in workspace $\mathcal{X}$. While a mechanism to respect kinematic constraints may be devised to operate in conjunction with $\mathrm{g}^{(\mathcal{X})}$, a key issue that remains is that the locus of critical values - singularities in workspace - is hard to derive, and may include nodes, and cusps. Furthermore, the joint limits in workspace do not form a linear variety.

To overcome the shortcomings associated with behavioural modelling in workspace $\mathcal{X}$, we instead focus on modelling the behaviour in joint space joint space $\mathcal{Q}$. Modelling in joint space offers an additional advantage when working with 3R robots. For $3R$ manipulators, as discussed in Section , the singularities are independent of q_1, allowing for singularity-free behaviour replication in the reduced space parameterized by (q_2, q_3) without any loss of information on singularity. Consequently, the joint limits of q_1 can be handled independently, ensuring constraint compliance without altering the behaviour in (q_2, q_3)-space.

The joint space behaviour, hereby represented as $\mathrm{g}^{(\mathcal{Q})}$, will be modelled as a library of joint space DSs, and incrementally generalised (refer to Section 12). For now, we can consider $\dot{\mathbf{q}} = \mathrm{g}^{(\mathcal{Q})}(\mathbf{q})$ to be the control policy in the joint space, learnt using any one of the IKS of the workspace demonstration.

In the proposed framework, the modelled workspace behaviour serves as a proxy of the user intent, whereas the model of joint space behaviour embedded with kinematic constraints, serves as the robot control policy. In doing so, the resulting robot behaviour mirrors the user intent while respecting kinematic constraints.

Robot Categorisation

To comprehensively embed kinematic constraints in the control policy of the robot from the *user–robot* setup, and enable extension to multiple robots, we propose an exhaustive homotopy-based classification of the possible singularities for non-cuspidal $3R$ robots. This classification is instrumental to deriving a stable control law that respect's the robot's kinematic constraints (in Section 1). As a result, the framework is robot-agnostic when learning user behaviour, while still generating robot-specific feasible behaviours during execution.

Singularities of noncuspidal $3R$ robots

If the trajectory obtained from $\mathrm{g}^{(Q)}$ crosses singularities or joint limits before reaching the goal configuration, we propose a method (refer to Section 1) to navigate along the constraint boundary until reaching a feasible joint configuration that lies on the generated trajectory. Once the trajectory re-enters the feasible aspect, it remains constraint-compliant for the rest of the motion, ultimately reaching the goal configuration without further violations. This approach ensures that the trajectory generalizes the expert demonstrated behaviour while adhering to kinematic constraints. Understanding the properties of singularities is essential, as the behaviour near singularities must be deterministic to guarantee convergence to a specific feasible joint configuration. To achieve this, we begin by exhaustively analysing the singularities for noncuspidal $3R$ robots.

For noncuspidal $3R$ robots, the mapping $r : (q_2, q_3) \mapsto (t_2, t_3)$ is a bijection for $q_{2,3} \in (-\pi, \pi)$ (refer Section). As the number of solutions for a fixed value of t_2 (resp. t_3) is at most four (resp. two), a joint path along a q_2- (resp. q_3-) generator may intersect the generic loops at most four (resp. two) times. A similar bound applies for nongeneric robots except for the configurations satisfying the nongenericity condition. At such configurations, the path along the generator coincides with the singularity, and we get infinite solutions.

Topological and differential properties

Using the information related to the degrees of each variable in Jacobian determinant $\det \mathbf{J}(\mathbf{q})$, singularities $\mathcal{S}$ can be classified as the union of branches that form loops on $\mathbb{T}^2$. For example, the singularities $\mathcal{S}$ in Figure 2(A2) can be classified as $2(1, 0) + 2(0, 1)$ with a total of four branches, two

of them encircle q_2 generator and the remaining two encircle the q_3 generator. The classification of singularities $\mathcal{S}$ can be made in multiple ways depending on the purpose of the analysis. (*28*) and (*29*) investigated the number of aspects for a given singularities $\mathcal{S}$ and thus used the enumeration of loops to establish equivalence among 3R robots. In the presented work, we aim to leverage the properties to design robust behaviour near singularity surfaces and, thus, classify the robots based on the encircling nature of the branches and the number of horizontal/vertical turning points on them. This is done as two homeomorphic singularities may require different treatment for the behaviour near the singularity.

Using Equation S2, the upper bound of the turning points can be calculated. (*29*) showed that, at most, eight real horizontal turning points can exist for a generic robot. A similar analysis can be done for vertical turning points. Classification of robots based on the turning points in addition to encircling characteristics have not been reported earlier to the best of the authors' knowledge.

Proposition 1. *The singularity expression of a generic 3R robot has at most three factors.*

Proof. The degrees of t_2 and t_3 in the Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ are known to be 2 and 4, respectively. This allows for at most two intersections along a generator of q_3 and at most four intersections along the generator of q_2. Consequently, the joint space can be partitioned into at most eight aspects when all singularities are aligned with the torus generators formed by q_2 and q_3. Since each factor of $\det \mathbf{J}(\mathbf{q})$ divides the torus into at least two distinct regions, and a maximum of eight such regions is possible, the number of distinct factors is bounded above by three ($2^3 = 8$). An example robot exhibiting this structure is illustrated in Figure S4. □

Each factor f_i of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ may have one, two or four closed loops on the $\mathbb{T}^2$ referred to as branches of the factor.

Proposition 2. *The vertical turning points for any* $3R$ *robot are either at most twelve finite points or infinite points in* $\mathbb{R}$.

Proof. By eliminating t_2 from the rational parameterization of the singularity surface $\det \mathbf{J}(\mathbf{q}) = 0$ and the turning condition $\gamma\, \partial \det \mathbf{J}/\partial q_3 = 0$, we obtain a 14th-degree polynomial in t_3. This polynomial factors as $(t_3^2 + 1) \cdot p_2(t_3)$, where the first factor has only complex roots. Disregarding this, the remaining real polynomial is of degree 12, implying at most 12 real vertical turning points for any non-degenerate branch. If the branch is degenerate, then Equation S2 admits infinitely many solutions. □

Proposition 3. *The horizontal turning points for any branch of singularities of a* $3R$ *robot are either zero, two, or infinite.*

Proof. We prove this by considering the possible topologies of singularity branches $\mathcal{S}$.

Case $(n, 1)$*:* These branches encircle the q_3-generator once. Suppose, for contradiction, that there exist two horizontal turning points $\mathbf{h}_1, \mathbf{h}_2$ on such a branch. Then, for any intermediate q_3 value between $\mathbf{h}_1$ and $\mathbf{h}_2$, a path along the q_2-generator would intersect the branch at least three times, contradicting the known upper bound of two intersections along q_2. Hence, no such horizontal turning points can exist on branches of class $(n, 1)$. A similar contradiction arises for any number greater than two.

Case $(n, 0)$*:* These include closed loops (for $n = 0$) and branches encircling the q_2-generator. By the Weierstraß theorem, generic branches in this class, such as $(0, 0)$ or $(1, 0)$, must have at least two horizontal turning points. If a third point $\mathbf{h}_3$ existed, then for some nearby value along q_2, a path would intersect the branch at least three times, violating the same upper bound. An exception occurs in nongeneric $(1, 0)$ branches where $\det \mathbf{J}(\mathbf{q})$ factors such that one factor is of the form $\cos(q_3 + k_1) + k_2$. In that case, there could be infinitely many horizontal turning points.

Thus, for any branch, the number of horizontal turning points is either zero, two, or infinite. □

The singular configurations of the robot can be characterised by the branch types of the factors present in the determinant of the Jacobian (*28, 29*). For instance, if the factor has two branches of type $(1, 0)$, the factor description can be expressed as $2(1, 0)$. If there are two factors, one with description $2(1, 0)$ and another with description $2(0, 1)$, the singularity is collectively described as $2(1, 0) + 2(0, 1)$

Robot categories

For a $2R$ robot, a single type of singularity $2(1, 0)$ exists and is invariant of the link lengths. For a $3R$ robot, eight homotopy classes of generic robots are known (refer to (*28, 29*)). The presented work extends the classes to nongeneric noncuspidal robots and further classifies the singularities based on the number of turning points. (*45*) conjectured that the homotopy class $3(0, 0)$ and $4(0, 0)$ do not exist (refer to Conjecture 4). The proof of this conjecture has not yet been presented. Still, as the proposed work focuses on only noncuspidal robots, we present an extension to the proposition from (*45*) to exclude the singularities with homotopy class $n(0, 0), n \geq 2$.

Proposition 4. *Any 3R robot belonging to the $n(0,0)$ homotopy class with $n > 1$ must be cuspidal.*

Proof. Ignoring joint limits, we invoke Proposition 4 from Salunkhe (*45*), which states: *A 3R robot in the $n(0,0)$ homotopy class with $n > 1$ must be a quaternary robot*, i.e., a robot that admits at least one pose in the workspace with four inverse kinematic solutions (IKS).

Since the Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ is a C^∞ function, neighboring aspects must have opposite signs of $\det \mathbf{J}(\mathbf{q})$. Therefore, any regular configuration must have an equal number of IKS with positive and negative signs of the determinant.

In the $n(0,0)$ class, there are $(n+1)$ aspects, n interior aspects formed by $(0,0)$ loops, all with the same sign of $\det \mathbf{J}(\mathbf{q})$, and one unique exterior aspect with the opposite sign. Without loss of generality, assume the exterior aspect has a positive determinant. Then, for $n > 1$, the robot has four IKS, of which two must have positive and two negative signs. However, only one aspect has positive sign, forcing both positive IKS to lie within it. Hence, the robot is cuspidal, it contains multiple IKS of the same sign within the same aspect.

□

Previous works enumerated each branch of singularities $\mathcal{S}$ as (n_1, n_2); we extend it by adding the information on the number of horizontal and vertical turning points. Each branch of the singularity is enumerated as $(n_1, n_2)[n_3, n_4]$, which suggests that the given branch encircles the q_2- (resp. q_3-) generator n_1 (resp. n_2) times, and has n_3 (resp. n_4) horizontal (resp. vertical) turning points. In the context of this work, an asterisk ('*') in place of n_1, n_2, n_3 or n_4 suggests any non-negative integer value within bounds or infinity (i.e. $n_1 \leq 2, n_2 \leq 1, n_3 \leq 8$ or $\infty, n_4 \leq 12$ or ∞). Each singularity can now be represented as a union of these branches, and thus a singularity can be written as $\Sigma_{i=1}^{k}(n_{1i}, n_{2i})[n_{3i}, n_{4i}]$, where i denotes each branch of the singularity. For example, the robot singularity in Figure S3A is denoted as $2(1,0)[2,0] + 2(1,0)[\infty,0]$ as it has two branches of type $(1,0)[2,0]$ and two of $(1,0)[\infty,0]$.

Using the aforementioned novel enumeration of the branches, Propositions 4, and 5, the singularities of noncuspidal $3R$ robot can be classified into six categories:

1. Non-loop, non-intersecting branches with ONLY horizontal turning points

2. Non-loop, non-intersecting branches with NO horizontal turning points

3. Non-loop, intersecting branches

4. Non-loop, non-intersecting branches with folds

5. Loop non-intersecting branch

6. Intersecting branches including a loop

Category 1: Non-loop, non-intersecting branches with only horizontal turning points: The singularity of robots in this category is made out of the union of branches that only have horizontal turning points, and do not intersect. Few types of singularities exhibited by robots in this category: $2(1,0)[2,0] + 2(1,0)[\infty,0]$ (refer to Figure S3(A)), $4(1,0)[2,0]$.

Category 2: Non-loop, non-intersecting branches with no horizontal turning points: The singularity of robots in this category is made out of the union of branches that do not have any horizontal turning points, and the branches that do not intersect. Few types of singularities exhibited by robots in this category: $(0,1)[0,4] + (0,1)[0,2]$ (refer to Figure S3(B)), $(1,1)[0,2] + (1,1)[0,0]$ (refer to Figure S3(E)), $2(0,1)[0,a>0], 2(1,1)[0,a>0]$, where a is the finite number of vertical turning points on a branch of singularity.

It can be seen that the joint limits further divide the joint space, creating more partitions, and methods different from *Category 1* are needed to be employed to investigate connectivity between two configurations (as discussed in Section 12).

Category 3: Non-loop, intersecting branches:

The singularity of robots in this category is made out of the union of branches that do not belong to the $(0,0)$ homotopy class and the branches that intersect. Few types of singularities exhibited by robots in this category: $2(1,0)[\infty,0] + 2(0,1)[0,*]$ (refer to Figure S3(F)), $4(1,0)[\infty,0]+2(0,1)[0,\infty]$ (refer to Figure S4, $2(1,0)[\infty,0]+2(1,1)[0,0]$ (refer to Figure S3(H)). This category consists of non-generic robots only, and the $(1,0)[\infty,0]$ branch comes from a factored Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ and divides joint space $\mathcal{Q}$ in at least four aspects. The singularities in Figure 2(A2) is one such example of $2(1,0)[\infty,0] + 2(0,1)[0,2]$. We also show an example of a robot with eight aspects (without joint limits), which is the maximum limit of an aspect without joint limits for *any* $3R$ robot (refer to Figure S4). Interestingly, the intersecting singularities

must have $(1,0)[\infty,0]$ branch as the maximum number of intersections a path can have along the q_2-generator that is not a singularity itself is two.

Category 4: Non-loop, Non-intersecting branches with folds: The singularity of robots in this category is made out of the union of branches that belong to the $(1,0)[2,*]$ homotopy class with at least one branch with $(1,0)[2,>0]$ homotopy classification, and the branches do not intersect. Here, a is a non-zero definite number of vertical turning points. These singularities are the most challenging when defining behaviour near the boundaries, as they have branches that encircle the q_2 generator; however, for some regions, they locally exhibit loop-like behaviour. The sense of direction may not remain constant while traversing between any two configurations on the branch. For such singularities, the sense of direction has to be isolated in regions by identifying the turning points along the branch. Similar to the loop singularities, the relative position of the configurations in an aspect may change due to the branches intersecting with joint limits, which further complicates the motion along a branch of singularity of this class.

The theoretical upper bound for the vertical turning points on a branch is twelve, but we could only find examples with two and four vertical turning points on a branch. Figure S9 shows an example of robot classes with $(1,0)[2,0] + (1,0)[2,2]$ singularities. Figure S3 shows singularities where one branch (bottom) has two horizontal and two vertical turning points and encircles the q_2-generator. Figure S9 shows singularities where one branch (top) has two horizontal and two vertical turning points and encircles the q_2- generator, while the other branch (bottom) has four vertical turning points.

Category 5: Loop non-intersecting branch:

The singularity of robots in this category is made out of a single branch of the $(0,0)$ homotopy class. Few types of singularities exhibited by robots in this category: $(0,0)[2,2]$ and $(0,0)[2,6]$(refer to Figure S3(C)). This branch is guaranteed to have at least two horizontal and vertical turning points. The motion along the loop boundaries differs from the above categories. In this category, the sense of motion is defined in a clockwise or counterclockwise direction.

Category 6: Intersecting branches including a loop: The singularity of robots in this category is

made out of the union of branches that belong to the $(0,0)[2,*]$ and $(1,0)[\infty,0]$ types, and the branches intersect. This class has only one type of nongeneric robot with $2(1,0)[\infty,0]+(0,0)[2,2]$ class of singularities. The $(1,0)[\infty,0]$ branch comes from a factored $\det\mathbf{J}(\mathbf{q})$ and divides $\mathcal{J}$ in at least four aspects. This case requires the consideration of both intersecting (Category 3) and loop (Category 5) singularities. An example illustration of this type of robot is given in Figure S3(G), where the loop is complete in the $(-\pi,\pi)$ range and in Figure S8, where the loop is split due to joint limits.

Algorithm 1. Branch types in factor f_i

Solve $\mathrm{f}=0, \partial\,\mathrm{f}/\partial q_2=0 \Rightarrow Q_h$ (horizontal turning points)

Solve $\mathrm{f}=0, \partial\,\mathrm{f}/\partial q_3=0 \Rightarrow Q_v$ (vertical turning points)

If $|Q_h|>0$ and $Q_v=\emptyset$:

    Return $(1,0)[\geq 2,0]$

Else if $Q_h=\emptyset$ and $|Q_v|>0$:

    Check if branch encircles the q_2-generator

    If $\exists\,\mathbf{q}\in Q_v$ s.t. path $\mathbf{q}(q_2)$ has no solution:

        Return $(0,1)[0,*]$

    Else: Return $(1,1)[0,*]$

Else if $|Q_h|>0$ and $|Q_v|>0$:

    If $|Q_h|=2$: Return $(0,0)[2,\geq 2]$

    Else: Return $(1,0)[2,*]$

Else:

    Compute $J=\{\mathbf{q}\mid \mathrm{f}(\mathbf{q}(q_2=\pi))=0\}$

    If $|J|>2$: Return $(2,1)[0,0]$

    Else: Return $(1,1)[0,0]$

To perform the categorisation of a robot into one of the six categories, we present Algorithm 1 that analyses all the branches associated with a single factor f_i, based on the encirclement of the generators, combined with the number of horizontal turning points Q_h, and vertical turning points Q_v. By repeating this for each factor of Jacobian determinant $\det\mathbf{J}(\mathbf{q})$ of a robot, we can comprehensively

generate a description of the robot.

If there are nonzero horizontal and zero vertical turning points, the factor is a union of $(1, 0)$ branches, as discussed in Proposition 4. If only vertical turning points exist, the factor cannot have a $(1, 0)$ or a $(0, 0)$ branch. Similarly, nonzero vertical turning points cannot exist in $(2, 1)$ branch, and so the only possibility is that the factor is $2(0, 1)$ or $2(1, 1)$. If the given factor has no horizontal turning points, then the singularities may belong to either $(0, 1)$ or $(1, 1)$ or $(2, 1)$. In this case, the fact that singularities with $(0, 1)$ homotopy class does not encircle the q_2-generator is leveraged to identify such singularities.

In the case that both horizontal and vertical turning points exist, the only classes that can be factors of type $2(1, 0)$ wherein at least one branch has non-zero vertical turning points or $(0, 0)[2, *]$. In such cases, the classification can be done based on the number of horizontal turning points. If a factor is of a $(0, 0)$ homotopy class, then it must have strictly two horizontal turning points as shown in Proposition 4, while a factor with description $2(1, 0)$ homotopy class will strictly have four horizontal turning points. If there are no horizontal/vertical turning points, then the factor is either $2(1, 1)$ or $2(2, 1)$ class. As the branch with $2(2, 1)$ class have two branches that encircle q_2-generator twice, the number of intersections of the branch with any path along the q_3-generator will be four. Thus, the singularity class of the factor can be determined by the number of roots of the factor at $q_2 = \pi$.

In this section, we have identified six categories that exhaustively classify generic and nongeneric $3R$ noncuspidal robots by using the homotopy-based classification with additional properties. This novel categorisation is instrumental to autonomously embed kinematic constraints into the control policy as discussed in Sections 1, and 12.

Behaviour near Kinematic Constraints

The current joint space behaviour $\mathrm{g}^{(\mathcal{Q})}$, though stable and converges at the goal configuration $\overset{*}{\mathbf{q}}$, does not yet explicitly embed the kinematic constraints prior to convergence. For a given instantaneous joint configuration $\mathbf{q}$, the trajectory mapped by following the DS $\dot{\mathbf{q}} = \mathrm{g}^{(\mathcal{Q})}(\mathbf{q})$ has guaranteed convergence at the goal configuration $\overset{*}{\mathbf{q}}$. However, this trajectory may violate the joint limits or encounter singular configuration(s) enroute.

To avoid such scenarios, the modelled behaviour $\mathrm{g}^{(\mathcal{Q})}$ is selectively overlayed with a *near-boundary* control policy around the joint limits and singularities that enclose the aspect. Effectively, this policy ensures that the start configuration $\mathbf{q}$, the goal configuration $\overset{*}{\mathbf{q}}$, and the intermediate trajectory, are strictly within the bounds of a single aspect in the joint space when $\mathrm{g}^{(\mathcal{Q})}$ may lead to a violation. We present the near-boundary behaviour for non-cuspidal $3R$ robots in Section 5.

Boundary regions for $2R$ robots

The kinematic constraints in the joint space of a $2R$ robot are due to the first joint's limits $q_1^{\times} = \{q_1 = \underline{q_1}, q_1 = \overline{q_1}\}$, a singularity boundary $S^{\times} = \{q_2 = 0\}$, and the second joint's limits $q_2^{\times} = \{q_2 = \underline{q_2}, q_2 = \overline{q_2}\}$. Each of the two aspects is the joint space are bound by these constraints.

We begin by defining regions $(\mathcal{B}_{q_1^{\times}}, \mathcal{B}_{q_2^{\times}})$ that encapsulate the neighbourhoods of the joint limits in joint space $\mathcal{Q}$

$$\begin{aligned} \mathcal{B}_{q_1^{\times}} &= \{\mathbf{q} \in \mathcal{Q} \mid \mathbf{q}(q_1) < \underline{q_1} + \epsilon_q \vee \mathbf{q}(q_1) > \overline{q_1} - \epsilon_q\} \\ \mathcal{B}_{q_2^{\times}} &= \{\mathbf{q} \in \mathcal{Q} \mid \mathbf{q}(q_2) < \underline{q_2} + \epsilon_q \vee \mathbf{q}(q_2) > \overline{q_2} - \epsilon_q\} \end{aligned} \tag{S6}$$

where $\epsilon_q > 0$ is the safety margin (in radians) around the joint limits. The joint limit safety margin is next used to calculate the singularity threshold $\epsilon_S = |\det \mathbf{J}(q_1 = 0, q_2 = \epsilon_q)|$, which is used to define the neighbourhood region of singularity $\mathcal{B}_{S^{\times}}$

$$\mathcal{B}_{S^{\times}} = \{\mathbf{q} \in \mathcal{Q} \mid \epsilon_S > |\det \mathbf{J}(\mathbf{q})|\}$$

Collectively, $\mathcal{B}_{q_1^{\times}}, \mathcal{B}_{q_2^{\times}}$, and $\mathcal{B}_{S^{\times}}$ encapsulate the regions of joint space $\mathcal{Q}$ in vicinity of the joint limits and singularities. Since each aspect in joint space $\mathcal{Q}$ is in itself bounded by the joint limits and singularities, the region near boundaries of the aspect belongs to a subset of $\mathcal{B}_{q_1^{\times}} \cup \mathcal{B}_{q_2^{\times}} \cup \mathcal{B}_{S^{\times}}$. While following $\mathrm{g}^{(\mathcal{Q})}$, if the robot assumes a configuration in the boundary region, the continued

usage of $g^{(Q)}$ may result in a joint trajectory that crosses singularities or joint limits before reaching the goal configuration.

Identifying boundary crossings for $2R$ robots

Consider a trajectory $\mathbf{Q}$ that can be generated by $g^{(Q)}$ from a query point $\mathbf{q} \in \mathcal{A}$. Due to lack of embedded constraints, it is possible for this trajectory to cross the boundaries $(q_1^{\times}, q_2^{\times}, S^{\times})$ enclosing the current aspect. However, it must be noted that both the query $\mathbf{q}$ (start of trajectory) and the goal configuration $\overset{*}{\mathbf{q}}$ lie in the same aspect $\mathcal{A}$. This implies that if a trajectory crosses boundary(ies) and transitions into different aspect(s), it must eventually return to the original aspect and cross some boundary one last time to achieve convergence at the goal configuration.

Proposition 5. *Let $\mathbf{q}_t \in \mathcal{A}$ and $\overset{*}{\mathbf{q}} \in \mathcal{A}$ be the initial and goal configurations in a joint-space aspect $\mathcal{A} \subset \mathcal{Q}$, enclosed by the set of boundaries $\{q_1^{\times}, q_2^{\times}, S^{\times}\}$. Let $\mathbf{Q} = \{\mathbf{q}(t)\}_{t\geq 0}$ be a trajectory generated by the control policy $g^{(Q)}$. Then, regardless of how many times $\mathbf{Q}$ leaves and re-enters $\mathcal{A}$, there exists a time $T < \infty$ such that $\mathbf{q}(t) \in \mathcal{A}$ for all $t > T$, i.e., the trajectory must make a final crossing into $\mathcal{A}$ to converge to $\overset{*}{\mathbf{q}}$.*

Proof. The control policy $g^{(Q)}$ is defined as a globally asymptotically stable dynamical system with respect to the attractor $\overset{*}{\mathbf{q}} \in \mathcal{A}$. Thus, for any initial configuration $\mathbf{q}_t \in \mathcal{Q}$, the trajectory $\mathbf{Q}$ satisfies $\lim_{t\to\infty} \mathbf{q}(t) = \overset{*}{\mathbf{q}}$. Since the goal lies within $\mathcal{A}$, and the aspect is bounded by joint limits and singularity manifolds (i.e., the set $\mathcal{B}$), convergence requires that the trajectory eventually remains within $\mathcal{A}$. Therefore, there exists a finite time T such that $\mathbf{q}(t) \in \mathcal{A}$ for all $t > T$.

□

Consequently, the generated trajectory $\mathbf{Q}$ can be segregated into three segments - the initial constraint-compliant portion originating at $\mathbf{q} \in \mathcal{A}$ before the first boundary violation and exit from $\mathcal{A}$, the final constraint-compliant portion in $\mathcal{A}$ after the last violation resulting in convergence at goal, and the intermediate region which begins with an exit from $\mathcal{A}$ and terminates at the last reentry back into the this aspect. The intermediate portion of the trajectory is not constraint compliant, and may encounter multiple violations. For this region, we propose a control law that overlays $g^{(Q)}$ beginning in the boundary region of the first exit from $\mathcal{A}$, to reactively navigate along the constraint boundary within the boundary region, until reaching the last reentry joint configuration. This approach ensures that the trajectory remains as faithful as possible to the expert behaviour,

without exiting the aspect and adhering to the kinematic constraints.

Algorithm 2. First Exit and Last Entry configurations for $2R$ Robots

Input: Joint trajectory $\mathbf{Q}$

Detect singularity crossings

$Z \leftarrow \text{sign}(\det \mathbf{J}(\mathbf{q} \in \mathbf{Q})) \cdot \text{sign}(\det \mathbf{J}(\overset{*}{\mathbf{q}}_{\text{goal}}))$

$\Rightarrow$ Assign $+1$ for configurations in same aspect as goal,

0 for singularities, and -1 for other aspects

Detect joint limit violations

$V \leftarrow \{i \mid \mathbf{Q}_i(q_1) \notin [\underline{q_1}, \overline{q_1}] \ \vee \ \mathbf{Q}_i(q_2) \notin [\underline{q_2}, \overline{q_2}]\}$

For all $i \in V$, set $Z_i \leftarrow -1$

Identify transition indices

$\text{first exit} \leftarrow \max\{i \mid \forall j < i,\ Z_j = +1\}$

$\text{last entry} \leftarrow \min\{i \mid \forall j > i,\ Z_j = +1\}$

Return: $(\mathbf{Q}_{\text{first exit}},\ \mathbf{Q}_{\text{last entry}})$

To model the behaviour for the intermediate non-compliant segment of $\mathbf{Q}$, we reduce our analysis to only the first and last aspect boundary crossings - the first exit configuration from the current aspect ($\mathbf{q}_{\text{first exit}}$) and the last entry configuration ($\mathbf{q}_{\text{last entry}}$) - respectively marking the beginning and end of the non-compliant segment. While it is possible that crossing over from the current aspect and back into it occurs multiple times along $\mathbf{Q}$, considering only the first exit and last entry to determine the behaviour prevents limit cycles and self-intersections in the boundary region. The exit and entry points can be determined on the basis of discontinuity in aspect criteria i.e. $\text{sign}(\det \mathbf{J}(\mathbf{q}))$. The procedure to determine these joint configurations for $2R$ robots is presented in Algorithm 2.

Motion along boundaries for $2R$ robots

The behaviour to reach $\mathbf{q}_{\text{last entry}}$ from $\mathbf{q}_{\text{first exit}}$ involves traversing along the normal to the gradient of the boundaries. During this traversal, the boundary region of a configuration $\mathbf{q} \in \mathcal{Q}$ is identified

as $\mathcal{B}(\mathbf{q}) \in \{\mathcal{B}_{q_1^\times}, \mathcal{B}_{q_2^\times}, \mathcal{B}_{S^\times}\}$. The overall behaviour depends on the combination of $\mathcal{B}(\mathbf{q}_{\text{last entry}})$ and $\mathcal{B}(\mathbf{q}_{\text{first exit}})$, as the predicted trajectory can cross the aspect boundary in multiple ways. For a given combination of $(\mathbf{q}_{\text{last entry}}, \mathbf{q}_{\text{first exit}})$ we define a track cycle $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ which associates joint configurations in each boundary region of the relevant aspect with a goal velocity direction $\mathbf{v}(\mathbf{q}) = \mathrm{C}(\mathcal{B}(\mathbf{q}))$ and guides the robot safely to $\mathbf{q}_{\text{last entry}}$ configuration. The normal to the violation boundary $\mathbf{t}$ is determined using Algorithm 3. The near-boundary control policy $\mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$ that determines the joint velocity for a point $\mathbf{q}$ in the boundary $\mathcal{B}(\mathbf{q})$ on an aspect is described by the DS

$$\dot{\mathbf{q}} = \mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}}) = \|\mathbf{q}_{\text{last entry}} - \mathbf{q}\| \frac{\mathbf{v} \cdot \mathbf{t}^\perp}{\|\mathbf{v}\|} \mathbf{t}^\perp \tag{S7}$$

where $\mathbf{t}^\perp$ is the normal to $\mathbf{t}$.

Algorithm 3. Normal vector to the constraint boundary for $2R$ Robot

Input: Configuration $\mathbf{q}$ near aspect boundary

Boundary normal to $q_1^\times$

If $\underline{q_1} \leq \mathbf{q}(q_1) < \underline{q_1} + \epsilon_q$:

$\mathbf{t}_{q_1^\times} \leftarrow [1,\ 0]^T$

Else if $\overline{q_1} - \epsilon_q \leq \mathbf{q}(q_1) < \overline{q_1}$:

$\mathbf{t}_{q_1^\times} \leftarrow [-1,\ 0]^T$

Boundary normal to $q_2^\times$

If $\underline{q_2} \leq \mathbf{q}(q_2) < \underline{q_2} + \epsilon_q$:

$\mathbf{t}_{q_2^\times} \leftarrow [0,\ 1]^T$

Else if $\overline{q_2} - \epsilon_q \leq \mathbf{q}(q_2) < \overline{q_2}$:

$\mathbf{t}_{q_2^\times} \leftarrow [0,\ -1]^T$

Boundary normal to singularity surface $S^\times$

$\mathbf{t}_{S^\times} \leftarrow \mathrm{sign}(\det \mathbf{J}(\mathbf{q})) \cdot [\nabla_{q_1,q_2} \det \mathbf{J}(\mathbf{q})]^T$

Return: $(\mathbf{t}_{q_1^\times},\ \mathbf{t}_{q_2^\times},\ \mathbf{t}_{S^\times})$

The track cycle behaviour that connects $\mathbf{q}_{\text{first exit}}$ and $\mathbf{q}_{\text{last entry}}$ is determined using Algorithm 4. An example is presented in Figure S10, wherein the direction in red specifies the behaviour in $\mathcal{B}_{q_1^\times}$,

while direction in blue specifies the behaviour in $\mathcal{B}_{S^\times}$. In Figure S10, the path $\mathbf{Q}$ generated by $g^{(Q)}$, encounters the first violation at $\mathbf{q}_{\text{first exit}}$, $\mathcal{B}(\mathbf{q}_{\text{first exit}}) = \mathcal{B}_{S^\times}$. The last configuration in $\mathbf{Q}$ in violation of the kinematic constraints is $\mathbf{q}_{\text{last entry}}$ $\mathcal{B}(\mathbf{q}_{\text{last entry}}) = \mathcal{B}_{S^\times}$, and $\mathbf{q}_{\text{first exit}}(q_1) > \mathbf{q}_{\text{last entry}}(q_1)$. To generate the resulting trajectory, we start at $\mathbf{q} = \mathbf{q}_{\text{first exit}}$ and terminate at $\mathbf{q}_{\text{last entry}}$. At the start, $\mathbf{v}_{\mathbf{q}_{\text{first exit}}} = [\mathbf{q}_{\text{last entry}}(q_1) - \mathbf{q}_{\text{first exit}}(q_1), 0]^T$ forces the joint velocity to be along negative q_1 axis (blue). Progressively, the joint configuration, $\mathbf{q}$, approaches $\mathcal{B}_{q_1^\times}$. When $\mathbf{q} \in \mathcal{B}_{q_1^\times}$, $\mathbf{v} = [0, -\mathbf{t}_{\text{last entry}}(q_2)]^T$ forces the joint velocity along positive q_2 axis directed parallel to $q_1^\times$ while progressively approaching $\mathcal{B}_{q_2^\times}$ (blue). Lastly, when $\mathbf{q} \in \mathcal{B}_{q_2^\times}$, $\mathbf{v} = [\mathbf{q}_{\text{last entry}}(q_1) - \mathbf{q}(q_1), 0]^T$ forces the joint velocity along q_1 axis and approach $\mathbf{q}_{\text{last entry}}$ configuration (red). The overall behaviour encapuslated by $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ is thus

$$(\mathbf{q}_{\text{first exit}} \in \mathcal{B}_{S^\times}) \rightarrow (\mathbf{q} \in \mathcal{B}_{q_1^\times}) \rightarrow (\mathbf{q} \in \mathcal{B}_{q_2^\times}) \rightarrow (\mathbf{q}_{\text{last entry}})$$

The track cycle $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ determines the behaviour to reach $\mathbf{q}_{\text{last entry}}$ from any point in any of the boundary regions in the current aspect.

Algorithm 4. Algorithm: Track Cycle for a $2R$ Robot

Input: Configuration $\mathbf{q}$, aspect boundary normals

Output: Velocity direction vector $\mathbf{v}$ along tangential track cycle

Compute $\mathbf{t}_{\text{first exit}}$, $\mathbf{t}_{\text{last entry}}$, $\mathbf{t}$ using Algorithm 3

Case 1: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{S^\times} \cup \mathcal{B}_{q_2^\times}$

If $\mathbf{q} \in \mathcal{B}_{q_1^\times}$:

$\mathbf{v} \leftarrow [0,\ -\mathbf{t}_{\text{last entry}}(q_2)]^T$

Else if $\mathbf{q} \in \mathcal{B}_{q_2^\times} \cup \mathcal{B}_S^\times$ and $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) < 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_1) - \mathbf{q}_{\text{first exit}}(q_1),\ 0]^T$

Else if $\mathbf{q} \in \mathcal{B}_{q_2^\times} \cup \mathcal{B}_S^\times$ and $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) \geq 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_1) - \mathbf{q}(q_1),\ 0]^T$

Case 2: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{S^\times} \cup \mathcal{B}_{q_1^\times}$

If $\mathbf{q} \in \mathcal{B}_{q_2^\times} \cup \mathcal{B}_S^\times$:

$\mathbf{v} \leftarrow [-\mathbf{t}_{\text{last entry}}(q_1),\ 0]^T$

Else if $\mathbf{q} \in \mathcal{B}_{q_1^\times}$ and $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2)]^T$

Else if $\mathbf{q} \in \mathcal{B}_{q_1^\times}$ and $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) \geq 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2)]^T$

It is possible that a joint configuration may lie simultaneously in two boundary regions - namely $\mathcal{B}_{q_1^\times} \cap \mathcal{B}_{q_2^\times}$ and $\mathcal{B}_{q_1^\times} \cap \mathcal{B}_{s^\times}$. Such configurations have two potential candidates for the boundary normal $\mathbf{t}$. If $\mathbf{q}_{\text{last entry}}$ is one such configuration, then both normals qualify and any one can be used as $\mathbf{t}_{\text{last entry}}$ to determine the track cycle behaviour. Otherwise, the joint configurations in the overlapping regions, constitute the transition from one boundary region to the next - as determined by the track cycle. The ideal goal velocity direction $\mathbf{v}_{\text{ideal}}$ is the direction which has a non-negative projection on all $\mathbf{t}$ candidates. The track cycle design ensures that there is a single direction that matches this criteria and maintains a non-repeating sequence of boundary regions to traverse, terminating at $\mathbf{q}_{\text{last entry}}$. The ideal boundary normal $\mathbf{t}_{\text{ideal}}$ is the boundary normal associated with this chosen direction. The resulting joint velocity is calculated using Algorithm 5.

Algorithm 5. Joint Velocity for $2R$ Robot in Overlapping Boundary Regions

---

Input: Configuration $\mathbf{q} \in \mathcal{B}_{q_1^\times} \cap \mathcal{B}_{q_2^\times}$ or $\mathbf{q} \in \mathcal{B}_{q_1^\times} \cap \mathcal{B}_{S^\times}$

Output: Desired joint velocity $\dot{\mathbf{q}}$

Call Algorithm 3 to compute all boundary normals:

$\{\mathbf{t}\} \leftarrow$ set of normals at $\mathbf{q}$

Determine all candidate velocity vectors:

$\{\mathbf{v}\} \leftarrow$ track-cycle velocities from classification $\mathrm{C}(\mathbf{q}_{\text{last entry}})$

Determine ideal goal velocity direction

$\mathbf{v}_{\text{ideal}} \leftarrow \{\mathbf{v} \in \{\mathbf{v}\} \mid \mathbf{v} \cdot \mathbf{t} \geq 0,\ \forall\, \mathbf{t} \in \{\mathbf{t}\}\}$

Determine ideal boundary normal

$\mathbf{t}_{\text{ideal}} \leftarrow$ boundary normal corresponding to $\mathbf{v}_{\text{ideal}}$

Return projected velocity

$$\dot{\mathbf{q}} = \|\mathbf{q}_{\text{last entry}} - \mathbf{q}\| \, \frac{\mathbf{v}_{\text{ideal}} \cdot \mathbf{t}_{\text{ideal}}{}^{\perp}}{\|\mathbf{v}\|} \, \mathbf{t}_{\text{ideal}}{}^{\perp}$$

The near-boundary control policy $\mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$ from Equation S7 is also globally asymptotically stable at $\mathbf{q}_{\text{last entry}}$. Consequently, the track cycle models the behaviour across all boundary regions, while approaching $\mathbf{q}_{\text{last entry}}$ in a stable manner. An example track-cycle DS is shown in Figure S10.

Overall behaviour for $2R$

The overall behaviour in an aspect $\mathcal{A}$ of the joint space, is a resultant of the modelled behaviour $\mathrm{g}^{(\mathcal{Q})}(\mathbf{q})$ and the near-boundary control policy $\mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$.

For a $2R$ robot, the joint velocity is regulated as

$$\dot{\mathbf{q}} = \begin{cases} \mathrm{g}^{(\mathcal{Q})}(\mathbf{q}) & \text{if } \mathbf{q} \notin \{\mathcal{B}_{q_1^\times}, \mathcal{B}_{q_2^\times}, \mathcal{B}_{S^\times}\} \\ \mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}}) & \text{otherwise} \end{cases} \tag{S8}$$

where the track cycle $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ is determined using Algorithm 4 as required.

Motion along boundaries for non-cuspidal $3R$ robots

For $3R$ robots, it has been shown how the singularities can be studied in a reduced dimension parameterized by q_2 and q_3 only. Thus, the extension of near-boundary policy to $3R$ non-cuspidal robots leverages the reduced space parameterisation (q_2, q_3). Violations due to first joint's limits do not lead to crossing over into another aspect, as the separation of aspects in joint space is independent of q_1. Thus, the joint velocities for second and third joints $\dot{\mathbf{q}}(q_2, q_3)$ is collectively determined by taking the tangent to the relevant factor of the singularity in the $q_2 - q_3$ slice. The behaviour near the first joint limit's violation $q_1^\times$ is presented in Section 9.

The process for navigating the near-boundary region is an extension to the case of $2R$ robot, and the procedure can be summarised as:

- Identify the first and last boundary crossings from the aspect in $q_2 - q_3$ slice
- Determine the track cycle $\mathrm{C}(\mathbf{q}_{\text{last entry}})$

A $2R$ robot always has exactly two aspects. However, for a $3R$ robot, this depends on a case-by-case basis. When not considering joint limits for defining aspects, there may be upto eight aspects in the joint space (refer to Proposition 2). However, by considering the joint limits, the joint space may get further divided into more aspects. As a result, robots having singularities with identical classifications, may vary in the number of aspects. Figure S12 presents the aspects for a $3R$ robot from Category 3 with singularity class $2(1,0)[\infty,0] + 2(0,1)[0,2]$, in the $q_2 - q_3$ slice of the joint space containing nine aspects.

The aspect for $3R$ robots can be identified using the tuple composed of sign of each factor of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$. For instance, if there are two factors

$$\mathcal{A}(\mathbf{q}) = (\text{sign}(\text{f}_1(\mathbf{q})), \text{sign}(\text{f}_2(\mathbf{q}))) \tag{S9}$$

Such factor-sign based identification, in conjunction with $q_2^{\times}$ and $q_3^{\times}$ joint limits, is used to identify the first exit from and last entry into the relevant aspect in the joint space (similar to Algorithm 2). We study each factor independently to determine the near-boundary control policy $\text{g}^{(C)}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$. This is done so that the intersection points in singularities of nongeneric robots can be handled efficiently and the boundary near each branch can be designed with proper offsets.

Algorithm 6. Boundary Crossings for $3R$ Robots (Reduced Space)

Input: Trajectory $\mathbf{Q}$ in joint space, goal configuration $\overset{*}{\mathbf{q}}$

Output: First exit point and last entry point in $\mathbf{Q}$

Identify Singularity crossings

Assign:

$+1$ for configurations in same singularity-free region as $\overset{*}{\mathbf{q}}$

0 for singularities, and -1 otherwise

$Z_{\text{sim}} \leftarrow \text{sign}(\det \mathbf{J}(\mathbf{q} \in \mathbf{Q})) \cdot \text{sign}(\det \mathbf{J}(\overset{*}{\mathbf{q}}))$

Identify Joint limit violations

$\text{violation}(q_2) \leftarrow \{i \mid \mathbf{Q}[i](q_2) \notin [\underline{q_2}, \overline{q_2}]\}$

$\text{violation}(q_3) \leftarrow \{i \mid \mathbf{Q}[i](q_3) \notin [\underline{q_3}, \overline{q_3}]\}$

$\text{violations} \leftarrow \text{violation}(q_2) \cup \text{violation}(q_3)$

For all indices in violations: set $Z_{\text{sim}}[i] \leftarrow -1$

Find transition indices

$i_{\text{first exit}} \leftarrow \max\{i \mid \forall j < i,\ Z_{\text{sim}}[j] = +1\}$

$i_{\text{last entry}} \leftarrow \min\{i \mid \forall j > i,\ Z_{\text{sim}}[j] = +1\}$

Return: $(\mathbf{Q}[i_{\text{first exit}}],\ \mathbf{Q}[i_{\text{last entry}}])$

The behaviour to reach $\mathbf{q}_{\text{last entry}}$ from $\mathbf{q}_{\text{first exit}}$ involves $\dot{\mathbf{q}}$ along the tangent to the boundary. The normal to the boundary $\mathbf{t}$ is calculated in $q_2 - q_3$ slice, similar to Algorithm 3. However, as Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ can have up to three factors, the normal to the singularity is calculated by considering the relevant factor of the singularity ($\nabla_{q_2,q_3} f(\mathbf{q})\ \text{sign}(f(\mathbf{q}))$). To determine the boundary region near singularity, we determine the singularity threshold as

$$\epsilon_S = \min\left\{\{\mathrm{f}_i\left(\mathbf{q}(q_2, q_3) + [0, \epsilon_S]^T\right), \mathrm{f}_i\left(\mathbf{q} + [\epsilon_S, 0]^T\right)\}\right\}$$

using all the factors f_i of $\det \mathbf{J}$. Here, $\mathbf{q}_h(q_2, q_3) \in Q_h \cup I_{SL}(q_2^{\times})$ represents the collection of horizontal turning points and the points of intersection between the singularity and $q_2^{\times}$ joint limits, whereas $\mathbf{q}_v(q_2, q_3) \in Q_v \cup I_{SL}(q_3^{\times})$ represents the collection of vertical turning points and the points of intersection between the singularity and $q_3^{\times}$ joint limits. The boundary region around each factor is defined as

$$\mathcal{B}_{\mathrm{f}_i^{\times}} = \{\mathbf{q} \in \mathcal{Q} \mid \epsilon_S > |\mathrm{f}_i(\mathbf{q})|\}$$

Algorithm 7. Normal to the Boundary for $3R$ **Robots**

Input: Near-boundary region $\mathcal{B}_i$ containing the current joint configuration $\mathbf{q}_t$

Output: Normal direction to the constraint boundary

If $\mathcal{B}_i = \mathcal{B}_{q_2^{\times}}$:

  If $\mathbf{q}_t(q_2) = \underline{q_2}$:

    Return $[1,\ 0]^T$

  Else:

    Return $[-1,\ 0]^T$

Else if $\mathcal{B}_i = \mathcal{B}_{q_3^{\times}}$:

  If $\mathbf{q}_t(q_3) = \underline{q_3}$:

Return $[0,\ 1]^T$

Else:

Return $[0,\ -1]^T$

Else (near singular configuration):

$f \leftarrow$ factor of $\det \mathbf{J}$ associated with $\mathcal{B}_i$

Return $\nabla_{q_2,q_3} f(\mathbf{q}_t) \cdot \text{sign}(f(\mathbf{q}_t))$

Unlike $2R$ robots, the singularities in $3R$ robots vary depending on the robot architecture and need to be considered case-wise using the classification studied in Section 4. Consequently, multiple types of track cycles exist depending on the singularity class. Singularities with branches $(m, 0)[2, n]$, where $m \in 0, 1$ and $n \in [2, 12]$ have to be treated seperately. This is because the singularity branch either forms a closed loop or has a 'meander' like curve (refer to Figures S6 and S9). Other classes of singularities have branches that encircle at least one generator and only have horizontal or vertical turning points on a branch. An example of such a singularity with its aspects colored and the boundaries near the singularity and the joint limit violations marked is shown in Figure S12.

The track cycle behaviour that connects $\mathbf{q}_{\text{first exit}}$ and $\mathbf{q}_{\text{last entry}}$ is determined using Algorithm 8 for robots from Categories 1 through 3. As an example, let us consider a $3R$ robot with singularity class $2(1, 0)[2, 0] + 2(1, 0)[\infty, 0]$ (refer to Figure S13). The Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ for this robot has two factors. The trajectory $\mathbf{Q}$ starting at $\mathbf{q}_0$ has the goal configuration, $\overset{*}{\mathbf{q}}$ and is generated by a control policy $\mathrm{g}^{(Q)}$. The case where $\mathcal{B}(\mathbf{q}_{\text{first exit}}) = \mathcal{B}(\mathbf{q}_{\text{last entry}}) = \mathcal{B}_{f_h^\times}$. Figure S13 illustrates an example corresponding to this case. For cases with vertical turning points (Category 2 and Category 3), the configuration branch is identified with the horizontal component of the normal.

Algorithm 8. Track Cycle in Joint Space for Robots from Categories 1–3

Input: Joint configuration $\mathbf{q}$ and boundary normals $\mathbf{t}$

Determine boundary normals

Compute $\mathbf{t},\ \mathbf{t}_{\text{first exit}},\ \mathbf{t}_{\text{last entry}}$ at current, first exit, and last entry configurations

Category 1: Non-loop, non-intersecting singularities with only horizontal turning points

Let $\mathcal{B}_{\mathrm{f}_h^\times}$ denote the boundary region of such factors.

Case 1a: If $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{\mathrm{f}_h^\times} \cup \mathcal{B}_{q_3^\times}$:

If $\mathbf{q} \in \mathcal{B}_{q_2^\times}$:

$\mathbf{v} \leftarrow [0,\ -\mathbf{t}_{\text{last entry}}(q_3)]^T$

Else if $\mathbf{q} \in \mathcal{B}_{q_3^\times} \cup \mathcal{B}_{\mathrm{f}_h^\times},\ \mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) < 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3),\ 0]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) > 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

Case 1b: If $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{q_2^\times}$:

If $\mathbf{q} \in \mathcal{B}_{q_3^\times} \cup \mathcal{B}_{\mathrm{f}_h^\times}$:

$\mathbf{v} \leftarrow [-\mathbf{t}_{\text{last entry}}(q_2),\ 0]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3)]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) > 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}(q_3)]^T$

Category 2: Non-loop, non-intersecting singularities with no horizontal turning points

Let $\mathcal{B}_{\mathrm{f}^\times}$ denote the boundary region of the single factor.

Case 2a: If $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{\mathrm{f}^\times} \cup \mathcal{B}_{q_2^\times}$:

If $\mathbf{q} \in \mathcal{B}_{q_3^\times}$:

$\mathbf{v} \leftarrow [-\mathbf{t}_{\text{last entry}}(q_2),\ 0]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3)]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) > 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}(q_3)]^T$

Case 2b: If $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{q_3^\times}$:

If $\mathbf{q} \in \mathcal{B}_{q_2^\times} \cup \mathcal{B}_{\mathrm{f}^\times}$:

$\mathbf{v} \leftarrow [0,\ -\mathbf{t}_{\text{last entry}}(q_3)]^T$

Else if $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) < 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3),\ 0]^T$

Else if $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) > 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

Category 3: Non-loop, intersecting singularities

Let up to two factors be horizontal-type with region $\mathcal{B}_{f_h^\times}$, and one vertical-type with region $\mathcal{B}_{f_v^\times}$.

Case 3a: If $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{f_v^\times} \cup \mathcal{B}_{q_2^\times}$:

If $\mathbf{q} \in \mathcal{B}_{q_3^\times} \cup \mathcal{B}_{f_h^\times}$:

$\mathbf{v} \leftarrow [-\mathbf{t}_{\text{last entry}}(q_2),\ 0]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3)]^T$

Else if $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) > 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}(q_3)]^T$

Case 3b: If $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{f_h^\times} \cup \mathcal{B}_{q_3^\times}$:

If $\mathbf{q} \in \mathcal{B}_{q_2^\times} \cup \mathcal{B}_{f_v^\times}$:

$\mathbf{v} \leftarrow [0,\ -\mathbf{t}_{\text{last entry}}(q_3)]^T$

Else if $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) < 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2),\ 0]^T$

Else if $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) > 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

Return v

The trajectory generated by motion control law violates kinematic constraints at $\mathbf{q}_{\text{first exit}}$ and $\mathbf{q}_{\text{last entry}}$. The path between these two points is generated from the track cycle in Algorithm 8. As we $\mathcal{B}(\mathbf{q}_{\text{first exit}}) = \mathcal{B}_{f_1^\times}$, the velocity generated at this configuration is along the direction of $[\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2), 0]^T$, in this case generates a leftward velocity direction. Progressively, the joint configuration, $\mathbf{q}$, approaches $\mathcal{B}_{q_2^\times}$. When $\mathbf{q} \in \mathcal{B}_{q_2^\times}$, $\mathbf{v} = [0, -\mathbf{t}_{\text{last entry}}(q_2)]^T$ forces the joint velocity along positive q_2 axis directed parallel to $q_2^\times$ while progressively approaching $\mathbf{q} \in \mathcal{B}_{f_1^\times}$ such that $\mathbf{t}_{\mathbf{q}}.\mathbf{t}_{\text{last entry}} \geq 0$. This forces the joint velocity along the singularity branch towards $\mathbf{q}_{\text{last entry}}$ as

seen from the last row of the corresponding table in Algorithm 8. The overall behaviour encapuslated by $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ is thus

$$(\mathbf{q}_{\text{first exit}} \in \mathcal{B}_{f_1^\times}, \mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0) \rightarrow (\mathbf{q} \in \mathcal{B}_{q_2^\times})$$
$$\rightarrow (\mathbf{q} \in \mathcal{B}_{f_1^\times}, \mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) > 0) \rightarrow (\mathbf{q}_{\text{last entry}})$$

The track cycle $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ determines the behaviour to reach $\mathbf{q}_{\text{last entry}}$ from any point in any of the boundary regions in the current aspect. To model the joint velocity during the track cycle behaviour between $\mathbf{q}_{\text{first exit}}$ and $\mathbf{q}_{\text{last entry}}$, we introduce a track-cycle DS. The joint velocity in the $q_2 - q_3$ slice for a point $\mathbf{q}$ in the boundary $\mathcal{B}(\mathbf{q})$ is determined as

$$\dot{\mathbf{q}}(q_2, q_3) = \|\mathbf{q}_{\text{last entry}} - \mathbf{q}\| \frac{\mathbf{v}(q_2, q_3) \cdot \mathbf{t}^\perp}{\|\mathbf{v}\|} \mathbf{t}^\perp \tag{S10}$$

where $\mathbf{t}^\perp$ is the normal to $\mathbf{t}$. If $\mathbf{q}_{\text{last entry}}$ lies simultaneously in multiple near-boundary regions, any region (and associated boundary normal $\mathbf{t}_{\text{last entry}}$ can be used to determine the track cycle.

Algorithm 9. Joint Velocity for $3R$ Robot During Overlapping Boundary Regions

Input: Current configuration $\mathbf{q}$ in overlapping edge region

Output: Joint velocity $\dot{\mathbf{q}}$ to reach $\mathbf{q}_{\text{last entry}}$

Identify overlap region

$\mathbf{q} \in E_{1_2^\times} \cap E_{1_3^\times}$ or $\mathbf{q} \in E_{1_2^\times} \cap E_{\mathrm{f}_i^\times}$

Extract boundary edge set

$\{E(\mathbf{q})\} \leftarrow$ either $\{E_{1_2^\times}, E_{1_3^\times}\}$ or $\{E_{1_2^\times}, E_{\mathrm{f}_i^\times}\}$

For each edge in $\{E(\mathbf{q})\}$

Call Algorithm 3 $\Rightarrow \{\mathbf{t}\}$ (set of boundary normals)

Retrieve corresponding track-cycle velocities

$\{\mathbf{v}\} \leftarrow$ velocity candidates from transition logic $\mathcal{T}$

Determine ideal goal velocity direction

$\mathbf{v}_{\text{ideal}} \leftarrow \{\mathbf{v} \in \{\mathbf{v}\} \mid \mathbf{v} \cdot \mathbf{t} \geq 0,\ \forall\, \mathbf{t} \in \{\mathbf{t}\}\}$

Determine ideal boundary normal

$\mathbf{t}_{\text{ideal}} \leftarrow$ normal corresponding to $\mathbf{v}_{\text{ideal}}$

$\mathbf{t}_{\text{ideal}} \leftarrow \mathrm{argmin}_{\mathbf{t} \in \{\mathbf{t}\}} (\mathbf{t} \cdot \mathbf{v}_{\text{ideal}})$

Return projected joint velocity

$$\dot{\mathbf{q}} \leftarrow \|\mathbf{q}_{\text{last entry}} - \mathbf{q}\| \, \frac{\mathbf{v}_{\text{ideal}} \cdot \mathbf{t}_{\text{ideal}}{}^{\perp}}{\|\mathbf{v}\|} \, \mathbf{t}_{\text{ideal}}{}^{\perp}$$

Return $\dot{\mathbf{q}}$

The near-boundary control policy is globally asymptotically stable at $\mathbf{q}_{\text{last entry}}$ in the $q_2 - q_3$ slice. Consequently, the track cycle models the behaviour across all boundary regions, while approaching $\mathbf{q}_{\text{last entry}}$ in a stable manner. The method to determine the track cycles for robots from the remaining categories - Categories 4 through 6 - require different considerations and are presented at the end of this section.

When there are violations only in the $q_2 - q_3$ slice, the first joint's velocity is overlayed in accordance with the near-boundary control policy as

$$\begin{aligned} \dot{\mathbf{q}}(q_1) &= \mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}}) \\ &= \text{sign}(\mathbf{q}_{\text{last entry}}(q_1) - \mathbf{q}(q_1)) \cdot \|\mathbf{q}_{\text{last entry}} - \mathbf{q}\| \end{aligned} \tag{S11}$$

Equation S11 ensures that the robot converges at the $\mathbf{q}_{\text{last entry}}$ across all axis in the joint space, before proceeding with further replication towards $\overset{*}{\mathbf{q}}$. In the scenario where $\mathbf{q}_{\text{last entry}}(q_1)$ exceeds the first joint's limit, it is reset to the nearest permissible joint value i.e. $\overline{q_1} - \epsilon_q$ or $\underline{q_1} + \epsilon_q$.

Behaviour near first joint's boundary

We now extend the near-boundary control policy to joint configurations in the boundary of first joint's limits $\mathcal{B}_{q_1^{\times}}$, when that is the only type of violation. In such scenario, the velocity of the first joint is overlayed as

$$\dot{\mathbf{q}}(q_1) = 0$$

which prevents violations along first joint's limits. For the behaviour of second and third joints, the predicted path $\mathbf{Q}$ is used to determine the first joint configuration satisfying $q_1^{\times}$. This point $\mathbf{q}_{\text{target}} \in \mathcal{Q}$ is used to overlay the joint velocities as

$$\begin{aligned} \dot{\mathbf{q}}(q_2, q_3) &= \mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{target}}) \\ &= \text{sign}(\mathbf{q}_{\text{target}}(q_2, q_3) - \mathbf{q}(q_2, q_3)) \cdot \|\mathbf{q}_{\text{target}} - \mathbf{q}\| \end{aligned} \tag{S12}$$

Equation S12 ensures that the robot converges at the $\mathbf{q}_{\text{target}}$ across all axis in the joint space, before proceeding with further replication towards $\overset{*}{\mathbf{q}}$.

Overall behaviour for $3R$

The overall behaviour, resulting due to the control policy learned from demonstration $\mathrm{g}^{(\mathcal{Q})}$ for the feasible joint configurations of an aspect $\mathcal{A}$, as well as the near-boundary control policy $\mathrm{g}^{(\mathcal{B})}$ for configurations near the aspect's boundaries has been summarised in Table S1. It must be noted that it is not possible to have violation in both in the $q_2 - q_3$ slice, and along q_1 simultaneously since a violation in $q_2 - q_3$ slice and the resulting near boundary behaviour will prevent any further

The combination of the these policies ensures that kinematic constraints are embedded into the joint velocity during the replication of behaviour modelled from a demonstration.

When the initial joint configuration $\mathbf{q}$ is in a near-boundary region, the trajectory $\mathbf{Q}$ generated by $\mathrm{g}^{(\mathcal{Q})}$ starting at $\mathbf{q}$ and converging at $\overset{*}{\mathbf{q}}$, is used to determine the track cycle and subsequently the near-boundary behaviour $\mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$. The method for generating $\mathbf{Q}$ is elaborated in Section 12. Unlike point-by-point integration methods, our approach allows the full trajectory to be computed in a single step, enabling efficient and rapid validation.

Algorithms for track cycle

The method to determine the track cycle for robots from Categories 1 through 3 was presented in Algorithm 8. In this section, we present the remainder of algorithms to determine the track cycle for non-cuspidal $3R$ robots from Categories 4 through 6. The boundary normals $\mathbf{t}, \mathbf{t}_{\text{first exit}}, \mathbf{t}_{\text{last entry}}$ at the current joint configuration, first exit, and last entry configurations are determined using Algorithm 7 as the first step of determining the track cycle.

Robots from Category 4

There exists a single factor f in the expression of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ containing 2 branches of type $(1, 0)$, and at least one of the branches has finite vertical turning points The horizontal tuning points are represented as Q_h, and the vertical ones as Q_v. I_{SL} is the set of configurations at intersection between the singularity and joint limits in the $q_2 - q_3$ slice. It is

possible that the branches may intersect with $q_3^{\times}$, and more than four times with $q_2^{\times}$. In such cases, the horizontal or vertical turning points no longer lie on a continuous branch in the $q_2 - q_3$ slice. For instance in Figure S6, the branch looks like it has been split or separated by the joint limits. In such cases, we translate the $q_2 - q_3$ slice along q_2 and (or) q_3 axis such that the horizontal and vertical turning points on the branches lie along a continuous path. To achieve this, the relevant points on each branch Q_h, Q_v, I_{SL} are translated using Algorithm 20. To determine which of the two branches each of these joint configurations lie on, we use the criteria

$$B(\mathbf{q}) = \begin{cases} +1 & \text{if } \operatorname{sign}(\mathrm{f}(\mathbf{q} + [0, \epsilon_q]^T)) > 0 \\ -1 & \text{if } \operatorname{sign}(\mathrm{f}(\mathbf{q} + [0, \epsilon_q]^T)) < 0 \end{cases} \tag{S13}$$

Joint configurations from (Q_h, Q_v, I_{SL}) are next sorted two sets of joint configurations Q^+ and Q^- viz:

$Q^+ \leftarrow$ Configurations with $B = +1$ along the branch in ascending order of q_2 values, in order of occurrence

$Q^- \leftarrow$ Configurations with $B = -1$ along the branch in ascending order of q_2 values, in order of occurrence

Configurations from Q^+ and Q^- are translated back to the original $q_2 - q_3$ slice, in case Algorithm 20 was previously used. To specify the behaviour near constraints, we define bounding boxes B_i^+ for $i \in [1, 2 \cdots |Q^+| - 1]$, made using Q_i^+ and Q_{i+1}^+ as diagonal vertices along the first branch. For the second branch, B_i^- for $i \in [1, 2 \cdots |Q^-| - 1]$ are similarly defined. To identify which bounding box does a joint configuration $\mathbf{q} \in \mathcal{B}_{\mathrm{f}^{\times}}$, we calculate all the projections of $\mathbf{q}$ on f along q_2 and q_3, and identify the projection closest as $\mathbf{q}_{\text{proj}}$. The bounding box containing $\mathbf{q}_{\text{proj}}$ also holds $\mathbf{q}$. In case of multiple suitable boxes, anyone can be selected. To move near the branch rightward, from a box B_i to B_{i+1} on either branches, the velocity vector $\mathbf{v}$ gets two candidate values $[Q_{i+1}(q_2) - \mathbf{q}(q_2), 0]^T$ and $[0, Q_{i+1}(q_3) - \mathbf{q}(q_3)]^T$. The suitable velocity is later determined using Algorithm 9. For leftward traversal, we instead use Q_{i-1} to determine $\mathbf{v}$. Figure S14 illustrates the track cycle in $q_2 - q_3$ slice for a robot from Category 4. The method to determine the track cycle is presented in Algorithm 10.

Algorithm 10. Track Cycle in Joint Space for Robots from Category 4

Input: Current configuration $\mathbf{q}$, entry/exit points $\mathbf{q}_{\text{first exit}}$, $\mathbf{q}_{\text{last entry}}$

Determine branches for $\mathbf{q}_{\text{first exit}}$ and $\mathbf{q}_{\text{last entry}}$

Identify type of violation at both points

Case 1: $\mathbf{q}_{\text{last entry}} \in B_j^+$

If $\mathbf{q} \in \mathcal{B}_{q_2^\times}$:

$\mathbf{v} \leftarrow [0,\ I_{SL}^+(q_3) - I_{SL}^-(q_3)]^T$

If $\mathbf{q} \in B_k^+$:

If $k < j$: Perform rightward traversal

If $k > j$: Perform leftward traversal

If $k = j$: $\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

If $\mathbf{q} \in B^-$

If $[\mathbf{q}_{\text{last entry}} - \mathbf{q}_{\text{first exit}}](q_2) \geq 0$:

Rightward traversal

Else: Leftward traversal

If $\mathbf{q} \in \mathcal{B}_{q_3^\times}$:

If $\mathbf{q}$ does not lie between relevant I_{SL}^+ points:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2),\ 0]^T$

Else: Use B_k^+ that contains relevant I_{SL}^+

Case 2: $\mathbf{q}_{\text{last entry}} \in B_j^-$

If $\mathbf{q} \in \mathcal{B}_{q_2^\times}$:

$\mathbf{v} \leftarrow [0,\ I_{SL}^-(q_3) - I_{SL}^+(q_3)]^T$

If $\mathbf{q} \in B^+$:

If $[\mathbf{q}_{\text{last entry}} - \mathbf{q}_{\text{first exit}}](q_2) \geq 0$:

Rightward traversal

Else Leftward traversal

If $\mathbf{q} \in B_k^-$:

If $k < j$: Rightward traversal

If $k > j$: Leftward traversal

If $k = j$: $\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

If $\mathbf{q} \in \mathcal{B}_{q_3^\times}$:

If $\mathbf{q}$ does not lie between relevant I_{SL}^{-} (I_{SL}^{-} points with $q_3 = \mathbf{q}(q_3)$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2),\ 0]^T$

Else: Use B_k^{-} that contains relevant I_{SL}^{-}

Case 3: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{q_2^{\times}}$

If $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3)]^T$

Else:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}(q_3)]^T$

If $\mathbf{q} \in \mathcal{B}_{q_3^{\times}}$ or in B^{+} or B^{-}:

Use direction based on whether $\mathbf{q}_{\text{last entry}}(q_2) = \overline{q_2}$ or $\underline{q_2}$

(rightward or leftward traversal)

Case 4: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{q_3^{\times}}$

If $\mathbf{q}_{\text{last entry}}$ not between relevant I_{SL}^{-} or I_{SL}^{+}:

If $\mathbf{q} \in \mathcal{B}_{q_2^{\times}}$: $\mathbf{v} \leftarrow [0,\ -\mathbf{t}_{\text{last entry}}(q_3)]^T$

If $\mathbf{q} \in \mathcal{B}_{q_3^{\times}}$: $\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

If $\mathbf{q} \in \{B^{+}, B^{-}\}$ and $\Delta q_2 \geq 0$: rightward; else: leftward

If $\mathbf{q}_{\text{last entry}}$ lies between relevant I_{SL}^{+}:

If $\mathbf{q} \in B^{-}$ and $\Delta q_2 \geq 0$: rightward; else: leftward

If $\mathbf{q} \in B^{+}$: use velocity from B_k^{+} containing any I_{SL}^{+}

If $\mathbf{q}_{\text{last entry}}$ lies between relevant I_{SL}^{-}:

If $\mathbf{q} \in B^{+}$ and $\Delta q_2 \geq 0$: rightward; else: leftward

If $\mathbf{q} \in B^{-}$: use velocity from B_k^{+} containing any I_{SL}^{+}

Robots from Category 5

There will be single factor in the expression of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ containing a single branch of type $(0, 0)$. The branch will have only two horizontal turning points, and a finite number of vertical turning points. Q_h, Q_v, and I_{SL} represent the horizontal tuning points, vertical tuning

points, and intersection between the singularity and joint limits in the $q_2 - q_3$ slice respectively. Similar to Category 4, it is possible that the horizontal and(or) vertical turning points do not lie on a continuous branch in the $q_2 - q_3$ slice. In such cases, we translate the $q_2 - q_3$ slice along q_2 and (or) q_3 axis such that the horizontal and vertical turning points on the branches lie along a continuous path. To achieve this, the points Q_h, Q_v, I_{SL} are translated using Algorithm 21.

To specify the behaviour, it is required to differentiate between the interior of the loop created by the branch, and its exterior. To do so, we identify the horizontal turning point $\mathbf{q}_h$ with least q_3 value. A configuration $\mathbf{q}$ is inside the loop if $\text{sign}(\text{f}(\mathbf{q})) = \text{sign}(\text{f}(\mathbf{q_h} + [0, \epsilon_q]^T))$, and outside otherwise.

$Q \leftarrow \{Q_1, Q_2 \cdots Q_n\}$ ordered set of configurations from (Q_h, Q_v, I_{SL}) along the $(0, 0)$ branch in clockwise fashion, starting from the horizontal turning point with least q_3

We once again define bounding boxes B_i for $i \in [1, 2 \cdots |Q|]$, made using Q_i and $Q_{i+1 \bmod n}$ as diagonal vertices. These bounding boxes are adjacent with a common edge vertex between subsequent boxes, distributed along the loop like branch. To identify which bounding box does a joint configuration $\mathbf{q} \in \mathcal{B}_{\text{f}^\times}$, we once again calculate all the projections of $\mathbf{q}$ on f along q_2 and q_3, and identify the projection closest as $\mathbf{q}_{\text{proj}}$. The box containing $\mathbf{q}_{\text{proj}}$ also contains $\mathbf{q}$. To specify behaviour near the constraints, we define two directions - clockwise and anticlockwise as:

For a point inside the loop, in the bounding box B_i:

- $\mathbf{v}_C = \text{track}(Q_{i+1})$
- $\mathbf{v}_{\text{AC}} = \text{track}(Q_{i-1})$

In $\mathcal{B}_{q_2^\times}$:

- $\mathbf{v}_C = [0, -\text{sign}(q_2)]^T$
- $\mathbf{v}_{\text{AC}} = [0, \text{sign}(q_2)]^T$

In $\mathcal{B}_{q_3^\times}$

- $\mathbf{v}_C = [\text{sign}(q_3), 0]^T$
- $\mathbf{v}_{\text{AC}} = -[\text{sign}(q_3), 0]^T$

track(Q_i) returns two velocity vectors from $[0, Q_i(q_3) - \mathbf{q}(q_3)]^T$ and $[Q_i(q_2) - \mathbf{q}(q_2), 0]^T$ to traverse from $\mathbf{q}$ to $\mathbf{q}_{\text{last entry}}$, and the suitable velocity is later determined using Algorithm 9. Figure S15 illustrates the track cycle in $q_2 - q_3$ slice for a robot from Category 5. The method to determine the track cycle is presented in Algorithm 11.

Algorithm 11. Track cycle in joint space for robots from Category 5

Input: $\mathbf{q}_{\text{last entry}}$, $\mathbf{q}_{\text{first exit}}$

Determine if $\mathbf{q}_{\text{last entry}}$ (and $\mathbf{q}_{\text{first exit}}$) lie inside or outside the loop

Determine type of violation at these indices

Case 1: $\mathbf{q}_{\text{last entry}} \in B_j$ lies **inside** the loop

In case $\mathbf{q}$ lies near joint limit, identify the bounding box for $\mathbf{q}$ by using the nearest I_{SL} while traversing in cloc

Select bounding box containing the relevant I_{SL} vertex

If $\mathbf{q} \in B_k$:

If $k < j$: $\mathbf{v} \leftarrow \mathbf{v}_C$

If $k > j$: $\mathbf{v} \leftarrow \mathbf{v}_{\text{AC}}$

If $k = j$: $\mathbf{v} \leftarrow \text{track}(\mathbf{q}_{\text{last entry}})$

Case 2: $\mathbf{q}_{\text{last entry}}$ lies **outside** the loop and $I_{SL} = \emptyset$

Subcase: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{S^\times}$

If $\mathbf{q} \in B_k$:

$k < j$: $\mathbf{v} \leftarrow \mathbf{v}_C$

$k > j$: $\mathbf{v} \leftarrow \mathbf{v}_{\text{AC}}$

$k = j$: $\mathbf{v} \leftarrow \text{track}(\mathbf{q}_{\text{last entry}})$

If $\mathbf{q} \in \mathcal{B}_{q_2^\times}$:

If $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow (\mathbf{q}_{\text{first exit}}(q_3) - \mathbf{q}_{\text{last entry}}(q_3))\mathbf{v}_C$

Else:

$\mathbf{v} \leftarrow (\mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}(q_3))\mathbf{v}_C$

If $\mathbf{q} \in \mathcal{B}_{q_3^\times}$:

$\mathbf{v} \leftarrow (\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2))\mathbf{v}_C$

Subcase: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{q_2^\times}$

If $\mathbf{q} \in \mathcal{B}_{q_3^\times}$:

$\mathbf{v} \leftarrow [-\mathbf{t}_{\text{last entry}}(q_2),\ 0]^T$

If $\mathbf{t}(q_2)\mathbf{t}_{\text{last entry}}(q_2) < 0$:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}_{\text{first exit}}(q_3)]^T$

Else:

$\mathbf{v} \leftarrow [0,\ \mathbf{q}_{\text{last entry}}(q_3) - \mathbf{q}(q_3)]^T$

Subcase: $\mathbf{q}_{\text{last entry}} \in \mathcal{B}_{q_3^\times}$

If $\mathbf{q} \in \mathcal{B}_{q_2^\times}$:

$\mathbf{v} \leftarrow [0,\ -\mathbf{t}_{\text{last entry}}(q_3)]^T$

If $\mathbf{t}(q_3)\mathbf{t}_{\text{last entry}}(q_3) < 0$:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}_{\text{first exit}}(q_2),\ 0]^T$

Else:

$\mathbf{v} \leftarrow [\mathbf{q}_{\text{last entry}}(q_2) - \mathbf{q}(q_2),\ 0]^T$

Case 3: $\mathbf{q}_{\text{last entry}}$ lies **outside** the loop and $I_{SL} \neq \emptyset$

If $\mathbf{q} \in B_k \in \mathcal{B}_{S^\times}$:

If $k < j$: $\mathbf{v} \leftarrow \mathbf{v}_C$

If $k > j$: $\mathbf{v} \leftarrow \mathbf{v}_{\text{AC}}$

If $k = j$: $\mathbf{v} \leftarrow \text{track}(\mathbf{q}_{\text{last entry}})$

If $B_k \in \{\mathcal{B}_{q_2^\times}, \mathcal{B}_{q_3^\times}\}$:

If $k < j$: $\mathbf{v} \leftarrow \mathbf{v}_{\text{AC}}$

If $k > j$: $\mathbf{v} \leftarrow \mathbf{v}_C$

If $k = j$:

If $\mathbf{t} \cdot \mathbf{t}_{\text{last entry}} > 0$: $\mathbf{v} \leftarrow \mathbf{q}_{\text{last entry}} - \mathbf{q}$

If $\mathbf{t} \cdot \mathbf{t}_{\text{last entry}} = 0$: $\mathbf{v} \leftarrow -\mathbf{t}_{\text{last entry}}$

If $\mathbf{t} \cdot \mathbf{t}_{\text{last entry}} < 0$: revise $k \leftarrow n + 1$ and determine direction

Robots from Category 6

There will be two factors, one of which is of type $(0,0)$ with a single branch. The other factor that exist can only be of type $2(1,0)[\infty,0]$. The factor of type $(0,0)$ is represented using f_1, and f_2 represents the factor corresponding to the $(1,0)$ branches, Q_h, and Q_v represent the horizontal and vertical turning points associated with f_1, and I_{SL} represents the intersections between f_1 and $q_2^\times, q_3^\times$. I_{SS} represents the intersections between f_1 and f_2. $Q \leftarrow \{Q_1, Q_2 \cdots Q_n\}$ ordered set of configurations from $(Q_h, Q_v, I_{SL}, I_{SS})$ along the $(0,0)$ branch in clockwise fashion, starting from the horizontal turning point with least q_3. Using Q, bounding boxes are defined as before. The interior/exterior of the $(0,0)$ branch, and the method of determining the bounding box for a joint configuration $\mathbf{q} \in \mathcal{B}_{\mathrm{f}^\times}$ is identical to Appendix 10. $\mathcal{B}_{\mathrm{f}_2^\times}$ is the boundary region of f_2, and treated identical to $\mathcal{B}_{q_3^\times}$. The method to determine the track cycle is presented in Algorithm 12.

Algorithm 12. Track cycle in joint space for robots from Category 6

Input: $\mathbf{q}_{\text{last entry}}$

Output: Tangential joint velocity $\mathbf{v}$

If $\mathbf{q}_{\text{last entry}}$ lies inside the loop:

$\mathbf{v} \leftarrow$ Call Algorithm 11

Else if $\mathbf{q}_{\text{last entry}}$ lies outside the loop:

Let Q_k be the region containing $\mathbf{q}$

If $Q_k \in \mathcal{B}_{S^\times}$:

If $k > j$:

If I_{SS} lies clockwise between j and k: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{AC}}$

Else: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{C}}$

If $k < j$:

If I_{SS} lies clockwise between j and k: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{C}}$

Else: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{AC}}$

If $k = j$: $\mathbf{v} \leftarrow \mathrm{track}(\mathbf{q}_{\text{last entry}})$

Else if $Q_k \in \{\mathcal{B}_{q_2^\times}, \mathcal{B}_{q_3^\times}\}$:

If $k < j$:

If I_{SS} lies clockwise between j and k: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{C}}$

Else: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{AC}}$

If $k > j$:

If I_{SS} lies clockwise between j and k: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{AC}}$

Else: $\mathbf{v} \leftarrow \mathbf{v}_{\mathrm{C}}$

If $k = j$:

If $\mathbf{t} \cdot \mathbf{t}_{\text{last entry}} > 0$: $\mathbf{v} \leftarrow \mathbf{q}_{\text{last entry}} - \mathbf{q}$

If $\mathbf{t} \cdot \mathbf{t}_{\text{last entry}} = 0$: $\mathbf{v} \leftarrow -\mathbf{t}_{\text{last entry}}$

If $\mathbf{t} \cdot \mathbf{t}_{\text{last entry}} < 0$: revise $k \leftarrow n + 1$ and determine direction

---

Modelling the Robot Control Policy

So far, we have discussed the robot behaviour while considering a single demonstration in one of the aspects of the joint space. In this section, we consider the joint space behaviour resulting from multiple demonstrations in multiple aspects. We present the different elements of the framework that allows us to generalise the behaviour of a demonstrated task from multiple demonstrations in workspace workspace $\mathcal{X}$, to a robust and reactive control law across all the aspects in joint space $\mathcal{Q}$, while also embedding information about singularities and joint-limits

We begin by considering a set of K_Q demonstrations in joint space, denoted as $\{\mathbf{Q}\}$. These demonstrations are the joint space counterparts of the demonstrations in workspace $\mathcal{X}$. The proposed method first leverages the available $\{\mathbf{Q}\}$ to learn a preliminary behaviour in the joint space. Later in Section 21, the framework incrementally updates the preliminary behaviour in an offline manner, to embed additional knowledge about the task from workspace into the joint space.

Grouping joint demonstrations by aspects

While the workspace behaviour $\mathrm{g}^{(\mathcal{X})}$ had a single goal position workspace goal position $\overset{*}{\mathbf{x}}$, there exist I configurations in joint space $\mathcal{Q}$ that correspond to workspace goal position $\overset{*}{\mathbf{x}}$. This set of distinct joint configurations $\{\overset{*}{\mathbf{q}}_i \mid \mathrm{FK}(\overset{*}{\mathbf{q}}_i) = \overset{*}{\mathbf{x}}\}_{i=1}^{I}$, are the inverse kinematic solutions for workspace goal position $\overset{*}{\mathbf{x}}$. Each of $\overset{*}{\mathbf{q}}_i$ resides in a unique aspect $\mathcal{A}_i$. An individual goal joint configuration $\overset{*}{\mathbf{q}}_i$ may be known via the last point of the expert demonstration in joint space $\mathcal{Q}$, solved numerically or provided externally.

As there exists a single workspace goal position $\overset{*}{\mathbf{x}}$, each associated joint demonstration strictly terminates at one of the configurations in goal configurations $\{\overset{*}{\mathbf{q}}_i \mid \mathrm{FK}(\overset{*}{\mathbf{q}}_i) = \overset{*}{\mathbf{x}}\}_{i=1}^{I}$, wherein $\overset{*}{\mathbf{q}}_i \in \mathcal{A}_i$. Furthermore, as the workspace demonstrations $\{\mathbf{X}\}$ do not encounter any singular configuration, their joint space counterparts $\{\mathbf{Q}\}$ also remain free of singular configurations. Hence, if a joint demonstration $\mathbf{Q}$ terminates at $\overset{*}{\mathbf{q}}_i \in \mathcal{A}_i$, it can be said the entire trajectory $\mathbf{Q} \in \mathcal{A}_i$ only.

To proceed with behavioural modelling, all the joint demonstrations belonging to the same aspect $\mathcal{A}_i$ (and hence by extension having a common goal configuration $\overset{*}{\mathbf{q}}_i$) are grouped into a set $\{\mathbf{Q}\}_{\mathcal{A}_i}$. Thus, for a query joint configuration $\mathbf{q}$ that lies in aspect $\mathcal{A}(\mathbf{q})$, the relevant joint attractor fulfils $\mathcal{A}(\overset{*}{\mathbf{q}}_i) = \mathcal{A}(\mathbf{q})$. The grouping of joint demonstrations is performed on basis of the aspects

of the respective joint attractors

$$\{\mathbf{Q}\}_{\mathcal{A}_i} = \{\mathbf{Q}_j \in \mathcal{Q} \mid \mathcal{A}(\mathbf{Q}_j) = \mathcal{A}_i\}$$
$$\forall\, i \in \{1, \ldots, I\},\ \forall\, j \in \{1, \ldots, K_Q\}$$

A $2R$ robot always has exactly two aspects. The aspect of a non-singular query point $\mathbf{q}$ can be determined as $\mathcal{A}(\mathbf{q}) = \text{sign}(\det \mathbf{Q}(\mathbf{q}))$. On the other hand, a $3R$ robot may have upto eight aspects of varying nature which depends on the specific architecture of the robot. The aspect can be identified using the tuple composed of sign of each factor of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$. For instance, if there are two factors

$$\mathcal{A}(\mathbf{q}) = (\text{sign}(\text{f}_1(\mathbf{q})), \text{sign}(\text{f}_2(\mathbf{q}))) \tag{S14}$$

Figure S16 (B) illustrates the two aspects in the joint space of a $2R$ robot. Each aspect contains two joint trajectories corresponding to the two workspace demonstrations.

Validation of joint demonstrations

As presented in Section , it stands that some of the joint demonstrations, corresponding to the workspace demonstration, may be infeasible. Such trajectories must be identified to prevent acquiring infeasible behaviour.

To identify such trajectories, we perform two checks.

First check: For each joint space trajectory $\mathbf{Q}_i \in \{\mathbf{Q}\}$, check for joint limit violations along the trajectory

$$\exists \mathbf{q} = [q_1, q_2, q_3]^T \in \mathbf{Q}_i \text{ such that } q_1 \notin [\underline{q_1} + \epsilon_q, \overline{q_1} - \epsilon_q]$$

or $q_2 \notin [\underline{q_2} + \epsilon_q, \overline{q_2} - \epsilon_q]$ or $q_3 \notin [\underline{q_3} + \epsilon_q, \overline{q_3} - \epsilon_q]$.

Second check: Check for singularity violations in the trajectory

$$\exists \mathbf{q} \in \mathbf{Q}_i \text{ such that any for any factor } \text{f}_i \text{ of } \det\mathbf{J},$$

$|\,\text{f}_i(\mathbf{q}_t)| < \epsilon_S$ where $\epsilon_S > 0$ is the safety threshold around singular configurations (refer Section 5).

Any trajectory $\mathbf{Q}_i$ which fails either of the two checks has violated the kinematic constraints. Instead of discarding such trajectories, we trim the trajectories to only include the terminal violation

free portion of the trajectory that starts after the last violation in the trajectory and converges at the respective $\overset{*}{\mathbf{q}}$. In doing so, all the trajectories from $\{\mathbf{Q}\}$ are trimmed on an as-needed basis, and are hence validated to be free of infeasible configurations.

The second joint trajectory in Aspect 2 of Figure S16 (B) is trimmed due to violation of first joint's limit prior to learning.

Modelling aspect-specific behaviour

Previously (Section 1), a single demonstration was used to model aspect behaviour. We now extend this to multiple demonstrations within the same aspect. Consider the aspect-specific set of grouped K joint trajectories $\{\mathbf{Q}\}_{\mathcal{A}_i}$ for modelling the behaviour in $\mathcal{A}_i$. These trajectories have a common goal configuration $\overset{*}{\mathbf{q}}_i$ where $\mathcal{A}(\overset{*}{\mathbf{q}}) = \mathcal{A}_i$.

Since this section focuses on modelling a single aspect's behaviour, we hereby drop the subscript i to refer to a specific aspect in this Section. To model the aspect-specific joint behaviour that converges at $\overset{*}{\mathbf{q}} \in \mathcal{A}$, we embed each associated demonstration as a dynamical system by following the procedure in Section . The library of these K DSs - $\{\mathrm{g}_j^{(Q)} \leftrightarrow (\psi_j, \mathbf{k}_j)\}_{j=1}^{K}$ - embed the instances of demonstrated behavior in $\mathcal{A}$ to be replicated (refer Equation). Corresponding to each embedding, we partition the aspect $\mathcal{A}$ into K deformed conical regions $\{\mathcal{R}_i\}_{i=1}^{K}$, where each region shares a common vertex at $\overset{*}{\mathbf{q}}$. $\mathcal{R}_i$ represents a region of $\mathcal{A}$ that makes up the *neighbourhood* of the demonstration $\mathbf{Q}_i$, and is exclusively governed by associated DS embedding $\mathrm{g}_i^{(Q)}$

$$\dot{\mathbf{q}} = \mathrm{g}_i^{(Q)}(\mathbf{q}) \text{ if } \mathbf{q} \in \mathcal{R}_i$$

The overall partitioning of aspect $\mathcal{A}$ follows

$$\mathcal{A} = \bigcup_{i=1}^{K} \mathcal{R}_i, \quad \mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset \text{ for some } i \neq j. \tag{S15}$$

There may exist overlap between some or all of the K different partitions. To disambiguate the selection of relevant DS in such cases, we assign a priority $P(\mathcal{R}_i) = i$ to each $\mathcal{R}_i$ based on the chronology of the demonstration that corresponds to the relevant embedding $\mathrm{g}_i^{(Q)}$. This makes the first demonstration of the least priority, and subsequent demonstrations get assigned progressively higher priorities. For any point $\mathbf{q} \in \mathcal{A}$, the assigned region $\mathcal{R}(\mathbf{q})$ is determined by:

$$R(\mathbf{q}) = \{\mathcal{R}_i \mid \mathbf{q} \in \mathcal{R}_i, P(\mathcal{R}_i) = \max_{\mathcal{R}_j, \mathbf{q} \in \mathcal{R}_j} P(\mathcal{R}_j)\} \tag{S16}$$

This formulation ensures that if multiple regions overlap at a point $\mathbf{q}$, the region (and by extension the associated DS embedding $\mathrm{g}_i^{(Q)}$) with the highest priority takes precedence. The velocity at a point $\mathbf{q} \in \mathcal{A}$ is determined as

$$\dot{\mathbf{q}} = \mathrm{g}_i^{(Q)}(\mathbf{q}) \text{ where } i = \arg\max\{P(\mathcal{R}_i) \mid \mathbf{q} \in \mathcal{R}_i\} \tag{S17}$$

It must be noted that the priority order may also be composed by the expert, or even randomly allocated, especially if the chronology of demonstrations is unknown, or the demonstrations are provided simultaneously.

Since all DS embeddings are globally asymptotically stable (Equation S5) and share a common attractor $\overset{*}{\mathbf{q}}$, there exists at least one highest-priority embedding $\mathrm{g}_i^{(Q)}$ for all $\mathbf{q} \in \mathcal{A}$ ensuring convergence at the attractor

$$P(\mathcal{R}(\mathbf{q})) = \max_{1 \le i \le K} P(\mathcal{R}_i) \implies$$

$$\lim_{t\to\infty} \mathbf{q} = \lim_{t\to\infty} \psi_i\big(\psi_i^{-1}(\mathbf{q})\big) \tag{S18}$$

$$= \psi_i\big(\lim_{t\to\infty} \psi_i^{-1}(\mathbf{q})\big) \tag{S19}$$

$$= \psi_i(\mathbf{1}_n^T) = \overset{*}{\mathbf{q}}, \tag{S20}$$

Partitioning in the latent space

The neighbourhood of a demonstration $\mathbf{Q}_i$ in aspect $\mathcal{A}$, given by region $\mathcal{R}_i$, is smoothly mapped to a region $\mathcal{L}_i = \psi_i^{-1}(\mathcal{R}_i)$ around the latent embedding of the demonstration $\psi_i^{-1}(\mathbf{Q}_i)$ in latent space $\sqcap$. As the topology and differentiable structure of the neighbourhood region is preserved between $\mathcal{R}_i \subset \mathcal{A}$ and $\mathcal{L}_i \subset \mathcal{U}$, we can instead define the neighbourhood in latent space $\sqcap$. Doing so allows us to leverage the (quasi)linear structure in latent space $\sqcap$, to determine the non-linear neighbourhood region in joint space $\mathcal{Q}$. In latent space $\sqcap$, we define the neighbourhood $\mathcal{L}_i$ as a straight volume cone with an opening angle λ_i, whose axis is aligned with the vector $\mathbf{v} = \overset{*}{\mathbf{u}} - \mathbf{0}$ along the latent embedding of the demonstration, and its vertex is at $\overset{*}{\mathbf{u}}$. Here, $\overset{*}{\mathbf{u}} = \mathbf{1}_n^T$ is the latent embedding of $\overset{*}{\mathbf{q}} \in \mathcal{A}$ and $\mathbf{0}$ is the latent embedding of the first point in demonstration $\mathbf{Q}_i$. The cone is defined as the collection of joint space points

$$\mathcal{L}_i = \{\angle(\overset{*}{\mathbf{u}} - \psi_i^{-1}(\mathbf{q}), \mathbf{v}) \le \lambda_i\} \; \forall \; \mathbf{q} \in \mathcal{A} \tag{S21}$$

and represents the set of all points in latent space $\sqcap$ for DS embedding $\mathrm{g}_i^{(Q)}$, for which the angle between the direction $\psi_i^{-1}(\mathbf{q}) - \overset{*}{\mathbf{q}}$ and the axis $\mathbf{v}$ does not exceed λ_i. The latent embedding of the demonstration $\mathbf{Q}_i$ lies along the axis of this *cone of influence*, ensuring that the cone encapsulates a neighbourhood around the demonstration in latent space $\sqcap$. Figure S16 (C) illustrates the cones of influence, in the latent space associated with each DS embedding, for every aspect. In $\mathcal{A}$, $\mathcal{R}_i = \psi_i(\mathcal{L}_i)$ defines the neighbourhood. Consequently, Equation S17 can be reformulated as:

$$\dot{\mathbf{q}} = \mathrm{g}_i^{(Q)}(\mathbf{q}),\ i = \arg\max\{P(\mathcal{L}_i) \mid \psi_i^{-1}(\mathbf{q}) \in \mathcal{L}_i\} \tag{S22}$$

Calculating the partitions

Each cone $\mathcal{L}_i$ is uniquely parametrised by its opening angle $\lambda_i \in [\epsilon_\lambda, \pi]$, where ϵ_λ is the minimal angular value (in radians) for the opening angle. Higher the opening angle, wider the cone, and hence more region in latent space $\sqcap$ and joint space $\mathcal{Q}$ which is governed by $\mathrm{g}_i^{(Q)}$. As a result, these angles determine the overall influence of each of the learnt embeddings $\mathrm{g}_i^{(Q)}$ over the resulting behaviour in joint space. The first (and lowest priority) embedding is assigned a cone angle of π, effectively spanning the entire aspect. As more (and higher priority) embeddings are available, they get assigned progressively lower values. Maximising the neighbourhood region of each embedding, will lead to a better overall generalisation of the expert demonstrations in $\mathcal{A}$. However, it is also crucial to minimise the overlap among the partitions $\mathcal{R}_i$, to prevent an unrelated higher priority embedding $\mathrm{g}_H^{(Q)}$ from incorrectly overlaying the the desired dynamics which were to be ideally generated using a lower priority embedding $\mathrm{g}_L^{(Q)}$. To identify and subsequently eliminate such overlaps, we measure the influence of a DS learnt using a higher priority demonstration $\mathrm{g}_H^{(Q)}$ over a lower priority demonstration $\mathbf{Q}_L \in \mathcal{A}$. For a faithful behaviour replication, at least no joint configuration from $\mathbf{Q}_l$ should lie in the influence cone of $\mathrm{g}_H^{(Q)}$. Thus, the cone angle for $\mathrm{g}_H^{(Q)}$ is set as the maximum permissible value such that no joint configurations from a lower priority demonstration $\mathbf{Q}_l$ lie in it.

To do so, we define the set of projection angles $\Theta_H(\mathbf{Q}_l)$ for embedding $\mathrm{g}_H^{(Q)}$ as:

$$\Theta_H(\mathbf{Q}_l) = \{\angle(\psi_H^{-1}(\mathbf{q}_j) - \mathbf{1}_n^T, \mathbf{1}_n^T - \mathbf{0}_n^T)\}_{j=1}^{M_l} \tag{S23}$$

where M_l is the number of demonstrated points in $\mathbf{Q}_l$, and $\psi_H^{-1}(\mathbf{q}_j)\ \forall\ \mathbf{q}_j \in \mathbf{Q}_l$ denotes the latent embedding of $\mathbf{Q}_l$ generated in the latent space associated with $\mathrm{g}_H^{(Q)}$. Furthermore, $\Theta_H(\mathbf{Q}_l)$ is

the set of the angles subtended by the vectors between the aforementioned latent embedding of demonstrated points $\psi_i^{-1}(\mathbf{Q}_l)$ and latent space's origin, and the vector between the origin and the attractor $(\mathbf{1}_n^T)$ for some lower priority demonstration $\mathbf{Q}_l$. Points in $\mathbf{Q}_l$ that have higher projection angles represent more significant deviation of the demonstration, in contrast to those with lower projection angles.

Assuming ascending order of priority based on chronology of demonstrations, the opening angles $\lambda_i, i \in \{1, 2 \cdots K\}$ that prevent a lower priority demonstration being overlayed by a higher priority demonstrations DS can be determined as:

$$\begin{aligned} &\lambda_1 = \pi \text{ (least priority demonstration for entire aspect)} \\ &\lambda_i = \min(\{\Theta_i(\mathbf{Q}_l)\}_{l=1}^{i-1}) \text{ for } 2 \leq i \leq K \\ &\text{if } \lambda_i < \epsilon_\lambda, \quad \lambda_i \leftarrow \epsilon_\lambda \end{aligned} \tag{S24}$$

The set of these opening angles, collectively referred to as $\Lambda = \{\lambda_i\}_{i=1}^K$, ensures that the cone of influence of some high priority demonstration $\mathcal{L}_H$ does not overlap with the latent embedding of any of the lower priority demonstrations $\{\psi_H^{-1}(\mathbf{Q}_l)\}_{l=1}^{H-1}$. The cone angle for the first demonstration is set to π and encompasses the entire aspect. However, as soon as a second demonstration (with higher priority) is available, a region of the aspect is now governed by this second demonstration's embedded DS. The cone angles get updated as new demonstrations become available (refer to Section 12). Enforcing the lower bound of ϵ_λ on the cone angles ensures that in case there are intersections between two distinct demonstrations (in workspace or their joint space counterparts), the cone angle in Equation S24 does not assume a negligible value. If $\lambda_i \approx 0$ for demonstration $\mathbf{Q}_i$, the learnt DS embedding would have no influence over the joint space behaviour, except for a very narrow volume of joint space around $\mathbf{Q}_i$. Having a lower bound of $\epsilon_\lambda > 0$ prevents this. Though this may lead to a higher priority DS embedding abruptly switching the dynamics of the trajectory being generated by a lower priority DS if left unchecked, the extension to prevent this during robot execution is presented in Section 12.

The composition of the priority ordering of the embeddings P, the opening angles Λ, and the learnt DS embeddings $\{\mathrm{g}_i^{(Q)}\}_{i=1}^K$, comprehensively describe the behaviour in the aspect $\mathcal{A}$ of the

joint space $\mathrm{B}^{\mathcal{A}}$ as

$$\begin{aligned}
\mathrm{B}^{(\mathcal{A})} &= \{P, \Lambda, \{\mathrm{g}_i^{(Q)} \leftrightarrow (\psi_i, \mathbf{k}_i)\}\} \\
\dot{\mathbf{q}} &= \mathrm{B}^{(\mathcal{A})}(\mathbf{q}) = \mathrm{g}_i^{(Q)}(\mathbf{q}) \text{ where} \\
R(\mathbf{q}) &= \mathcal{R}_i,\ P(\mathcal{R}_i) = \max_{1 \le j \le K} P(R_j)
\end{aligned} \tag{S25}$$

Incremental learning

When a new demonstration $\mathbf{Q}_{K+1}$ becomes available, only the corresponding embedding $(\psi_{K+1}, \mathbf{k}_{K+1})$ is learned and the list of embeddings gets updated. The set of opening angles Λ are also iteratively modified to avoid overlap with the new demonstration

$$\begin{aligned}
\lambda_i &\leftarrow \min(\lambda_i, \Theta_i(\mathcal{Y}_{K+1}))\ \forall\ i \in \{1, 2 \cdots K-1\} \\
\lambda_K &\leftarrow \min(\Theta^K(\mathcal{Y}_{K+1})) \\
\lambda_{K+1} &\leftarrow \frac{1}{K} \sum_{i=1}^{K} \lambda_i
\end{aligned} \tag{S26}$$

Lastly, the priority order of the associated partition is updated $P(L_{K+1}) = K + 1$

Overall joint space behaviour

Post sorting all the joint demonstrations, each set aspect-specific set of demonstrations $\{\mathbf{Q}\}_{A_i}$ is used to learn the respective aspect-specific behaviour $\mathrm{B}^{(\mathcal{A}_i)}$ (refer Equation S25) Thus, for a query joint configuration $\mathbf{q}$, the dynamics are characterised by aspect containing $\mathbf{q}$. The aspect-specific behaviours collectively describe the behaviour in joint space $\mathcal{Q}$ hereby referred to as $\mathrm{B}^{(\mathcal{Q})} = \{\mathrm{B}^{(\mathcal{A}_1)}, \mathrm{B}^{(\mathcal{A}_2)} \cdots \mathrm{B}^{(\mathcal{A}_{N_{\mathcal{A}}})}\}$ where $N_{\mathcal{A}}$ is the number of aspects. For a $2R$ robot, there will be upto two functionals in $\mathrm{B}^{(\mathcal{Q})}$, whereas for a non-cuspidal $3R$ robot, there can be upto eight. Figure S16 illustrates the process of modelling the behaviour from multiple demonstrations, both in the workspace and the joint space. The workspace behaviour $\mathrm{B}^{(\mathcal{X})}$ is learnt using two demonstrations. In the joint space, each of the two aspects also have two demonstrations each. The aspect specific behaviours $\mathrm{B}^{(\mathcal{A}_1)}$, and $\mathrm{B}^{(\mathcal{A}_2)}$ are learnt using the two demonstrations in each aspect. The corresponding cones of influence in the latent space, both for the workspace behaviour and aspect-specific behaviours, correspond to regions in workspace and joint space where a specific demonstration influences the behaviour.

Certified execution

To generate the trajectory from a query joint configuration $\mathbf{q}$, it is crucial to first determine the aspect which $\mathbf{q}$ belongs to and then utilise the relevant aspect-specific behaviour $\mathrm{B}^{(\mathcal{A}_i)}$. This is because $\mathrm{B}^{(Q)}$ is a multi-attractor system where the attractors or goal configurations are given by $\{\overset{*}{\mathbf{q}}_i \mid \mathrm{FK}(\overset{*}{\mathbf{q}}_i) = \overset{*}{\mathbf{x}}\}_{i=1}^{\mathcal{I}}$.

To formulate the selection process, we leverage the fact that each $\mathrm{B}^{(\mathcal{A}_i)}$ has a specific goal configuration $\overset{*}{\mathbf{q}}_i$ associated with it. Thus, each aspect specific behaviour can be identified via the goal joint space configuration it converges at. Now, instead of determining the aspect of query configuration $A(\mathbf{q})$, we can simply check which goal configuration $\overset{*}{\mathbf{q}}_i$ belongs to the same aspect as $\mathbf{q}$.

Connectivity analysis

Establishing a *connection* between the query and goal configuration without violating any aspect boundary uniquely determines the aspect-specific behaviour to choose. Recently, roadmap algorithms were proposed for connectivity analysis (*46*), which become challenging as they use real algebraic formulation and the rational parametrization of trigonometric terms at π radians is singular. Though this algorithm can be extended to consider joint limits, no work has been reported regarding it. In Algorithm 14, we propose a certified connectivity check algorithm that leverages the properties of singularities to determine the connectivity of the query and goal joint configurations, while also considering joint limits. Here, certified algorithm means an algorithm that provides worst-case and beyond worst-case performance guarantees (*47*). To assess the connectivity between query and goal, we consider each factor f_i (upto three) of the expression of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$ independently. Each factor is a branch of the singularity, and a boolean *connectivity* with respect this factor between the query and goal is determined. If connectivity via each factor can be established, we can conclude that the goal and query are connected, and hence belong to a common aspect. The different types of factors, that can arise for non-cuspidal $3R$ robots (covered by Category 1 through Category 6) as discussed in Section 4 can be characterised by the description of their branches (using Algorithm 1). A factor can have branches of one of the five types (refer to Table S2) viz;

1. $(1,0)[\geq 2,0]$ (e.g. Category 1 (row (A)), Category 3 (rows (F) and (H)), and Category 5 in Figure S3)

2. $(0,1)[0,*]$ (e.g. Category 2 (row (B), and Category 3 (row (F)) in Figure S3)

3. $(0,0)[2,\geq 2]$ (e.g. Category 4 (row (C)), and Category 5 (row (G)) in Figure S3)

4. $(1,0)[2,>0]$ (refer to Category 6 (row (D)) in Figure S3)

5. $(>0,1)[0,*]$ (refer to Category 2 (row (E)), and Category 3 (row (H)) in Figure S3)

Case (i): As two joint configurations in the same aspect should have the same sign of f_i, the connectivity algorithm verifies the signature of $\mathrm{f}_i(\mathbf{q}_A)$ and $\mathrm{f}_i(\mathbf{q}_B)$. Later, to determine the connectivity between two joint configurations $\mathbf{q}_A$ and $\mathbf{q}_B$ with the same sign of f_i of robots belonging to the first case, we consider the paths along the q_2 generator that passes through each horizontal turning point. If we have a nongeneric branch, the branch itself is the path we consider. As joint limits are linear paths, they are also easily included as a path for consideration. Both $\mathbf{q}_A$ and $\mathbf{q}_B$ are projected on each path. Let $\mathbf{p}_A(q_3)$ be the value of q_3 where the f_i value at $\mathbf{q}_A$ and its projection on the path through q_3 match. If the number of common q_3 values for $\mathbf{p}_A$ and $\mathbf{p}_B$ is non zero, then $\mathbf{q}_A$ and $\mathbf{q}_B$ are connected as presented in Algorithm 14 and illustration of an example case is shown in Figure S17.

Case (ii): When factors encircle the q_3 generator, the analysis differs as now we can have more than two vertical turning points. In such cases, we check the intersection of the path, $q_3 = \mathbf{q}_A(q_3)$ with the singularity as well as the joint limits, and verify the branches of the neighborhood points of intersection (say $\mathbf{q}_{A1}$ and $\mathbf{q}_{A2}$ for $\mathbf{q}_A$). As there can exist at most two $(1,0)$ branches, we can identify them with the direction of the gradient of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$. The connectivity of the query and goal configuration is then decided by investigating the type of branches on which $\mathbf{q}_{A1}, \mathbf{q}_{A2}, \mathbf{q}_{B1}$ and $\mathbf{q}_{B2}$ lie on. The connectivity algorithm for this case is discussed in Algorithm 16, and illustration of an example case is shown in Figure S19.

Case (iii): For singularities with homotopy case $(0,0)$, we need to consider if the configurations are connected even after being split by joint limits. For this purpose, we consider a range of joint

space with length 2π where the loop is not split. The horizontal and vertical turning points, the intersections of singularities with joint limit and the configurations in query are all translated along q_2 and q_3 to lie on or inside the loop in this new range. Once the loop closure is done, the connectivity of query configurations is investigated based on the shift required to place the query configurations inside the loop. The algorithm for connectivity check of such singularities is presented in Algorithm 17, and illustration of an example case is shown in Figure S20.

Case (iv): For singularities with homotopy case $(1, 0)[2, a > 0]$, we need to consider if the configurations are connected even after being split by joint limits and later using an algorithm similar to Case (i) to decide connectivity. In this case, the knowledge of f_i is leveraged to know that there can be at most one branch that is split by q_2 joint limits as well as q_3 joint limits. The algorithm for connectivity check of such singularities is presented in Algorithm 18, and illustration of an example case is shown in Figure S18.

Case (v): For singularities with homotopy case $(1, 1)$ and $(2, 1)$, we leverage the fact that the singularities encircle both q_2 and q_3 generators. This implies that each aspect is necessarily bounded by singularities as well as joint limits. The joint limits for each aspect are unique, and thus, the range of joint limits is used to identify the aspect and, thus, the connectivity between two configurations. The algorithm for connectivity check of such singularities is presented in Algorithm 19, and illustration of an example case is shown in Figure S21.

The singularities for all non-cuspidal $3R$ robots, covered by Category 1 through Category 6, are a combination of the connectivity check applied to each type of factor occurring in the singularity. In doing so, the proposed framework can verify connectivity among the joint configurations in a deterministic manner with a constant time complexity, as the maximum number of steps in any algorithm is always finitely bounded. These properties of the connectivity check allow the query to be allocated a finite time and computation resource, which are important considerations for any real-time behavior.

The connectivity checks for the factor classes (refer Table S3 are presented at the end of this section.

Trajectory generation

The steps to generating the trajectory from a query joint configuration $\mathbf{q}$, while not violating the kinematic constraints, can be summarised as follows:

1. Determine suitable aspect-specific behaviour $\mathrm{B}^{(\mathcal{A})}$ from the known set of behaviours $\mathrm{B}^{(Q)} = \{\mathrm{B}^{(\mathcal{A}_1)}, \mathrm{B}^{(\mathcal{A}_2)} \cdots \mathrm{B}^{(\mathcal{A}_{N_\mathcal{A}})}\}$

2. Determine suitable joint DS embedding $\mathrm{g}^{(Q)}$ from the embeddings in aspect-behaviour $\mathrm{B}^{(\mathcal{A})} = \{P, \Lambda, \{\mathrm{g}_i^{(Q)} \leftrightarrow (\psi_i, k_i)\}\}$

3. Formulate near-boundary behaviour near violation boundaries if and when needed

We begin by identifying the correct goal configuration, and the aspect-specific behaviour $\mathrm{B}^{(\mathcal{A})}$ to be used, using the relevant connectivity check (refer to Table S3). Next, the relevant DS embedding $\mathrm{g}^{(Q)}$ is picked on the basis of the priority P and the cones of influence Λ using Equation S22.

Note that when determining the relevant embedding, we measure the angle subtended between the vectors $\overset{*}{\mathbf{u}} - \psi_i^{-1}(\mathbf{q})$ and $\overset{*}{\mathbf{u}} - \mathbf{0}$ (Equation S21) to check which embeddings' cones of influence does $\mathbf{q}$ reside in, and then promptly select the highest priority embeddings from amongst them. However, if there was an intersection between the distinct user demonstrations, merely prioritising the highest priority embedding may lead to an abrupt switching in the dynamics at the point of intersection. To avoid such situations, if the angle subtended for the suitable DS embeddings is lower than the minimal cone angle ϵ_λ, we retain the embedding that was being used instead of simply selecting the highest priority embedding.

$\mathrm{B}^{(Q)}$ models the behaviour in each aspect of joint space Q, such that stable convergence at the relevant attractor $\overset{*}{\mathbf{q}}$ is guaranteed. The learnt DS embeddings are globally asymptotically stable, and invariant to spatial or temporal perturbations. However, if due to a perturbation or otherwise, the instantaneous joint configuration $\mathbf{q}$ is detected to be in a near-boundary region, the near-boundary control law $\mathrm{g}^{(\mathcal{B})}$ (Table S1) is formulated and overlays the learnt behaviour. To determine the near-boundary behaviour, the projected trajectory $\mathbf{Q}$ starting from $\mathbf{q}$, encountering violations and converging at the goal configuration, is required to determine the $\mathbf{q}_{\text{last entry}}$ and $\mathbf{q}_{\text{first exit}}$ configurations in the current aspect.

Oneshot trajectory generation for violation detection

Utilising the properties of the latent embedding and the learnt diffeomorphic mapping, permits the entire trajectory $\mathbf{Q}$ to be evaluated in batch, without recursive integration. First, the latent embedding for the projected trajectory $\mathbf{U}$ is generated using a predefined temporal formulation. Next, the diffeomorphic transformation ψ is used to transform the embedding into the joint space trajectory $\mathbf{Q}$ for use in violation detection.

For an initial point $\mathbf{q}$, the identified DS embedding $\mathrm{g}^{(Q)} \leftrightarrow (\psi, \mathbf{k})$ provides

$$\begin{aligned} \mathbf{u}_{\text{start}} &= \psi^{-1}(\mathbf{q}) \\ \acute{\mathbf{u}}_{\text{start}} &= \mathbf{1} + \|\mathbf{1} - \mathbf{u}_{\text{start}}\| \frac{\mathbf{1} - \mathbf{0}}{\|\mathbf{1} - \mathbf{0}\|} \end{aligned} \tag{S27}$$

where $\mathbf{u}_{\text{start}}$ is the latent embedding of $\mathbf{q}$, and $\acute{\mathbf{u}}_{\text{start}}$ is its projection along the latent embedding of the original demonstration. The demonstration's embedding is a quasilinear distribution along the vector from $\mathbf{0}$ (start) to $\mathbf{1}$ (goal). From Appendix H (*32*), when $\|\mathbf{1} - \mathbf{u}_{\text{start}}\| \leq \|\mathbf{1}\|$, the latent embedding of the original demonstration follows

$$\mathbf{u}_t = \sin(\mathbf{k}t) \tag{S28}$$

The original formulation represents the constant $\mathbf{k} \in \mathrm{R}_+^n$ as $\dfrac{\mathbf{a}}{\Delta t}$ where $\mathbf{a} \in \mathrm{R}_+^n$, and $\Delta t > 0$ is the sampling interval associated with the demonstration.

By using $\acute{\mathbf{u}}_{\text{start}}$ and the temporal formulation of original demonstration's latent embedding (Equation S28), we can estimate the latent embedding between $\mathbf{u}_{\text{start}}$ and $\mathbf{1}$

$$\begin{aligned} t_0 &= \frac{\arcsin(\acute{\mathbf{u}}_{\text{start}})}{\mathbf{k}} \\ \acute{\mathbf{u}}_i &= \sin(\mathbf{k}(t_0 + i\Delta t)) \text{ for } i = \{0, 1, \cdots \frac{T}{\Delta t}\} \\ \mathbf{U} &= \{\mathbf{1} + \frac{\mathbf{u}_{\text{start}} - \mathbf{1}}{\|\mathbf{u}_{\text{start}} - \mathbf{1}\|} \|\acute{\mathbf{u}}_i - \mathbf{1}\|\}_{i=0}^{T/\Delta t} \end{aligned} \tag{S29}$$

where T is the total duration of the original demonstration, Δt is the sampling frequency, and $\mathbf{U}$ is the latent embedding of the projected trajectory.

In situations where $\|\mathbf{1} - \mathbf{u}_{\text{start}}\| > \|1\|$, we first identify the point $\mathbf{u}_0$ along the segment between $\mathbf{u}_{\text{start}}$ and $\|\mathbf{1}\|$, such that $\|\mathbf{1} - \mathbf{u}_0\| = \|\mathbf{1}\|$

$$\mathbf{u}_0 = \mathbf{1} + \frac{\mathbf{u}_{\text{start}} - \mathbf{1}}{\|\mathbf{u}_{\text{start}} - \mathbf{1}\|} \|\mathbf{1}\| \tag{S30}$$

The first portion of the latent embedding of the projected path $\mathbf{U}$ is uniformly distributed along the linear segment between $\mathbf{u}_{\text{start}}$ and $\mathbf{u}_0$. The remainder of embedding between $\mathbf{u}_0$ and the goal $\|\mathbf{1}\|$ is generated using Equation S29.

Finally, the joint space trajectory $\mathbf{Q}$ is generated using diffeomorphic transformation ψ as

$$\mathbf{Q} = \psi(\mathbf{U})$$

The resulting track cycle $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ now maps the trajectory between $\mathbf{q}_{\text{first exit}}$ and $\mathbf{q}_{\text{last entry}}$ from $\mathbf{Q}$. Once in the vicinity of $\mathbf{q}_{\text{last entry}}$, the $\mathrm{B}^{(Q)}$ once again resumes control and eventually converges at $\overset{*}{\mathbf{q}}$

Resulting behaviour

The hybrid approach leveraging both $\mathrm{B}^{(Q)}$ and the near-boundary behaviour $\mathrm{C}(\mathbf{q}_{\text{last entry}})$ when required, ensures stable convergence at relevant attractor configuration, being robust to perturbations without trying to overshoot the joint limits or cross a singularity. The control law is presented in Algorithm 13.

Algorithm 13. Joint Velocity at Query Configuration q

Input: Query joint configuration $\mathbf{q}$

Output: Joint velocity $\dot{\mathbf{q}}$

Set of goal configurations:

$\{\overset{*}{\mathbf{q}}\} \leftarrow \{\overset{*}{\mathbf{q}}_1, \overset{*}{\mathbf{q}}_2, \ldots, \overset{*}{\mathbf{q}}_{N_{\mathcal{A}}}\}$

Determine aspect:

Identify $\mathcal{A}$ such that $\mathcal{A} \leftarrow \text{connectivity}(\mathbf{q}, \overset{*}{\mathbf{q}}_i)$ for all $\overset{*}{\mathbf{q}}_i \in \{\overset{*}{\mathbf{q}}\}$

If $\mathbf{q}$ is not connected to any $\overset{*}{\mathbf{q}}_i$:

Do not initiate motion

Else: Identify corresponding aspect-specific behaviour $\mathrm{B}^{(\mathcal{A})}$

Identify relevant embedding in $\mathrm{B}^{(\mathcal{A})}$**:**

$\mathrm{B}^{(\mathcal{A})} = \{P, \Lambda, \{\mathrm{g}_i^{(Q)} \leftrightarrow (\psi_i, \mathbf{k}_i)\}\}$

$\mathrm{g}^{(Q)} \leftarrow \mathrm{g}_i^{(Q)}(\mathbf{q})$,

where $i = \arg\max\{P(\mathcal{L}_i) \mid \psi_i^{-1}(\mathbf{q}) \in \mathcal{L}_i\}$

If $\mathbf{q}$ is not in any near-boundary region:

Use the learned behaviour $g^{(Q)}$

Return $\dot{\mathbf{q}} = g^{(Q)}(\mathbf{q})$

Else: Overlay with near-boundary control policy

If track cycle $C(\mathbf{q}_{\text{last entry}})$ is not determined:

$\mathbf{Q} \leftarrow$ Oneshot formulation (Sec. 12)

Determine $C(\mathbf{q}_{\text{last entry}})$ (Table S3)

Formulate $g^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$

Note: $C(\mathbf{q}_{\text{last entry}})$ is computed only once per region entry

and reused until convergence at $\mathbf{q}_{\text{last entry}}$

Return $\dot{\mathbf{q}} = g^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$

Algorithms for connectivity check

We present the remainder of connectivity check algorithms presented in Section 12 (refer Table S3). The connectivity check is defined for each of the possible type of factors of expression of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$. In the following body, we presents checks to assess connectivity between query joint configurations $\mathbf{q}_A$ and $\mathbf{q}_B$.

Case (i)

The check for Case (i), when the factor class is $2(1,0)[*,0]$, is presented below in Algorithm 14.

Algorithm 14. Case (i): factor with branch type $(1,0)[*,0]$

Input: Joint configurations $\mathbf{q}_A$, $\mathbf{q}_B$

Output: Boolean indicating connectivity without crossing singularities

Check for singularity separation:

If $\text{sign}(\det \mathbf{J}(\mathbf{q}_A)) \neq \text{sign}(\det \mathbf{J}(\mathbf{q}_B))$:

Points are separated by a singularity

Return: false

Make horizontal connection lines H**:**

$H \leftarrow \{h_i : q_3 = \mathbf{q}^i(q_3)\}$ for all $\mathbf{q}^i \in Q^h \cup \{\underline{q_3}, \overline{q_3}\}$

Make vertical connection lines V:

$V \leftarrow \{v_1 : q_2 = \overline{q_2}, v_2 : q_2 = \underline{q_2}, \}$

Project points orthogonally onto horizontal and vertical lines:

$P_H^{A,B}(q_2, q_3) \leftarrow \{q_2^{A,B}, \mathbf{q}^h(q_3)\}$

$P_V^{A,B}(q_2, q_3) \leftarrow \{\mathbf{q}^v(q_2), q_3^{A,B}\}$

For each $m \in \{A, B\}$**:**

Identify projections with same sign of Jacobian determinant

$Z_{H,V}^m \leftarrow \{\det \mathbf{J}(\mathbf{p}_i) \cdot \det \mathbf{J}(\mathbf{q}_m) \geq 0 \; \forall \; \mathbf{p}_i \in P_{H,V}^m\}$

Obtain valid indices using continuity check (Algorithm 15):

$I_H^m \leftarrow P_H^m(q_3), Z_H^m, \mathbf{q}_m(q_3)$

$I_V^m \leftarrow P_V^m(q_2), Z_V^m, \mathbf{q}_m(q_2)$

Identify common valid lines:

$H_C \leftarrow I_H^A \cap I_H^B$

$V_C \leftarrow I_V^A \cap I_V^B$

Return: $|H_C| \geq 1$ and $|V_C| \geq 1$

The continuity check in Algorithm 15 is used in conjunction with Algorithm 14 for factor with branches $(1, 0)$ in Case (i) Table S3.

Algorithm 15. Continuity check for Case (i)

Input: Projections $P(q_i)$, query value q_i^c, boolean mask Z from Algorithm 14

Output: Index ranges $\{a : b, d : c\}$ indicating connectivity windows

Sort projections and align booleans:

Let π be the permutation that sorts $P(q_i)$ in ascending order

$P_{\text{sorted}} \leftarrow P(q_i)[\pi]$

$Z_{\text{sorted}} \leftarrow Z[\pi]$

Identify ascending connectivity interval:

$a \leftarrow \min\{k \mid P_{\text{sorted}}[k] \geq q_i^c\}$

$b \leftarrow$ smallest index such that $(b \geq a) \wedge (Z_{\text{sorted}}[b+1] = \text{false})$

Identify descending connectivity interval:

$c \leftarrow \max\{k \mid P_{\text{sorted}}[k] < q_i^c\}$

$d \leftarrow$ largest index such that $(d < c) \wedge (Z_{\text{sorted}}[d-1] = \text{false})$

Return: $\{a : b, d : c\}$

Case (ii)

Factor f with branch type $(0, 1)$ and no horizontal tuning points, has two branches by encircling the q_3 generator twice. The set I_{SL} contains the points of intersection between the joint limits $q_2^{\times}, q_3^{\times}$ and the factor f. The joint space regions in vicinity of these branches are denoted as B_+, B_- To determine a joint configuration is in vicinity of which of the two branches, we can utilise the sign of the factor:

$$B = \begin{cases} +1 & \text{if } \operatorname{sign} \mathrm{f}(\mathbf{q} + [\epsilon_q, 0]^T) > 0 \\ -1 & \text{if } \operatorname{sign} \mathrm{f}(\mathbf{q} + [\epsilon_q, 0]^T) < 0 \end{cases} \tag{S31}$$

Note, a configuration lies on $q_2^{\times}$, the branch is assigned using any configuration in I_{SL}.

The check is presented in Algorithm 16. Figure S19 presents the check process in $q_2 - q_3$ slice for robot from Category 2 (B).

Algorithm 16. Case (ii): factor with branch type $(0, 1)[0, *]$

Input: Joint configurations $\mathbf{q}_A$ and $\mathbf{q}_B$

Output: Boolean indicating if configurations are connected

Singularity check:

If $\operatorname{sign}(\det(\mathbf{J}(\mathbf{q}_A))) \neq \operatorname{sign}(\det(\mathbf{J}(\mathbf{q}_B)))$ then

Returnfalse

Construct horizontal lines through $\mathbf{q}_A$ and $\mathbf{q}_B$:

$H_A \leftarrow \{\mathbf{q} \mid \mathrm{f}(\mathbf{q}) = 0, \mathbf{q}(q_3) = \mathbf{q}_A(q_3)\} \cup \{[\underline{q_2}, \mathbf{q}_A(q_3)], [\overline{q_2}, \mathbf{q}_A(q_3)]\}$

$H_B \leftarrow \{\mathbf{q} \mid \mathrm{f}(\mathbf{q}) = 0, \mathbf{q}(q_3) = \mathbf{q}_B(q_3)\} \cup \{[\underline{q_2}, \mathbf{q}_B(q_3)], [\overline{q_2}, \mathbf{q}_B(q_3)]\}$

Identify nearest neighbours:

$\mathbf{q}_{A1}, \mathbf{q}_{A2} \leftarrow$ neighbours of $\mathbf{q}_A$ along q_2 in H_A

$\mathbf{q}_{B1}, \mathbf{q}_{B2} \leftarrow$ neighbours of $\mathbf{q}_B$ along q_2 in H_B

Identify branches for each neighbour:

$B_{A1}, B_{A2} \leftarrow$ branches for $\mathbf{q}_{A1}$ and $\mathbf{q}_{A2}$

$B_{B1}, B_{B2} \leftarrow$ branches for $\mathbf{q}_{B1}$ and $\mathbf{q}_{B2}$

Note: If point lies on $q_2^{\times}$, assign branch as 0

Check connectivity:

If $B_{A1} \neq B_{A2} \wedge B_{B1} \neq B_{B2}$:

Return: true

If $(B_{A1} = B_{A2} \wedge B_{B1} \neq B_{B2}) \vee (B_{B1} = B_{B2} \wedge B_{A1} \neq B_{A2})$:

Return: false

If $B_{A1} = B_{A2} \wedge B_{B1} = B_{B2}$:

If $\mathbf{q}_A(q_3)$ and $\mathbf{q}_B(q_3)$ lie between the same I_{SL} pair along the $q_2 = \overline{q_2}$ or $\underline{q_2}$:

Return: true

Else:

Return: false

Return: false

Case (iii)

Factor f with class $(0, 0)$ has a single branch, forming a closed loop, without encircling any generator.

I_{SS} is the set of configurations at the intersection of f and another factor (if any) of the expression of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$. To ensure that Q_h, Q_v, I_{SS}, I_{SL} lie on a continuous branch, not split by joint limits, we first translate these points using Algorithm 21. To specify the connectivity,

it is once again required to differentiate between the interior of the loop created by the branch and its exterior. To do so, we identify the horizontal turning point $\mathbf{q}_h$ with the least q_3 value. A configuration $\mathbf{q}$ is inside the loop if $\text{sign}(\text{f}(\mathbf{q})) = \text{sign}(\text{f}(\mathbf{q_h} + [0, \epsilon_q]^T))$, and outside otherwise. The check is presented in Algorithm 17. Figure S20 presents the check process in $q_2 - q_3$ slice for robot from Category 5 (C).

Algorithm 17. Case (iii): factor with branch type $(0,0)[2,*]$

Input: Joint configurations $\mathbf{q}_A$ and $\mathbf{q}_B$

Output: Boolean indicating connectivity inside a closed loop

Check singularity separation:

If $\text{f}(\mathbf{q}_A) \cdot \text{f}(\mathbf{q}_B) < 0$ then

Configurations are separated by singularity

Return: false

If both points lie outside the loop:

Use Algorithm 14

Else:

Use Algorithm 21 to obtain Q_h, Q_v

Displace $\mathbf{q}_A$ and $\mathbf{q}_B$ by $\mathbf{d}_A$ and $\mathbf{d}_B$ such that $\mathbf{q}_A$ and $\mathbf{q}_B$ are lies in the q_3 span of Q_h and q_2 span of Q_v

Note: Only displacements of $\pm 2\pi$ *along* q_2 *or* q_3 *are allowed.*

Same translation rule as in Algorithm 21

Check displacements:

If $\mathbf{d}_A = \mathbf{d}_B = 0 \ \vee \ (\mathbf{d}_A(q_2) = \mathbf{d}_B(q_2) = 0 \ \wedge \ \mathbf{d}_A(q_3) = \mathbf{d}_B(q_3))$:

Return: true

Else If $\mathbf{d}_A(q_2) = \mathbf{d}_B(q_2) \neq 0 \ \wedge \ \mathbf{d}_A(q_3) = \mathbf{d}_B(q_3)$:

Check intermediate singularities:

$I_{SL}^{\text{mid}} \leftarrow \{\mathbf{q}_{SL} \in I_{SL} \mid$

$\min(\mathbf{q}_A(q_3), \mathbf{q}_B(q_3)) < \mathbf{q}_{SL}(q_3) < \max(\mathbf{q}_A(q_3), \mathbf{q}_B(q_3))\}$

For each intermediate singularity $\mathbf{q}_i$:

Let ϵ be a small distance in direction of $\mathbf{q}_A - \mathbf{q}_i$

If $\det(\mathbf{J}(\mathbf{q}_i + [\epsilon, 0]^T)) \cdot \det(\mathbf{J}(\mathbf{q}_A)) < 0$:

Return: false

Else:

Return: false

Return: true

Case (iv)

Factor f with atleast one branch of type $(1,0)[2,>0]$ will create a *meander* or *fold* while encircling the q_2 generator.

The check is presented in Algorithm 18. Figure S18 presents the check process in $q_2 - q_3$ slice for a robot from Category 4, with factor class $2(1,0)[2,2]$

Algorithm 18. Case (iv): factor with a branch of type $(1,0)[2,>0]$

Input: Query configurations $\mathbf{q}_A$, $\mathbf{q}_B$

Check singularity separation:

If $\mathrm{f}(\mathbf{q}_A) \cdot \mathrm{f}(\mathbf{q}_B) < 0$ then

Configurations are separated by singularity

Return: false

Fold information retrieval:

Obtain (Q_h, Q_v) translated from Algorithm 20

Separate individual (1, 0) branches by separating the Q_v related to consecutive pairs of Q_h

Collect the bounding boxes corresponding to each branch (refer to section *Robots from Category 4*)

Reverse the branch closure.

Create a chain of connected bounding boxes

Collect series of connected bounding boxes by investigating common Q_h, Q_v or I_{SL}

A series $\mathcal{B}_{Si}$ is a union of multiple bounding boxes such that a point lying in any bounding box $\mathcal{B}_i \in \mathcal{B}_{Si}$ can reach any other point lying in $\mathcal{B}_j \in \mathcal{B}_{Si}$

Create a collection of union of series

Create two collections, $\mathcal{P}_{S_k}$ and $\mathcal{N}_{S_k}$

Start counter clockwise travel starting from (π, π) along the outer boundary of the joint space

if $\det(\mathbf{J}) > 0$ at (π, π), start with $\mathcal{P}_{S_1}$, otherwise with $\mathcal{N}_{S_1}$

go till next $\mathbf{q}_n \in \{I_{SL}$ or $I_{LL}\}$:

make a bounding box, $\mathcal{B}_{new}$, with previous I_{SL} or I_{LL}

add $\mathcal{B}_{new}$ to $(\mathcal{P}/\mathcal{N}_{S_k})$

if $\mathbf{q}_n \in I_{SL}$:

Identify the series $\mathcal{B}_{Si}|\mathbf{q}_n \in \mathcal{B}_i \in \mathcal{B}_{Si}$

Add $\mathcal{B}_{Si}$ to the collection $(\mathcal{P}/\mathcal{N}_{S_k})$

switch the collection $(\mathcal{P}/\mathcal{N}_{S_k} \rightarrow \mathcal{N}/\mathcal{P}_{S_k})$

$k \leftarrow k + 1$

else: continue

Group the collections to form bundary of each distinct region

Make a group, $\mathcal{G}_i$ that is a union of $\mathcal{P}_{S_j}|\exists\mathcal{B}_{Sk} \in \forall\mathcal{P}_{S_j}$

make similar groups for $\mathcal{N}_{S_j}$

Associate a query point with a bounding box

Project a query point $\mathbf{q}_A$ vertically and choose the closest point, $\mathbf{q}_{As}$ on the singularity ($\mathbf{q}_{As} \in \mathcal{S}$).

Project a query point $\mathbf{q}_B$ vertically and choose the closest point, $\mathbf{q}_{Bs}$ on the singularity ($\mathbf{q}_{Bs} \in \mathcal{S}$).

Identify the bounding box $\mathcal{B}_{Ai}$ such that ($\mathbf{q}_{As} \in \mathcal{B}_{Ai}$).

Identify the bounding box $\mathcal{B}_{Bi}$ such that ($\mathbf{q}_{Bs} \in \mathcal{B}_{Bi}$).

Decide on connectivity

Identify the series $\mathcal{B}_{SA}, \mathcal{B}_{SB}$ related to $\mathcal{B}_{Ai}$ and $\mathcal{B}_{Bi}$

if $\det(\mathbf{J}(\mathbf{q}_A)) > 0$:

identify $\mathcal{P}_{S_A}|\mathcal{B}_{SA} \in \mathcal{P}_{S_A}$ and $\mathcal{P}_{S_B}|\mathcal{B}_{SB} \in \mathcal{P}_{S_B}$

if $\exists\mathcal{G}_i|\mathcal{P}_{S_A} \in \mathcal{G}_i \wedge \mathcal{P}_{S_B} \in \mathcal{G}_i$:

Return True

else if $\det(\mathbf{J}(\mathbf{q}_A)) < 0$:

identify $\mathcal{N}_{S_A}|\mathcal{B}_{SA} \in \mathcal{N}_{S_A}$ and $\mathcal{N}_{S_B}|\mathcal{B}_{SB} \in \mathcal{N}_{S_B}$

if $\exists\mathcal{G}_i|\mathcal{N}_{S_A} \in \mathcal{G}_i \wedge \mathcal{N}_{S_B} \in \mathcal{G}_i$:

Return True

else:

Return False

Case (v)

Factor f with branch type $(1,1)[0,*]$, where each branch encircles both the q_2 and q_3 generators. The check is presented in Algorithm 19. Figure S21 presents the check process in $q_2 - q_3$ slice for a robot from Category 2 (E), with factor class $(1,1)[0,2] + (1,1)[0,0]$.

Algorithm 19. Case (V): factor with a branch of type $(1,1)$

Input: Configurations $\mathbf{q}_A$, $\mathbf{q}_B$

Output: Boolean indicating connectivity between the two points

$\mathbf{t}_{SL} \leftarrow$ normal at configuration $\mathbf{q} \in I_{SL}$

$I_{SL} \leftarrow I_{SL} \cup I_{LL}$

If $\mathbf{t}_{SL}(q_2) > 0$:

Arrange I_{SL} in clockwise order from $(\pi, -\pi)$

Else:

arrange from $(-\pi, -\pi)$

$I_{SL} \leftarrow I_{SL} \setminus I_{LL}$

$I_{SL}^{-} \leftarrow$ counterclockwise ordering of I_{SL}

Define aspect bounding:

$A_i \leftarrow \{I_{SL_i}, I_{SL_{i+1}}, I_{SL_i}^{-}, I_{SL_{i+1}}^{-}\}$

Identify aspect of $\mathbf{q}_A$:

$Q_{CH_A} \leftarrow \{\mathbf{q} \mid \det \mathbf{J}(\mathbf{q}_A(q_3)) = 0\} \cup \{(\overline{q_2}, \mathbf{q}_A(q_3)), (\underline{q_2}, \mathbf{q}_A(q_3))\}$

$Q_{CV_A} \leftarrow \{\mathbf{q} \mid \det \mathbf{J}(\mathbf{q}_A(q_2)) = 0\} \cup \{(\mathbf{q}_A(q_2), \overline{q_3}), (\mathbf{q}_A(q_2), \underline{q_3})\}$

$Q_{N_A} \leftarrow$ immediate neighbours of $\mathbf{q}_A$ in $Q_{CH_A} \cup Q_{CV_A}$

Expand search until joint limits are hit:

While no $\mathbf{q} \in Q_{N_A}$ lies on joint limits:

For each $\mathbf{q}_i \in Q_{N_A}$:

$Q_{CH_i} \leftarrow \{\mathbf{q} \mid \det \mathbf{J}(\mathbf{q}_i(q_3)) = 0\} \cup \{(\overline{q_2}, \mathbf{q}_i(q_3)), (\underline{q_2}, \mathbf{q}_i(q_3))\}$

$Q_{CV_i} \leftarrow \{\mathbf{q} \mid \det \mathbf{J}(\mathbf{q}_i(q_2)) = 0\} \cup \{(\mathbf{q}_i(q_2), \overline{q_3}), (\mathbf{q}_i(q_2), \underline{q_3})\}$

Add neighbors of $\mathbf{q}_i$ in $Q_{CH_i} \cup Q_{CV_i}$ to Q_{N_A}

Determine aspect of $\mathbf{q}_A$ as $\mathcal{A}_A$ by checking which A_i it lies in

Repeat Steps 7–9 for $\mathbf{q}_B$ to obtain $\mathcal{A}_B$

Compare aspects:

If $\mathcal{A}_A = \mathcal{A}_B$:

Return: true

Else:

Return: false

Branch Closure algorithms

For robots from Categories 4 through 6, it is possible that the branch(es) associated with the factor(s) may not form a continuous curve in the $q_2 - q_3$ slice bounded by the joint limits. In such cases, the horizontal or vertical turning points no longer lie on a continuous branch in the $q_2 - q_3$ slice. As a result, when moving along the constraint boundary between two turning points, joint-limits may be occur. This makes specifying a generalised behaviour near constraints challenging. To avoid this, we first translate the Q_h, Q_v, I_{SL}, I_{SS} configurations in the $q_2 - q_3$ slice such that no joint limit boundary is encountered between turning points. We refer to this translation process as *branch closure*. Next, the track cycle is determined. For determining the joint velocity, the Q_h, Q_v, I_{SL}, I_{SS} are translated back to the original slice bounded by joint limits.

We present the method for performing branch closure for a factor with a branch of type $(1, 0)[2, > 0]$ (Connectivity for Case (iv)) in Algorithm 20, and the method for factors with a branch of type $(0, 0)[]$ branch is presented in Algorithm 21

For factor with a branch type $(1, 0)[2, > 0]$

Algorithm 20. Branch Closure for factors with branch types $(1, 0)[*, > 0]$

Input: Q^h, Q^v, expression of $\det \mathbf{J}$, I_{SL}

Output: Ordered turning points maintaining continuity in 2π interval

Check vertical split and adjust turning points if needed:

If $\exists\, \mathbf{q}_{\text{SL}} \in I_{\text{SL}} \mid |\mathbf{q}_{\text{SL}}(q_3)| = \pi$:

$\mathbf{q}_{sh} \leftarrow \arg\min_{\mathbf{q}_i \in Q^h} \mathbf{q}(q_3)$

$Q_h \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in Q^h, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

$Q_v \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in Q^v, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

$I_{\text{SL}} \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in I_{\text{SL}}, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

Group turning points by branch:

$Q_h^+ \leftarrow \{\mathbf{q} \in Q_h \mid \det \mathbf{J}(\mathbf{q} + (0, \epsilon)) > 0\}$

$Q_h^- \leftarrow Q_h \setminus Q_h^+$

$Q_v^+ \leftarrow \{\mathbf{q} \in Q_v \mid \mathbf{q}(q_3) \in \text{span}(Q_h^+(q_3))\}$

$Q_v^- \leftarrow Q_v \setminus Q_v^+$

$I_{\text{SL}}^+ \leftarrow \{\mathbf{q} \in I_{\text{SL}} \mid \mathbf{q}(q_3) \in \text{span}(Q_h^+(q_3))\}$

$I_{\text{SL}}^- \leftarrow \{\mathbf{q} \in I_{\text{SL}} \mid \mathbf{q}(q_3) \in \text{span}(Q_h^-(q_3))\}$

Check and adjust horizontal split:

For each $m \in \{+, -\}$:

If $|\{\mathbf{q}_{\text{SL}} \in I_{\text{SL}}^m \mid |\mathbf{q}_{\text{SL}}(q_2)| = \pi\}| > 2$:

$\mathbf{q}_{sv} \leftarrow \arg\max_{\mathbf{q}_i \in Q_v^m} \mathbf{q}(q_2)$

$Q_h^m \leftarrow \{\mathbf{q} + (2\pi, 0) \text{ if } \mathbf{q}(q_2) < \mathbf{q}_{sv}(q_2), \text{ else } \mathbf{q} \mid \mathbf{q} \in Q_h^m\}$

$Q_v^m \leftarrow \{\mathbf{q} + (2\pi, 0) \text{ if } \mathbf{q}(q_2) < \mathbf{q}_{sv}(q_2), \text{ else } \mathbf{q} \mid \mathbf{q} \in Q_v^m\}$

Return: $(Q_h^+, Q_v^+, I_{\text{SL}}^+), (Q_h^-, Q_v^-, I_{\text{SL}}^-)$

For factor with a branch type $(0, 0)[2, > 0]$

Algorithm 21. Loop Closure for factors with branch types $(0, 0)[2, > 0]$

Input: Q^h, Q^v, factor f_i of $\det \mathbf{J}$, I_{SL}, I_{SS}

Output: Ordered saddle points maintaining continuity over 2π interval

Check vertical split and adjust saddle points if needed:

If $\exists \mathbf{q}_{\mathrm{SL}} \in I_{\mathrm{SL}}$ s.t. $|\mathbf{q}_{\mathrm{SL}}(q_3)| = \pi$:

$\mathbf{q}_{sh} \leftarrow \arg\min_{\mathbf{q}_i \in Q^h} \mathbf{q}(q_3)$

$Q_h \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in Q^h, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

$Q_v \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in Q^v, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

$I_{\mathrm{SL}} \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in I_{\mathrm{SL}}, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

$I_{\mathrm{SS}} \leftarrow \{\mathbf{q}_i + (0, 2\pi) \mid \mathbf{q}_i \in I_{\mathrm{SS}}, \mathbf{q}_i(q_3) \leq \mathbf{q}_{sh}(q_3)\}$

Check horizontal split $Q_v^{\mathrm{sorted}} \leftarrow$ sort Q_v in ascending order of q_2

Loop: For i in $\{1, 2, \ldots, |Q_v^{\mathrm{sorted}}| - 1\}$

If $f_i(\mathbf{q}_i) \cdot f_i(\mathbf{q}_{i+1}) < 0$:

$Q_v^{\mathrm{sorted}} \leftarrow \{\mathbf{q} + (2\pi, 0) \text{ if } \mathbf{q}(q_2) \leq \mathbf{q}_i(q_2)\}$

$Q_h \leftarrow \{\mathbf{q} + (2\pi, 0) \text{ if } \mathbf{q}(q_2) \leq \mathbf{q}_i(q_2)\}$

$I_{\mathrm{SS}} \leftarrow \{\mathbf{q} + (2\pi, 0) \text{ if } \mathbf{q}(q_2) \leq \mathbf{q}_i(q_2)\}$

$I_{\mathrm{SL}} \leftarrow \{\mathbf{q} + (2\pi, 0) \text{ if } \mathbf{q}(q_2) \leq \mathbf{q}_i(q_2)\}$

If $\exists \mathbf{q}_h \in Q^h$ s.t. $\mathbf{q}_h \notin \mathrm{span}(Q_v^{\mathrm{sorted}}(q_2))$:

$i \leftarrow 0$ (reset loop) Renumber the $Q_v in ascending q_2$

Else:

Return: $(Q_h, Q_v^{\mathrm{sorted}}, I_{\mathrm{SL}}, I_{\mathrm{SS}})$

Return: $(Q_h, Q_v^{\mathrm{sorted}}, I_{\mathrm{SL}}, I_{\mathrm{SS}})$

Incremental update by Workspace-to-Joint Space Transfer

In Section 12 we discussed the modelling of the demonstrated behaviour in an aspect of the joint space, using multiple demonstrations. The same procedure can also be used for modelling the workspace behaviour by leveraging multiple workspace demonstrations $\{\mathbf{X}\}_{i=1}^{W}$. The workspace contains a single goal position $\overset{*}{\mathbf{x}}$, and the workspace behaviour $\mathrm{B}^{(\mathcal{X})}$ can be modelled as $\mathrm{B}^{(\mathcal{X})} = \{P, \Lambda, \{\mathrm{g}_i^{(\mathcal{X})} \leftrightarrow (\psi_i, \mathbf{k}_i)\}_{i=1}^{W}\}$. The workspace behaviour also gets updated when novel workspace demonstrations are provided by the user (Section 12).

Using the available demonstration data, we have modelled the workspace behaviour $\mathrm{B}^{(\mathcal{X})}$ and the preliminary joint space $\mathrm{B}^{(\mathcal{Q})}$. The preliminary $\mathrm{B}^{(\mathcal{Q})}$ accurately replicates the demonstrated trajectories while handling perturbations and adhering to the robot's kinematic constraints. As a result, they provide a robust model of the behaviour across the joint space. However, to more effectively incorporate the embedded knowledge from user-demonstrated workspace behaviour, we introduce a generalization step.

Generalization is achieved through an iterative process that identifies joint-space trajectories contributing the most novel information. A trajectory is considered to improve generalization if a significant portion of its configurations lie outside the distribution of previously recorded demonstrations. In this section, we go over the steps to generalise the behaviour in joint space $\mathcal{Q}$

1. First, the joint space joint space $\mathcal{Q}$ is sampled to generate candidate configurations.

2. Next, from their workspace counterparts, *transfer* trajectories in workspace $\mathcal{X}$ are generated using workspace behavior $\mathrm{B}^{(\mathcal{X})}$. These trajectories are then inverted back into joint space.

3. A filtering step follows to detect and trim the *incorrect* transfer trajectories in each aspect of joint space $\mathcal{Q}$.

4. Identify the most novel transfer trajectory in each aspect and update $\mathrm{B}^{(\mathcal{Q})}$.

This step-by-step approach ensures a robust and accurate update of $\mathrm{B}^{(\mathcal{Q})}$, while maintaining that after each update the modelled behaviour remains stable, respects kinematic constraints, and is robust to perturbations.

Generating transfer trajectories

A uniform grid of points is generated in joint space $\mathcal{Q}$, ensuring even coverage across the range of motion for each joint. This grid provides a comprehensive set of candidate joint configurations for subsequent evaluation and analysis. Let the ranges of the two joints be $[\underline{q_1}, \overline{q_1}]$ and $[\underline{q_2}, \overline{q_2}]$, each discretized into m_{q_1} and m_{q_1} points, respectively. The set of sampled joint configurations is defined as

$$\{\mathbf{Q}\}^{\text{sample}} = \{(q_1, q_2) \mid q_1 \in Q_1,\ q_2 \in Q_2\},$$

where

$$Q_1 = \{\underline{q_1} + \epsilon_q + i\Delta q_1\}_{i=0}^{m_{q_1}}, \quad \Delta q_1 = \frac{\overline{q_1} - \underline{q_1} - 2\epsilon_q}{n_1 - 1}$$
$$Q_2 = \{\underline{q_2} + \epsilon_q + i\Delta q_2\}_{i=0}^{m_{q_2}}, \quad \Delta q_2 = \frac{\overline{q_2} - \underline{q_2} - 2\epsilon_q}{n_2 - 1}$$

where $\epsilon_q > 0$ is user-specified safety margin (in radians) around the joint limits. $\{\mathbf{Q}\}^{\text{sample}}$ is next mapped to workspace $\mathcal{X}$ to yield a set of candidate end effectir positions

$$\{\mathbf{x}\}^{\text{sample}} = \{\mathbf{x}_i = \text{FK}(\mathbf{q}_i \in \{\mathbf{Q}\}^{\text{sample}})\}_{i=1}^{|\{\mathbf{Q}\}^{\text{sample}}|}$$

Since the FK is a many-to-one mapping, there exist some repeated entries in $\{\mathbf{x}\}^{\text{sample}}$ which are eliminated. The candidate end effectir positions are used to generate a set of *transfer* trajectories $\{\mathbf{X}\}^{\text{transfer}}$ where each trajectory

$$\mathbf{X}_i^{\text{transfer}} = \{\mathbf{x}_t \mid \mathbf{x}_{t=0} = \mathbf{x}_i,\ t \in [0, \infty),\ \lim_{t\to\infty} \mathbf{x}_t = \overset{*}{\mathbf{x}}\}$$

is generated in workspace $\mathcal{X}$ by the relevant DS embedding $\text{g}^{(\mathcal{X})}$, starting from the initial state $\mathbf{x}_i$. The relevant DS embedding is determined from the workspace behaviour $\text{B}^{(\mathcal{X})} = \{P, \Lambda, \{\text{g}_i^{(\mathcal{X})} \leftrightarrow (\psi_i, \mathbf{k}_i)\}\}$ similar to Equation S22, but for workspace DS embeddings.

These trajectories can now be inverted back into joint space $\mathcal{Q}$, generating a set of joint trajectories $\{\mathbf{Q}\}^{\text{transfer}}$ which encapsulate additional information about the user behaviour in workspace $\mathcal{X}$.

Some of the transfer trajectories in $\{\mathbf{X}\}^{\text{transfer}}$ may be infeasible (refer to Section). Thus, we perform the validation and trimming procedure (refer to Section 12) for each transfer trajectory in $\{\mathbf{Q}\}^{\text{transfer}}$.

Measuring trajectory novelty

Each new joint space trajectory in $\{\mathbf{Q}\}^{\text{transfer}}$ encapsulates some additional information that can contribute towards updating the overall behaviour $\text{B}^{(\mathcal{Q})}$. However, instead of learning the embeddings for all of the trajectories, we selectively prioritise learning those embeddings which encapsulate the most additional information. If a transfer trajectory has a significant inclusion of joint configurations that exhibit little overlap with the already learnt behaviour in $\text{B}^{(\mathcal{Q})}$, this transfer trajectory embeds more novel information as compared with a trajectory that shows significant overlap with $\text{B}^{(\mathcal{Q})}$.

Selectively prioritising the most novel demonstrations enables us to transfer behaviour from workspace $\mathcal{X}$ to joint space $\mathcal{Q}$, while minimising the embeddings to be learnt. Doing so keeps the model of $\text{B}^{(\mathcal{Q})}$ parameter-efficient. The *novelty* metric for prioritising trajectories is based on their overlap with the known partitions in each aspect of joint space $\mathcal{Q}$. Lower overlap makes for a trajectory that encapsulates most additional behaviour for transfer.

To do so, first the aspect of a trajectory in $\{\mathbf{Q}\}^{\text{transfer}}$ is determined $\mathcal{A}_i = \mathcal{A}(\mathbf{Q}_i^{\text{transfer}})$. The aspect-specific behaviour $\text{B}^{(\mathcal{A}_i)}$ from the learnt joint behaviour $\text{B}^{(\mathcal{Q})}$ is queried. Next, from $\text{B}^{(\mathcal{A}_i)} = \{P_i, \Lambda_i, \{\text{g}^{(\mathcal{Q})} \leftrightarrow (\psi, \mathbf{k})\}_i\}$, the relevant DS embeddings $\{\text{g}^{(\mathcal{Q})} \leftrightarrow (\psi, \mathbf{k})\}_i$ are retrieved to calculate the projection angles (refer Equation S23) $\{\Theta\}_i$ subtended by the transfer trajectory in the latent space of the known DS embeddings

$$
\begin{aligned}
\Theta_j(\mathbf{Q}^{\text{transfer}}) &= \{\angle(\psi_j^{-1}(\mathbf{q}_t) - \mathbf{1}_n^T, \mathbf{1}_n^T - \mathbf{0}_n^T)\}_{t=1}^{|\mathbf{Q}^{\text{transfer}}|} \\
\{\Theta\}_i &= \{\Theta_j \forall\ \psi_j\ \ \in \{\text{g}^{(\mathcal{Q})} \leftrightarrow (\psi, \mathbf{k})\}_i\} \\
\text{novelty}(\mathbf{Q}^{\text{transfer}}) &= \min_{1 \le j \le |\,\text{B}^{(\mathcal{A}_i)}\,|} \left\{\{\Theta\}_j\right\}
\end{aligned}
\tag{S32}
$$

where $\mathbf{q}_t \in \mathbf{Q}^{\text{transfer}}$ is a joint configuration in the transfer trajectory. $\Theta_j(\mathbf{Q}^{\text{transfer}})$ contains the angles subtended by the latent embedding of transfer trajectory points generated using ψ_j. These angles correspond to the angular deviations between the latent embedding of the demonstration originally used to learn ψ_j, and the embeddings of points in the transfer trajectory ($\psi_j^{-1}(\mathbf{Q}^{\text{transfer}})$), computed in the latent space $\mathcal{U}$. $\{\Theta\}_i$ contains these angular deviations between the transfer trajectory $\mathbf{Q}^{\text{transfer}}$, and the original demonstrations associated with the DS embeddings of the identified aspect $\text{B}^{(\mathcal{A}_i)}$. The $\text{novelty}(\mathbf{Q}^{\text{transfer}}) \in \mathbb{R}^+$ corresponds to the least angular deviation of between the transfer trajectory and all of the recorded demonstrations. The most novel transfer trajectory from amongst

$\{\mathbf{Q}\}^{\text{transfer}}$ aims to have least overlap with existing demonstrations i.e. the maximum novelty value

$$\mathbf{Q}_{\text{novel}}^{\text{transfer}} = \text{argmax}\left\{\text{novelty}(\mathbf{Q}_i^{\text{transfer}}) \;\forall\; \mathbf{Q}_i^{\text{transfer}} \in \{\mathbf{Q}\}^{\text{transfer}}\right\} \tag{S33}$$

The most novel trajectory in each aspect is used to update the respective aspect-specific behaviour $\mathrm{B}^{(\mathcal{A}_i)}$, thereby updating the control policy $\mathrm{B}^{(\mathcal{Q})}$

Issues with numerical path-planning strategies

Null-space-based numerical strategies are inherently limited to redundant manipulators. When the Jacobian has full row rank and the number of joints matches the task dimension (as in non-redundant 6R arms), the null space is trivial and joint velocities reduce to $\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}}$, leaving no degree of freedom to encode posture optimisation, joint-limit avoidance, or aspect selection. As a result, any skill-transfer scheme that relies on shaping behaviour via null-space motions can only be applied to redundant robots, and cannot endow non-redundant robots with the same level of constraint-aware adaptability.

Even for redundant robots, numerical null-space controllers suffer from fundamental robustness and certification issues. A typical strategy computes

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\dot{\mathbf{x}} + \mathbf{N}(\mathbf{q})\,\mathbf{z},$$

where $\mathbf{J}^{\#}$ is a (damped) pseudoinverse, $\mathbf{N}$ the null-space projector, and $\mathbf{z}$ encodes secondary objectives (e.g., joint-limit avoidance, manipulability maximisation). Because the controller reasons only locally and never explicitly encodes aspects, singularity curves, or voids in the workspace, small changes in initialization, gains, or integration step can lead to qualitatively different joint-space behaviours for the same demonstrated path.

These effects are illustrated in Figure S22. In Figure S22(A), a particular initialization and parameter choice yield a seemingly ideal outcome: the redundant angle q_R remains nearly constant, the end-effector follows the demonstrated path, and no constraints are visibly approached. In Figure S22(B), for the same robot and task, but with different initial IKS, the path is not completed,

and the null-space evolution induces oscillations and shaking at the end of the trajectory, revealing sensitivity to the tuning of secondary objectives. In Figure S22(C), a different evolution of q_R steers the posture so that the robot tracks the workspace path while remaining close to singular manifolds along most of the motion; the task is accomplished, but the trajectory demonstrates erratic behaviour near constraints. Finally, in Figure S22(D), the null-space policy drives the arm into a configuration where two joint limits are reached, and the end-effector trajectory deviates from the demonstrated behaviour, resulting in an explicit failure.

More broadly, numerical planners operate on discretised waypoints and iterative optimisation or sampling between them. Even if all sampled configurations satisfy limits and avoid singularities, this only approximates the continuous path; detecting crossings of singularities, entries into IK holes, or aspect changes would require very dense sampling or additional symbolic checks that these methods do not natively provide. Consequently, they cannot distinguish genuine kinematic infeasibility (no start–goal connection exists in any aspect or in any redundant posture) from failures caused purely by local optimisation artefacts (an unfortunate null-space direction, an incorrect IK branch choice, or initialization). This motivates our certified execution layer, which explicitly factorises $\det \mathbf{J}$, computes aspects and their boundaries, and performs connectivity checks in reduced joint-space slices, providing deterministic, aspect-aware decisions that are not subject to the numerical fragility illustrated in Fig. S22.

Computational performance

Benchmarking was performed on a PC equipped with an Intel i9-10900K CPU (3.70 GHz). To evaluate the categorisation procedure in Figure 5(A2), we generated the factorised expressions of $\det J$ for 21,787 distinct 3R robots.

In the preprocessing for our kinematic benchmarking, we discretize the design parameter space of the $3R$ manipulator and, for each candidate, first test cuspidality using the known necessary-and-sufficient condition; only non-cuspidal robots are retained. For these, we start from the closed-form

expression of the Jacobian determinant, and symbolically factorise $\det \mathbf{J}$ after simplifying the equation by substituting zero link lengths/offsets and any joint angles fixed at 0° or 90°. We then compute horizontal and vertical turning points, as well as intersections of each factor with the boundary lines $q_{2,3} = \pm\pi$, using both half-angle tangent and cotangent substitutions to regularise the behaviour at 0 and $\pm\pi$.

For each robot, we assessed connectivity (C1) and constraint-violation (C2) checks using 100 joint-space configurations. These configurations were uniformly sampled on the q_2–q_3 slice with a fixed value for the first joint. To benchmark connectivity, conditioned on the robot's category, we computed the connectivity between each of the 100 sampled configurations and the reference configuration $[0, 0.5, 0.1]$. The average time required to determine connectivity across these 100 pairs was recorded.

For the near-constraint strategy (C4), we identified points lying close to violation boundaries within a common aspect and generated the corresponding near-boundary traversal strategy that respects all kinematic constraints. The time required to compute the control velocities for these near-boundary cases (C5) was also measured. These metrics together provide the average runtime per robot. The entire procedure was executed for all 22,000 robots, and the aggregated runtimes are reported in Figure 5.

To evaluate learning and replication of human-demonstrated behaviours, we used seven workspace demonstrations from the "Multi Models 1" subset of the LASA Handwriting Dataset (*48*). These demonstrations exhibit three sub-dynamics in the workspace and converge to a common attractor.

Figure 5(C) reports the time required to learn the aspect-specific behaviours for each robot; these tests were performed for approximately 500 robots. For each robot, we used the workspace demonstrations' starting points (mapped to joint space) to measure the time needed to (B) plan and identify the relevant joint primitive for replication and (C6) compute the corresponding joint velocities. We additionally evaluated the time taken to perform all-at-once sampling (C3) using the same starting points.

The average time to learn the model is presented in Figure 5(D1). Additionally, we computed the DTW score (using FastDTW (*49*)) between each demonstration's joint-space trajectory and its replication generated by the learnt joint-space control policy and have reported in (D2).

Robustness

Our framework separates *structural* kinematic robustness (how the singularity layout and aspect structure respond to modelling error) from *execution*-level robustness (how the closed-loop controller responds to noise, friction, and compliance).

Structural robustness to kinematic calibration errors. For a *generic* manipulator, small perturbations of the D–H parameters do not change the local homotopy type of the singularity set: the number of factors of $\det \mathbf{J}$, their winding behaviour, and the induced aspect structure are preserved in a neighbourhood of the nominal design. Only larger perturbations that move the mechanism across the discriminant set in parameter space (e.g., from a generic to a non-generic architecture, or from non-cuspidal to cuspidal) can change this topological structure. In our setting, we target industrial and collaborative robots whose calibration errors are typically a few percent and remain within the same class, so that the classification and connectivity results (robot category, factor types, aspects) remain valid under realistic parameter uncertainty.

Figure S23 illustrates this structural robustness on a representative 3R robot. The blue curve shows the nominal singularity, the red band corresponds to a safety margin, and the black curves show the singularities obtained by independently adding 5% noise to each D–H parameter that affects $\det \mathbf{J}$ (d_3, a_1, a_2, a_3, α_1, and α_2). In all cases, the perturbed singularities remain inside the red band and the qualitative structure in the region of interest is unchanged.

Perturbations of the D–H parameters that affect *joint orientations* (e.g., α_i) are known to be more delicate: even relatively small changes can move the mechanism from a non-generic configuration with intersecting singularity branches to a generic configuration where these branches no longer intersect, or even render a previously non-cuspidal robot cuspidal. We explicitly explored such perturbations and observed that, for orientation errors up to 5%, a safety margin of order 10^{-3} on $\det \mathbf{J}$ still encloses the perturbed singularity curves in the joint-space region where execution takes place (see additional illustration in Fig. S24). Thus, even when the global singularity class changes, the local boundary used by our feasibility and connectivity checks can be conservatively padded so that the true singularity manifold remains inside the safety margin used by the controller.

Safety margins and execution-level disturbances. Execution-level uncertainties (encoder noise, unmodelled friction, moderate compliance) are handled by introducing a *safety margin* around analytically computed constraints. Instead of treating the exact singularity curve $\det \mathbf{J} = 0$ and joint-limit surfaces as hard boundaries, we define a padded near-constraint band: configurations with $|\det \mathbf{J}| < \varepsilon$ or within a user-defined distance to joint limits are treated as if they were on the constraint. The value of ε can be tuned per robot; in our examples, values on the order of 10^{-3} are sufficient to cover the observed shifts due to 5% D–H perturbations, including those on orientation parameters.

Robustness of DS-based execution and track cycles. Within each aspect, the joint-space motion is governed by a continuous, globally stable dynamical system (DS) that realises the learned task-space policy. As long as the state remains inside the padded feasible region, small perturbations due to friction, compliance, or sensor noise are naturally rejected: the DS drives the configuration back towards its nominal flow. When the configuration enters the near-constraint band (either because the nominal DS would approach a boundary or due to disturbances), the violation-check step detects this and switches to the corresponding track-cycle rule. Track cycles are tangent and category-specific: by construction, they follow the local singularity or joint-limit boundary while remaining on the feasible side of the constraint. Because the underlying singularity structure is structurally robust for generic robots and the safety margin absorbs the residual shifts (including those induced by moderate orientation errors), the track-cycle execution remains feasible and behaves consistently under realistic modelling and sensing uncertainties.

We conducted a large-scale study for $\sim$ 22,000 non-cuspidal 3R designs, benchmarking classification, connectivity checks, violation checks, and track-cycle execution. All the steps in the process were completed without failures, and we provide the statistics underlying our timing table in Figure 5 of the main text. The corresponding benchmarking scripts and raw timing data are provided in the section 'Computational performance'. On all real robots considered (KUKA iiwa 7, Neura Maira M, and a 6-DoF Dyna arm with tight, non-symmetric joint limits), the factorisation, classification, and online checks succeeded without ambiguity. Each model was assigned to a unique non-cuspidal category, all singular branches were recovered, and we did not observe numerical failures or inconsistent behaviour. For a generic robot, our feasibility/connectivity checks and track-cycle execution

are structurally robust, since small perturbations of the D–H parameters do not alter the homotopy type of the singularity set. A user-defined safety margin around analytically computed constraints accommodates the resulting shifts in singularities

In summary, for generic robots, our feasibility and connectivity checks are structurally robust to realistic calibration errors, and the combination of user-defined safety margins, stable DS-based execution, and boundary-aligned track cycles ensures that encoder noise, friction, moderate compliance, and small orientation perturbations are absorbed without compromising the certified behaviour.

Failure modes

To demonstrate different failure modes, we tested a robot from Category 1 with varying inverse kinematic solutions (IKS): four in some regions and two in others. Figure 6(A1–A4) of the main text presents the user demonstration; the path transitions from a region with four IKS to a region with only two IKS. When replicating this behaviour from different possible IKS of the starting workspace position, our method constructs aspect-specific behaviour models in joint space and proceeds with execution only when the current configuration lies in a feasible aspect (Figure 6(C)); otherwise, the robot remains halted for safety (Figure 6(D)). When the demonstration transitions from a region with two IKS to a region with four IKS, only the terminal portion of the trajectory is used to learn the aspect-specific behaviour for the two newly gained feasible aspects. As a result, even when the robot is perturbed to a state where the complete path is unavailable, the control policy can still guide the robot to the goal without violating constraints, ensuring faithful and consistent reproduction of the intended motion.

In practice, certified execution exposes a set of well-defined, aspect-based failure modes. For non-redundant 6R robots, the positional subchain behaves as a standard 3R manipulator, and failures arise when (i) the goal is unreachable from the current aspect (no admissible path respecting joint limits and singularities exists), or (ii) perturbations drive the state towards joint limits or singularities. Before motion, we run an analytic connectivity check between the current and goal configurations; if no feasible connection exists in the current aspect, the controller refuses to move and reports the goal as unreachable. During execution, when the state enters a user-defined safety

margin around constraints, the controller switches to the appropriate track cycle; if re-entry into the interior is not feasible while preserving feasibility, it halts and reports a boundary-induced failure rather than violating constraints.

The same logic extends to wrist-partitioned redundant 7R arms. Fixing the redundant angle q_R defines a family of 3R subchains, and the above aspect-based checks are applied per subchain. An additional failure mode arises when no value of q_R yields a feasible aspect connecting start and goal; in this case, the task is declared infeasible for that robot, as illustrated in Figure 4(D) of the main text, where some aspects at certain redundant angles are detected as infeasible, and the robot remains halted. In all cases, the system explicitly reports whether the goal is reachable from the current aspect (and, for 7R, the current q_R) or whether execution stopped due to a boundary-induced or aspect-induced failure. The complete factor-wise feasibility framework and connectivity algorithms are detailed in the section *Certified Execution* of this supplementary text.

$q_2 - q_3$ slice	Along q_1	$\dot{\mathbf{q}}(q_2, q_3)$	$\dot{\mathbf{q}}(q_1)$
No violation	No violation	$\mathrm{g}^{(\mathcal{Q})}(\mathbf{q})$	$\mathrm{g}^{(\mathcal{Q})}(\mathbf{q})$
No violation	Violation	$\mathrm{g}^{(\mathcal{B})}(\mathbf{q}_{\text{target}}, \mathbf{q})$ (Eq. S12)	0
Violation	No violation	$\mathrm{g}^{(B)}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$ (Eq. S10)	$\mathrm{g}^{(B)}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$ (Eq. S11)
Violation	Violation	$\mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$ (Eq. S10)	$\mathrm{g}^{(\mathcal{B})}(\mathbf{q}, \mathbf{q}_{\text{last entry}})$ (Eq. S10)

Table S1: The resultant of the modelled behaviour policy $\mathrm{g}^{(\mathcal{Q})}$ and the near-boundary policy $\mathrm{g}^{(\mathcal{B})}$ in an aspect of the joint space of a $3R$ robot.

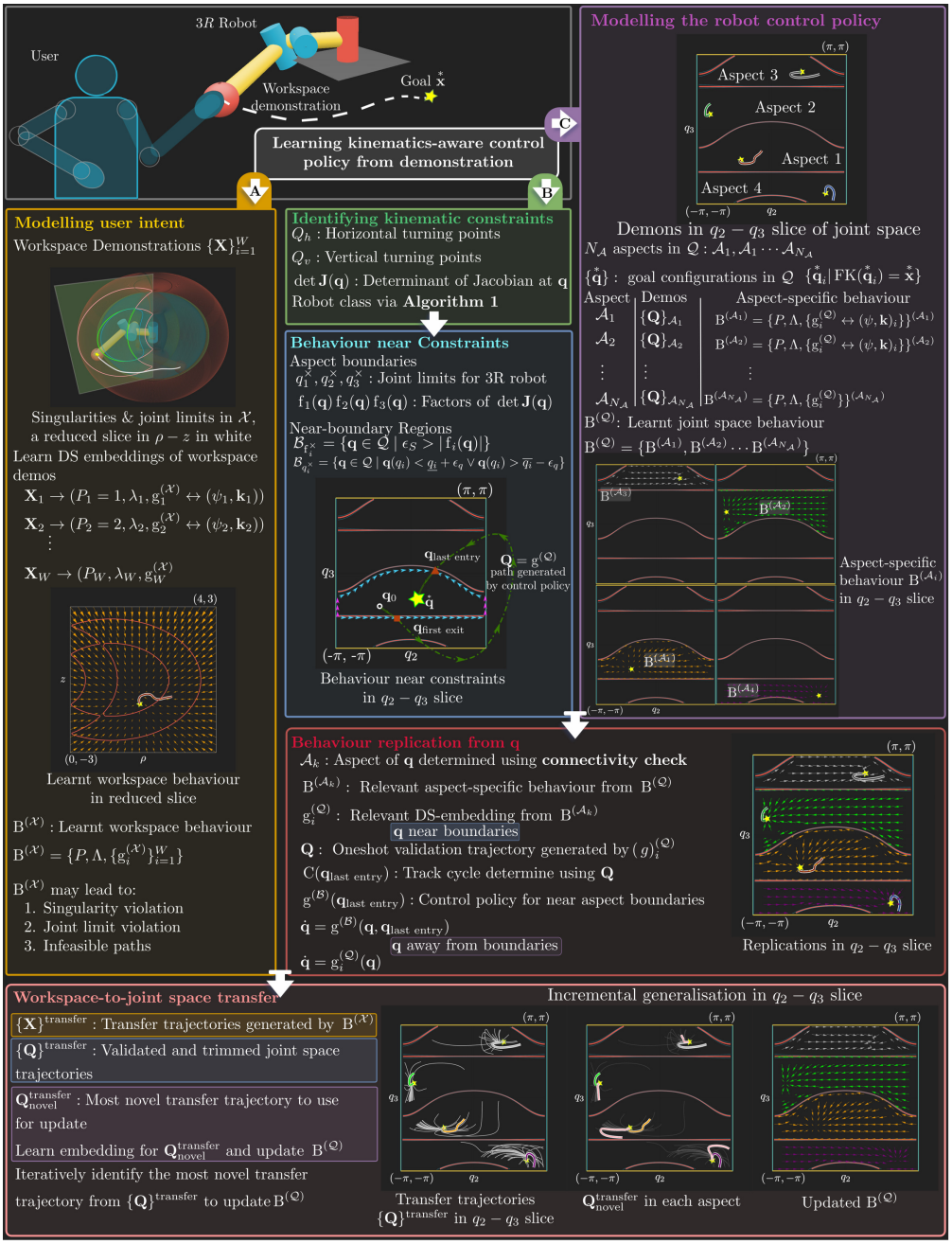


Figure S1: Overview of the proposed framework for learning kinematics-aware control policies from demonstration, and incremental generalization via workspace-to-joint space transfer. The figure illustrates both the operational flow, from demonstration to generalization, and highlights the sections corresponding to each component.

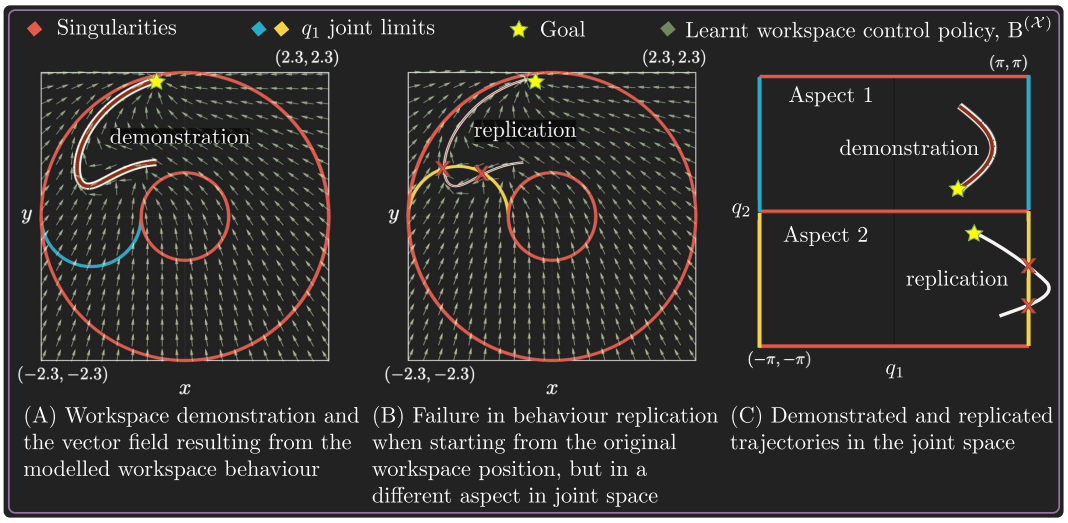


Figure S2: Modelling of the user demonstrated behaviour in the workspace, for a $2R$ robot. (A) and (B) represent the workspace, while (C) represents the joint space. (A) depicts the workspace demonstration, wherein joint configurations are in Aspect 1 of the joint space ((C) top). The workspace demonstration is used to learn the corresponding DS embedding and results in the workspace behaviour $\mathrm{B}^{(\mathcal{X})}$. During replication, when starting from the original workspace start position with the joint configuration in Aspect 2, the replication leads to violation of the first joint's limit (B). The joint space representation of the replication is presented in (C) bottom.

Case	**Homotopy class**	**Robot Examples** (Figure S3)	**Connectivity check**
(i)	$(1,0), Q_v = 0$	(A) Category 1; (F), (H) Category 3; Category 5	Algorithm 14
(ii)	$(0,1)$	(B) Category 2; (F) Category 3	Algorithm 16
(iii)	$(0,0)$	(C) Category 4; (G) Category 5	Algorithm 17
(iv)	$(1,0), Q_v > 0$	(D) Category 6	Algorithm 18
(v)	$(1,1)$	(E) Category 2; (H) Category 3	Algorithm 19

Table S2: Cases based on the types of factors in expression of Jacobian determinant $\det \mathbf{J}(\mathbf{q})$, homotopy class of branches in the factor, number of branches in the factor, corresponding robot examples from Figure S3, and the connectivity check algorithm specific to the case. Q_v is the set of vertical turning points in the factor

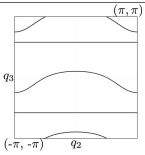
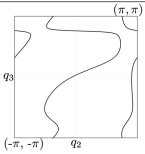
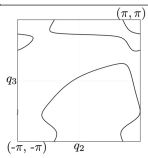
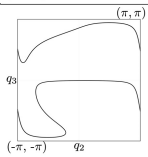
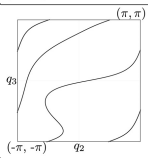
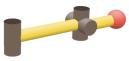
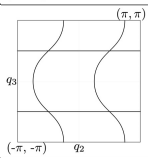
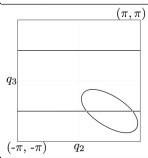
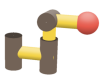
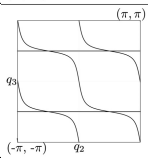
Robot parameters	Robot schematic	Joint space singularity in $q_2 - q_3$ slice	f_1		f_2		Singularity class	Robot category
			Q^h	Q^v	Q^h	Q^v		
(A) $\mathbf{d} = [0, 0.5, 0]$ $\mathbf{a} = [1, 0.5, 2]$ $\alpha = [\frac{\pi}{2}, \frac{\pi}{2}, 0]$			4	0	∞	0	$2(1,0)[4,0]$ + $2(1,0)[\infty,0]$	Category 1
(B) $\mathbf{d} = [0, 0.3, 0.3]$ $\mathbf{a} = [0.8, 1, 1.4]$ $\alpha = [\frac{\pi}{4}, \frac{7\pi}{18}, 0]$			0	6	–	–	$2(0,1)[0,6]$	Category 2
(C) $\mathbf{d} = [0, 0.5, 0.3]$ $\mathbf{a} = [1, 1, 1.4]$ $\alpha = [\frac{\pi}{4}, \frac{7\pi}{18}, 0]$			2	6	–	–	$(0,0)[2,6]$	Category 5
(D) $\mathbf{d} = [0, 0, 0.6]$ $\mathbf{a} = [1, 1, 1]$ $\alpha = [\frac{\pi}{2}, \frac{4\pi}{9}, 0]$			4	2	–	–	$2(1,0)[4,2]$	Category 4
(E) $\mathbf{d} = [0, 0, 1]$ $\mathbf{a} = [1, 1, 1]$ $\alpha = [\frac{7\pi}{18}, \frac{\pi}{2}, 0]$			0	2	–	–	$2(1,1)[0,2]$	Category 2
(F) $\mathbf{d} = [0, 0, 0.24]$ $\mathbf{a} = [1.2, 0, 0.24]$ $\alpha = [0, \frac{\pi}{2}, 0]$			0	4	∞	0	$2(0,1)[4,0]$ + $2(1,0)[\infty,0]$	Category 3
(G) $\mathbf{d} = [0, 1, 1]$ $\mathbf{a} = [1, 0, 0.5]$ $\alpha = [\frac{\pi}{2}, \frac{\pi}{6}, 0]$			2	2	∞	0	$(0,0)[2,2]$ + $2(1,0)[\infty,0]$	Category 6
(H) $\mathbf{d} = [0, 1, 0]$ $\mathbf{a} = [1, 0, 1]$ $\alpha = [0, \frac{4\pi}{9}, 0]$			0	0	∞	0	$2(1,1)[0,0]$ + $2(1,0)[\infty,0]$	Category 3

Figure S3: Eight non-cuspidal $3R$ robots, along with their proposed categories. The figure presents the D-H parameters $(\mathbf{d}, \mathbf{a}, \alpha)$ of each robot in the first column; robot schematic in second column; singularity plot in the $q_2 - q_3$ slice of joint space in third column; the factors - f_1, f_2 - of the expression of $\det \mathbf{J}$ in conjunction with each factor's total horizontal and vertical turning points (Q^h, Q^v) in the fourth column; the proposed homotopy-based singularity classification in the fifth column; resultant robot category in the sixth column.

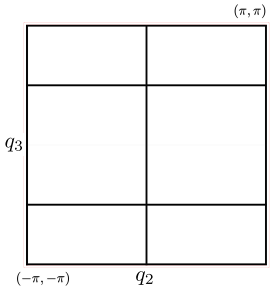


Figure S4: Robot class: $4(1,0)[\infty, 0]$ + $2(0,1)[0, \infty]$, $\mathbf{d} = [0, 0, 0]$ $\mathbf{a} = [1, 0.2, 0.2]$ $\alpha = [0, \frac{\pi}{2}, 0]$

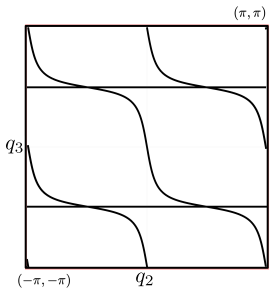


Figure S5: Robot class: $2(1,0)[\infty, 0]$ + $2(1,1)[0, 0]$, $\mathbf{d} = [0, 0.2, 0]$ $\mathbf{a} = [0, 1, 0]$ $\alpha = [0, \frac{4\pi}{9}, 0]$

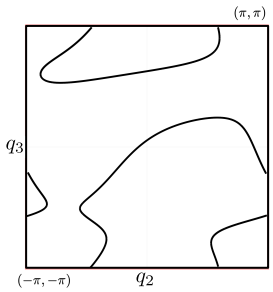


Figure S6: Example of a $(0,0)[2,6]$ branch, $\mathbf{d} = [1, 1, 1]$ $\mathbf{a} = [1.5, 1, 2]$ $\alpha = [\frac{\pi}{4}, \frac{4\pi}{9}, 0]$

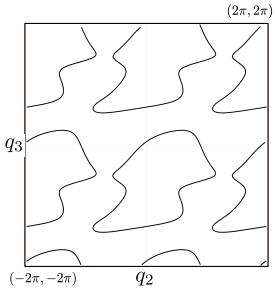


Figure S7: Scaled version of Figure S6.

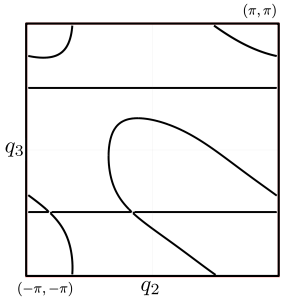


Figure S8: Robot class: $2(1,0)[\infty, 0] + (0,0)[2,2]$, $\mathbf{d} = [0, 1, 1]$ $\mathbf{a} = [1, 0, 3]$ $\alpha = [\frac{\pi}{2}, \frac{\pi}{4}, 0]$

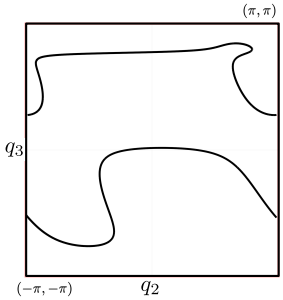


Figure S9: Robot class: $(1,0)[2,4] + (1,0)[2,2]$, $\mathbf{d} = [0, 0.4, 1]$ $\mathbf{a} = [1, 1.2, 1.5]$ $\alpha = [\frac{\pi}{2}, \frac{\pi}{4}, 0]$

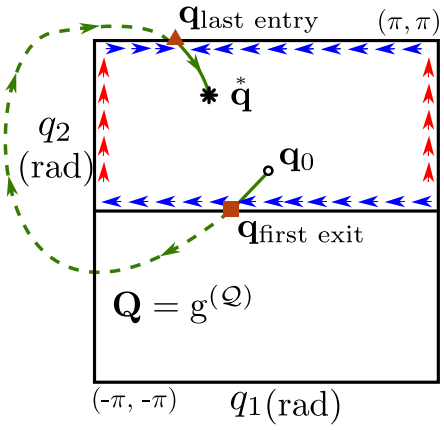


Figure S10: The track cycle in the joint space of a $2R$ robot, that determines the behaviour to reach $\mathbf{q}_{\text{last entry}}$ from any configuration in the boundary region of the relevant aspect.

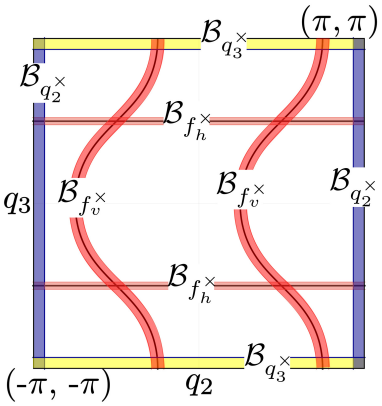


Figure S11: Singularity and joint limit boundaries of a $3R$ robot from Category 3. Visualized in the q_2–q_3 slice of the joint space.

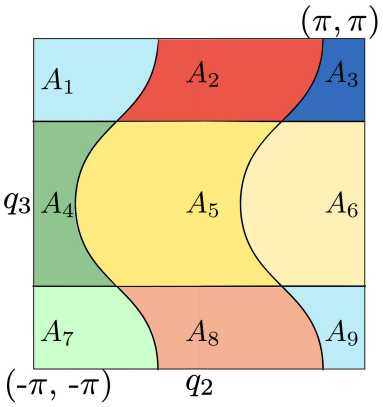


Figure S12: Colored aspect boundaries for the same $3R$ robot with class $2(1,0)[\infty, 0]$ + $2(0,1)[0,2]$ in the q_2–q_3 slice.

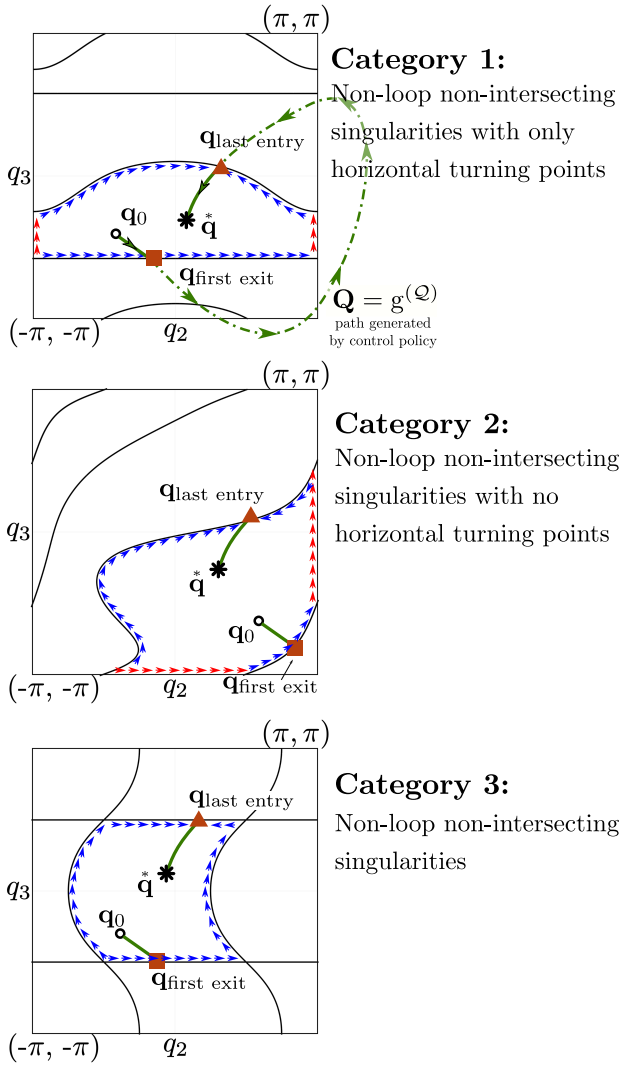


Figure S13: Track cycle for $3R$ robots from Category 1 through 3 (top to bottom) joint space, as discussed in Algorithm 8.

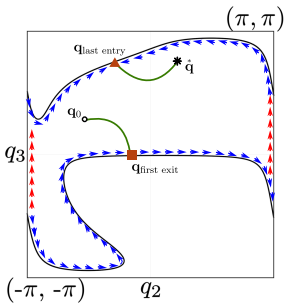


Figure S14: Track cycle for a $3R$ robot from Category 6, with singularity class $(1,0)[2,0] + (1,0)[2,2]$.

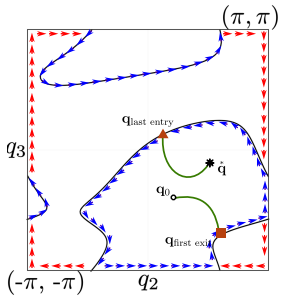


Figure S15: Track cycle in joint space for loop type singularities

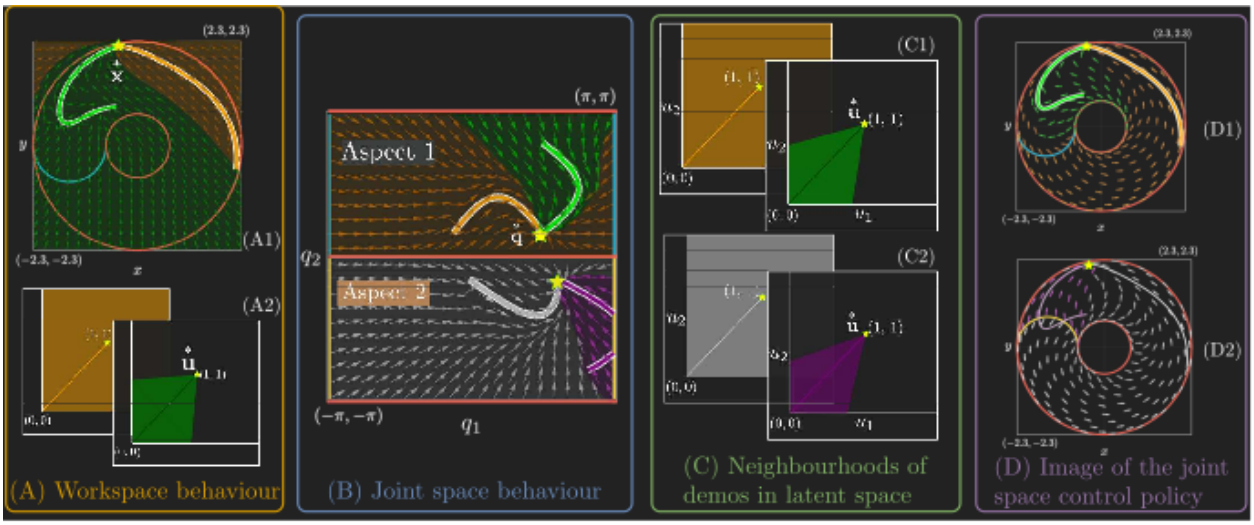


Figure S16: Utilising multiple demonstrations for modelling the user behaviour for a $2R$ robot. (A) represents the workspace behaviour modelled using two demonstrations, wherein (A1) depicts the local neighbourhoods of the corresponding DS embeddings and the resulting workspace behaviour. (A2) represents the latent embedding of these workspace demonstrations, and the cone of influence associated with each DS embedding. The DS embedding associated with the first demo has lower priority, whereas the embedding with second demo has a higher priority and a lower opening angle to avoid overlaps with first demo. (B) represents the joint space trajectories corresponding to the workspace demonstrations, in both aspects of the joint space. The joint space trajectories are validated and trimmed - second demo in Aspect 2 - prior to learning. (B) also depicts the local neighbourhoods of the DS embeddings in respective aspects, as well as the resulting jointspace behaviour. (C1) and (C2) represent the latent embeddings and the cones of influence, for each joint space trajectory in Aspect 1 and Aspect 2 respectively. (D1) and (D2) represent the resulting workspace behaviour when following the joint space control policy, for configurations in Aspect 1 and Aspect 2, respectively.

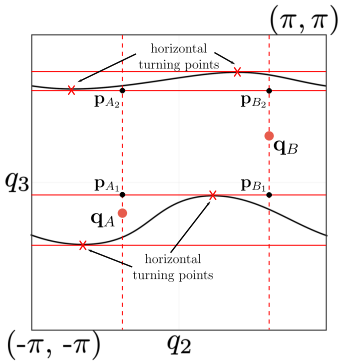


Figure S17: An illustration of the connectivity check for factor class $2(1,0)[2,0]$ in $q_2 - q_3$ slice for a robot from Category 1

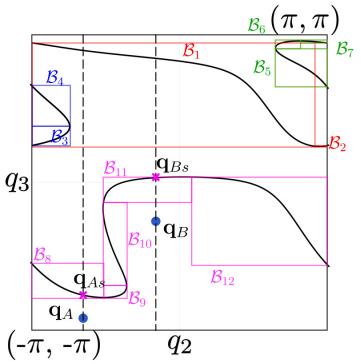


Figure S18: An illustration of the connectivity check for factor class $2(1,0)[2,2]$.

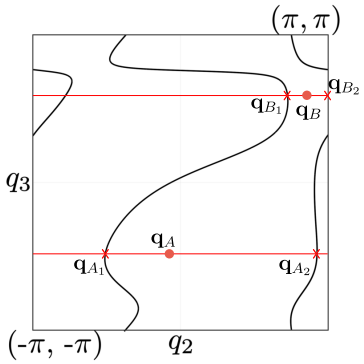


Figure S19: An illustration of the connectivity check for factor class $2(0,1)[0,*]$ in $q_2 - q_3$ slice for robot from Category 2 (B)

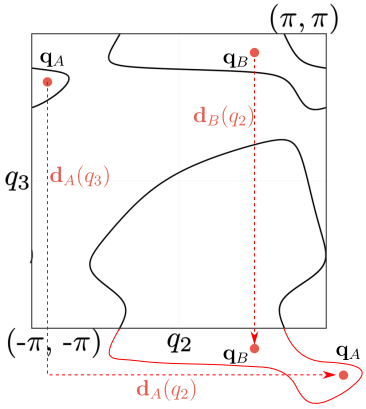


Figure S20: An illustration of the connectivity check for factor class $(0,0)[2,6]$ in $q_2 - q_3$ slice for robot from Category 4 (C)

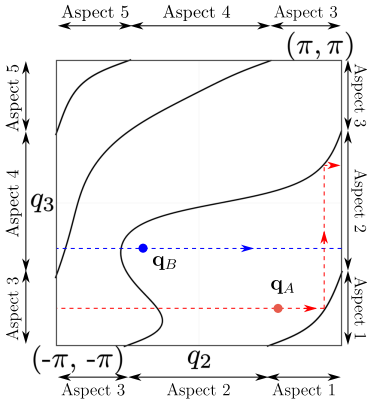
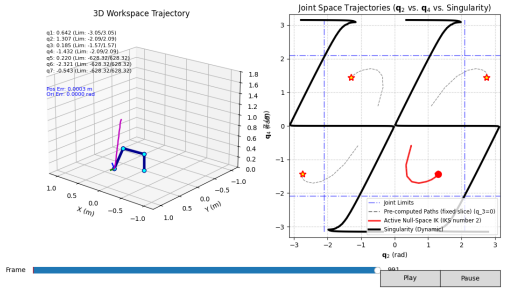


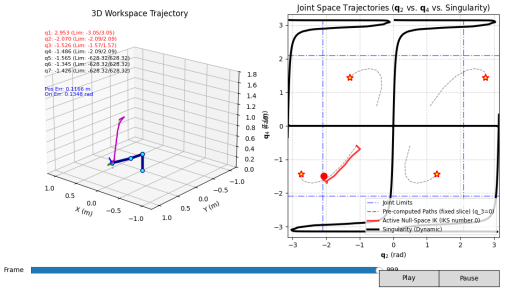
Figure S21: An illustration of the connectivity check for factor class $(1,1)[0,*]$ in $q_2 - q_3$ slice for robot from Category 2 (E).

Robot Category	Connectivity	Track cycle
Category 1 (A)	Algorithm 14	Algorithm 8
Category 2 (B)	Algorithm 16	Algorithm 8
Category 5 (C)	Algorithm 17	Algorithm 11
Category 4 (D)	Algorithm 18	Algorithm 10
Category 2 (E)	Algorithm 19	Algorithm 8
Category 3 (F)	Algorithm 14 + 16	Algorithm 8
Category 6 (G)	Algorithm 14 + 17	Algorithm 11 + 8
Category 3 (H)	Algorithm 14 + 19	Algorithm 8

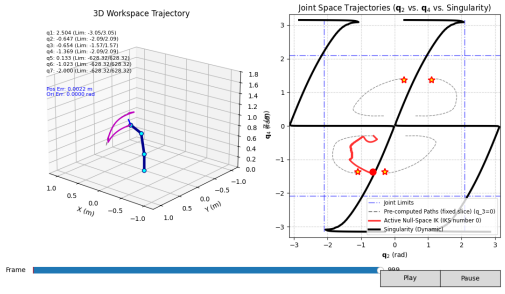
Table S3: Different types of robots and their respective connectivity analysis algorithms and motion near boundaries.



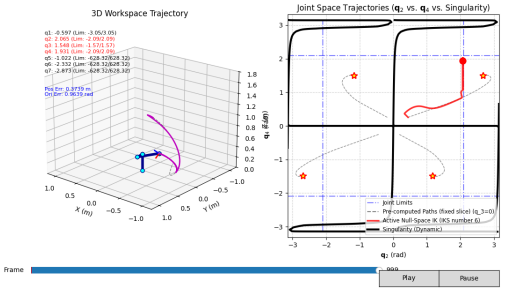
(a) Successful execution: the null-space strategy tracks the demonstrated path and the redundant angle q_R remains essentially constant and well inside joint limits.



(b) Same demonstration, but the null-space evolution leads to end-of-path shaking due to posture oscillations in the arm angle q_3.

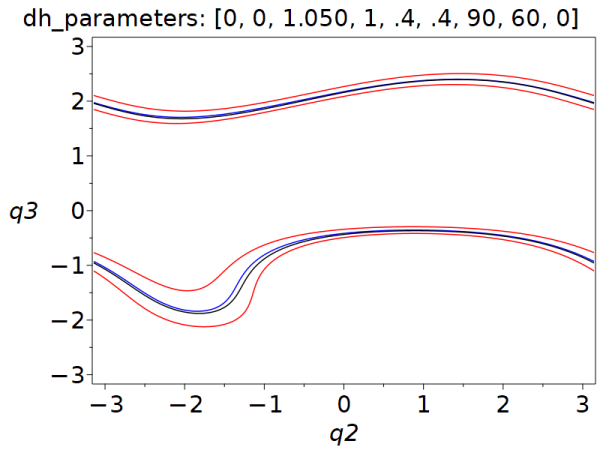


(c) Redundant posture selection yields a q_R that keeps the robot close to singularities along the entire path, although it completes the workspace task.

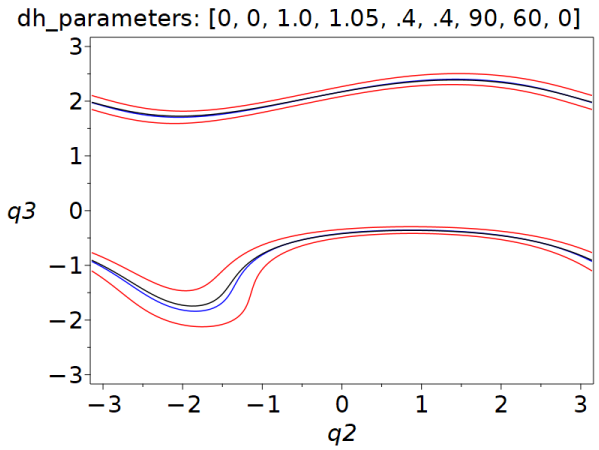


(d) Deviation from the intended behaviour: the null-space policy drives the arm into a configuration where two joint limits are reached, and the path departs from the demonstrated motion.

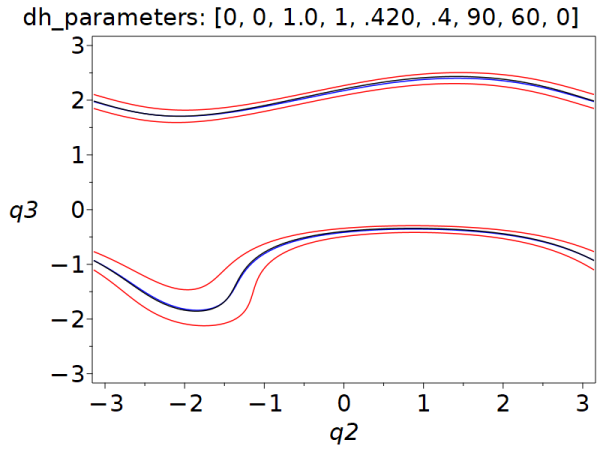
Figure S22: **Failure modes of numerical null-space strategies for redundant robots.** Small changes in initialization and secondary-task tuning can yield qualitatively different outcomes for the same demonstration: from clean execution (A), to undesirable oscillations (B), to trajectories that stay near the singularity (C), or even clear task failure with multiple joints saturating (D).



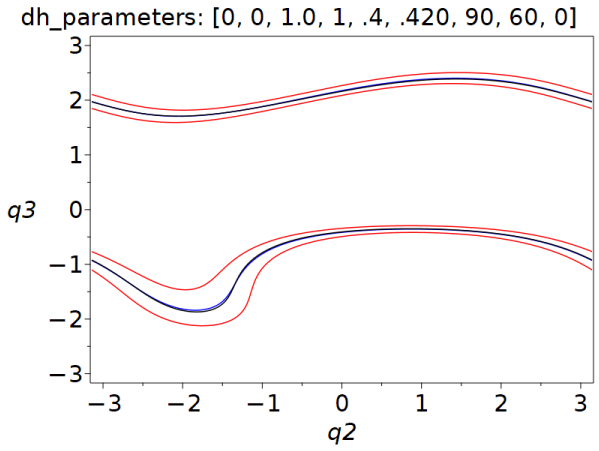
(a) Effect of 5% noise in d_3



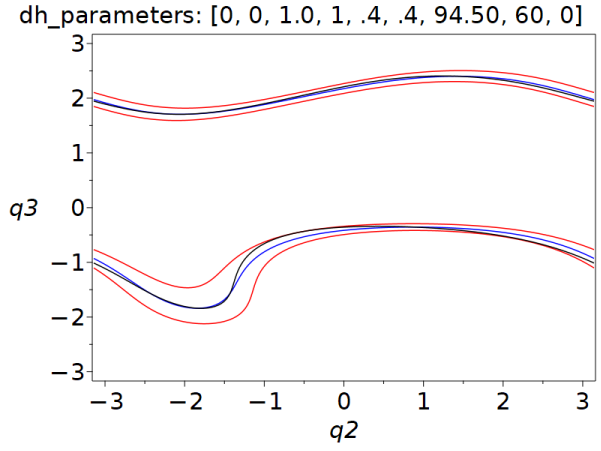
(b) Effect of 5% noise in a_1



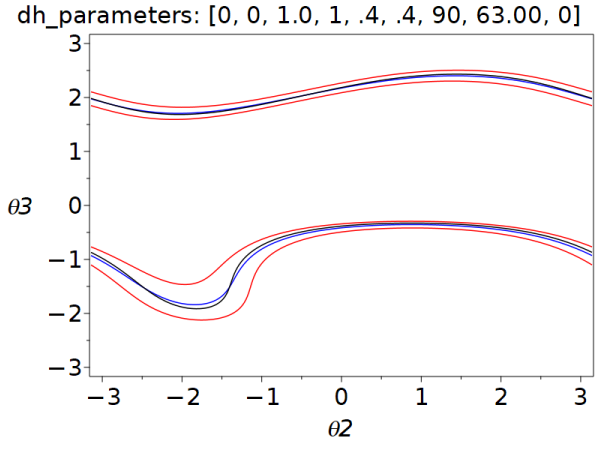
(c) Effect of 5% noise in a_2



(d) Effect of 5% noise in a_3

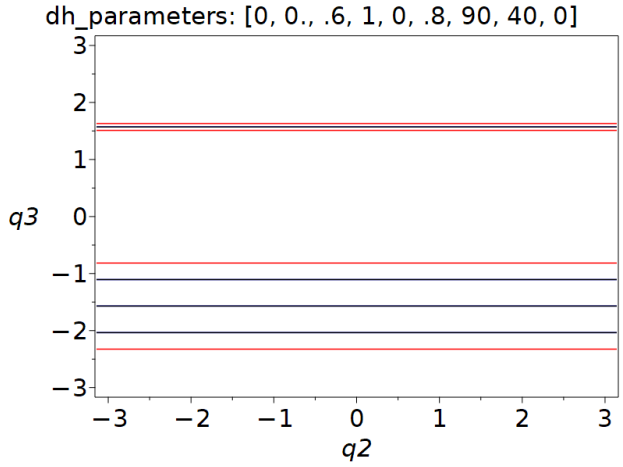


(e) Effect of 5% noise in α_1

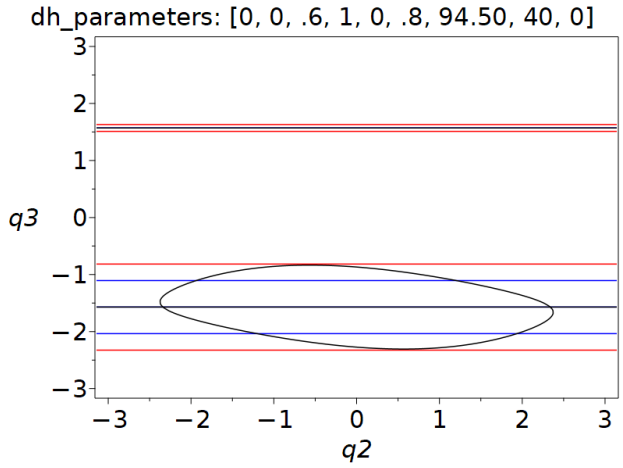


(f) Effect of 5% noise in α_2

Figure S23: The blue curves corresponds to original singularity; the red is a safety margin chosen at $\epsilon = 0.03$. The black curves are the singularity obtained by adding 5% noise in each D-H parameter separately. The first one is in d_3 and progressively in a_1, a_2, a_3, α_1, and α_2. The remaining D-H parameters do not affect singularity (and $d_2 = 0$ for this case). The D-H parameters of the base architecture is : $\mathbf{d} = [0, 0, 1]$, $\mathbf{a} = [1, 0.4, 0.4]$, $\alpha = [\pi/2, \pi/3, 0]$



(a) Effect of 5% noise in d_3



(b) Effect of 5% noise in $alpha_1$

Figure S24: Effect of kinematic parameter perturbations on the singularity layout. Left: a 5% perturbation in d_3 (black) produces no significant change with respect to the nominal singularity (blue), and remains well within the safety margin. Right: a 5% perturbation in α_1 alters the singularity curve more noticeably, but the perturbed singularity still lies entirely inside the padded band defined by $|\det \mathbf{J}| < 10^{-3}$ (red), so the controller continues to treat this region as forbidden.

Caption for Movie S1. Overview of behaviour transfer to anthropomorphic robots This video shows the challenges in transfer of demonstrated behaviour, briefly introduces the redundancy parametrisation in $7R$ robots. Furthermore, few kinematic properties for $3R$ robots are next illustrated. Lastly, the resulting behaviour on different commercial robots is presented.

Caption for Movie S2. Different types of singularities in non-cuspidal $3R$ robots The joint space singularities for multiple cases of non-cuspidal robots with their respective DH parameters are presented.

Caption for Movie S3. Transferred behaviour on KUKA IIWA The demonstrated characters from SCIENCE are used to learn control policies for the commercially available anthropomorphic robot.

Caption for Movie S4. Transferred behaviour on MAiRA robot from Neura Robotics The demonstrated characters from SCIENCE are used to learn control policies for the commercially available anthropomorphic robot.

Caption for Movie S5. Learning control policies from multiple demonstrations The video presents three cases for a $2R$ robot with varying number of user demonstrations to learn the behaviour from.

Caption for Movie S6. Experimental setup and cross-platform skill transfer The video presents the physical experimental setup used with three distinct robotic manipulators and illustrates how a single learned DS-based control policy, obtained from a reference demonstration, is executed across them. It highlights how the same task is reproduced across platforms while respecting each robot's kinematic constraints and feasible workspaces.