arXiv:2604.20468v3 [cs.RO] 27 Aug 2026

MOMO: A framework for seamless physical, verbal, and graphical robot skill learning and adaptation

Markus Knauer[1,2], Edoardo Fiorini[1], Maximilian Mühlbauer[2,1], Stefan Schneyer[1,2], Promwat Angsuratanawech[1,2], Florian Samuel Lay[1], Timo Bachmann[1], Samuel Bustamante[1,2], Korbinian Nottensteiner[1], Freek Stulp[1], Alin Albu-Schäffer[1,2], João Silvério[1], and Thomas Eiband[1]

***Abstract*— Industrial robot applications require increasingly flexible systems that non-expert users can easily adapt for varying tasks and environments. However, different adaptations benefit from different interaction modalities. We present an interactive framework that enables robot skill adaptation through three complementary modalities: kinesthetic touch for precise spatial corrections, natural language for high-level semantic modifications, and a graphical web interface for visualizing geometric relations and trajectories, inspecting and adjusting parameters, and editing via-points by drag-and-drop. The framework integrates five components: energy-based human-intention detection, a tool-based LLM architecture (where the LLM selects and parameterizes predefined functions rather than generating code) for safe natural language adaptation, Kernelized Movement Primitives (KMPs) for motion encoding, probabilistic Virtual Fixtures for guided demonstration recording, and ergodic control for surface finishing. We demonstrate that this tool-based LLM architecture generalizes skill adaptation from KMPs to ergodic control, enabling voice-commanded surface finishing. Validation on a 7-DoF torque-controlled robot at the Automatica 2025 trade fair demonstrates the practical applicability of our approach in industrial settings.**

***Index Terms*—Human-Robot Interaction, Multi-Modal Interfaces, Natural Language, Imitation Learning, Ergodic control**

I. Introduction

Modern industrial robot applications require flexible systems that can be easily adapted for varying tasks and environments, requiring frequent skill adjustments in response to changing product specifications, environmental conditions, or operational requirements [1]. However, adapting robot skills often requires manual reprogramming, a complex and time-consuming process accessible only to expert programmers [2], [3]. This creates a bottleneck: non-expert operators lack intuitive tools to quickly modify robot behaviors, resulting in extended downtime and reduced flexibility. Making robot skill adaptation accessible to non-experts is therefore essential for deploying adaptive robotic systems in industrial settings.

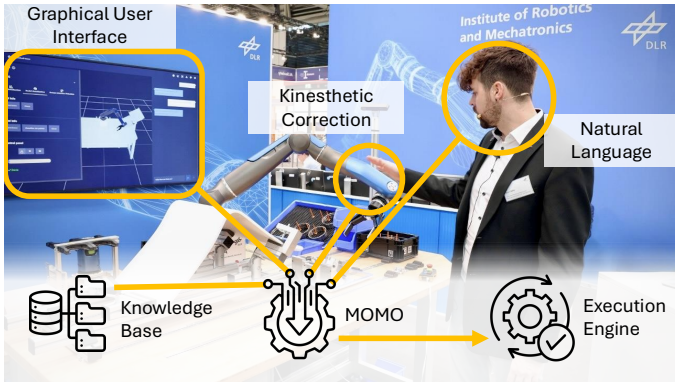


Fig. 1. Overview of the framework. The three interaction modalities (physical, verbal, and graphical) enable complementary skill adaptation, demonstrated on industrial robot tasks at a trade fair. The **MOMO** (**Mo**tion **Mo**dulation) module manages the different inputs and adapts the skill accordingly. The Execution Engine deploys the resulting trajectories on the robot.

Recent advances in Large Language Models (LLMs) show potential for more intuitive robot skill adaptation through natural language interaction, allowing users to communicate desired modifications in everyday language rather than specialized programming syntax. Unlike traditional teach-pendant programming, end-users can express their intentions conversationally, with the system interpreting and translating these into appropriate robot actions. However, LLMs are not optimal for all adaptation scenarios. While they excel at high-level semantic concepts, they may be unnecessarily indirect for fine-tuning or simple adjustments. Different adaptation tasks thus benefit from different interaction modalities: kinesthetic correction provides natural spatial feedback, graphical interfaces enable parameter inspection and adjustment, and natural language allows for intuitive semantic-level modifications. Rather than adopting a one-size-fits-all approach, the choice of interaction method should rest with the user. Furthermore, users have different strengths in using these interaction methods. For instance, a user might feel uncomfortable with physically correcting a moving robot and may prefer natural language interaction instead.

We propose an interactive framework enabling robot skill adaptation through multiple complementary modalities: kinesthetic feedback for force-guided corrections, natural language for high-level semantic modifications, and graphical interfaces for parameter adjustment and visualization. Fine-tuning

This work was partially funded by the DLR project "ASPIRO"; the European Union's Horizon Research and Innovation Program under Grant 101136067 (INVERSE); the Federal Ministry for Economic Affairs and Climate Protection with DARP funds based on a decision by the German Bundestag and by the European Union - NextGenerationEU; and partially supported by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG).

[1] All authors are with the German Aerospace Center (DLR), Institute of Robotics and Mechatronics (RMC), Münchener Str. 20, 82234 Weßling, Germany. {first}.{last}@dlr.de

[2] Markus Knauer, Maximilian Mühlbauer, Stefan Schneyer, Promwat Angsuratanawech, Samuel Bustamante and Alin Albu-Schäffer are also with the School of Computation, Information and Technology (CIT), Technical University of Munich (TUM), Arcisstr. 21, 80333 Munich, Germany. m.knauer@tum.de

Digital Object Identifier (DOI): 10.1109/RAP.2026.3724466

and small geometric adjustments are naturally performed via kinesthetic correction, larger task-level adaptations (obstacle avoidance, trajectory modifications) are efficiently expressed through natural language, and the graphical interface enables structured parameter inspection and adjustment. The main contributions of our work are:

1) A unified multi-modal robot skill adaptation framework combining physical interaction, natural language (voice and text), and a web-based graphical interface with drag-and-drop via-point editing, enabling end-users to select the modality best suited to each adaptation task and their personal strengths;
2) Integration of five complementary components: (i) energy-based human intention detection for kinesthetic corrections [4], (ii) a tool-based LLM architecture for safe natural language adaptation [5], (iii) KMP-based skill representation with online via-point adaptation [6], (iv) probabilistic Virtual Fixtures for guided demonstration recording [7], [8], and (v) ergodic control for surface finishing [9], demonstrating that the tool-based LLM architecture generalizes across skill representations;
3) An open-source release of the IROSA and KMP/TP-KMP with runnable examples (Sec. S-I).

We validate the framework on a 7-DoF torque-controlled robot performing industrial manipulation tasks. By combining established algorithms with deployment on real industrial hardware, we provide an overview of current trends in intuitive programming of cobots, using our Automatica 2025 trade fair demonstrator as an integrated case study, and demonstrate that interactive multi-modal robot adaptation is viable in production environments and accessible to non-expert users.

II. Related Work

Multi-modal interfaces for robot programming have gained attention in making skill acquisition more accessible. Early work focused on single modalities: kinesthetic teaching [2], natural language commands [10], or graphical interfaces [11]. Advances in Learning from Demonstration [3] have enabled flexible skill representations that can be adapted online. Large Language Models have been applied to robot control through code generation [12] and tool-augmented approaches [13], [14]. OVITA [15] uses LLMs for open-vocabulary trajectory adaptation via code synthesis, trading safety for flexibility. Kernelized Movement Primitives [6] provide a probabilistic, non-parametric framework for encoding demonstrations that naturally supports via-point-based adaptation. Physical human-robot interaction for skill correction has been explored using energy-based methods [16], enabling robots to distinguish intentional corrections from task-related forces. Virtual Fixtures [17] provide haptic guidance during teleoperation and demonstration recording, reducing operator workload and improving demonstration consistency. Ergodic control [18], [19] offers an alternative to movement primitives for coverage tasks, enabling online adaptation to perturbations. The Spectral Multiscale Coverage (SMC) approach [18] achieves ergodic coverage by minimizing a cost function that compares the Fourier coefficients of the target distribution with those of the observed coverage, generating control actions online at each time step. Despite these advances, no existing framework unifies physical, verbal, and graphical interaction modalities for both skill acquisition and adaptation.

III. Methodology

We present a unified framework that enables robot skill acquisition and adaptation through three interaction modalities: physical interaction, natural language, and graphical interface. Figure 1 shows the system architecture and the five integrated components. At the core, the **MOMO** (**Mo**tion **Mo**dulation) module manages the different inputs and adapts the skill accordingly. The Execution Engine then deploys the resulting trajectories on the torque-controlled robot. For motion encoding, Kernelized Movement Primitives (KMPs) [6] encode demonstrations as a probabilistic mapping from input $\boldsymbol{s}$ to output $\boldsymbol{\xi}$. Given reference data $\mathcal{D} = \{\boldsymbol{s}_n, \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^{N}$, with means $\boldsymbol{\mu}_n$ and covariances $\boldsymbol{\Sigma}_n$ extracted via Gaussian Mixture Models from demonstrations, the predictive mean evaluates to:

$$\mathbb{E}[\boldsymbol{\xi}(\boldsymbol{s}^*)] = \boldsymbol{k}^*(\boldsymbol{K} + \lambda_1 \boldsymbol{\Sigma})^{-1}\boldsymbol{\mu} \tag{1}$$

where λ_1 is a small regularization factor, $\boldsymbol{K}$ and $\boldsymbol{k}^*$ are kernel matrices measuring distances between training points and between training and test points, respectively (see [6] for details). Via-points $\bar{\mathcal{D}} = \{\bar{s}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}}\}$ can be added online after learning to adapt the trajectory, with small $\bar{\boldsymbol{\Sigma}}$ enforcing precise passage through $\bar{\boldsymbol{\mu}}$. This framework has been extended with interactive incremental learning and local trajectory modulation for generalizable skill adaptation [20]. For surface finishing, we extended SMC [9] by incorporating a variable tool contact area that adapts to nontrivial surface geometries, and learning both desired coverage distributions and preferred motion directions from human demonstrations.

A. Skill Acquisition with Virtual Fixtures

A Virtual Fixtures component [7], [8] combines dynamical system-based, position-based, and visual servoing fixtures. These fixtures provide probabilistic wrenches fused via Product of Experts arbitration. Variable impedance control [8] with coupled stiffness matrices guides users during demonstration recording, reducing workload while maintaining precision, with closed-loop passivity ensured by [21]. During demonstration recording, these probabilistic Virtual Fixtures assist the operator with haptic guidance, for instance while picking or placing objects. We use position-based trajectory and dynamical system-based velocity fixtures learned from demonstrations with the variable impedance formulation [8], [21]. The resulting fixtures, as shown in Fig. S4 in the supplementary material, support the operator in precise alignment, thus reducing operator workload and improving demonstration consistency, particularly for industrial tasks requiring high precision. Subsequent corrections of the encoded motion are still possible through physical interaction with the impedance controller during execution.

B. Physical Interaction Modality

For intuitive fine-tuning during execution, users can physically interact with the robot to correct trajectories. We employ an energy-tank-based intention detection [4], where an energy tank is assigned to each degree of freedom (DoF) to track interaction dynamics, mapping the user-injected energy to an intention index in the range $[0, 1]$. The tank's dissipation is adapted by the KMP uncertainty. High-variance regions dissipate energy faster, yielding shorter compliant windows for corrections, while low-variance regions dissipate more slowly, giving the operator more time for fine, precise corrections. The robot achieves compliance and facilitates physical contact in the relevant DoFs through varying stiffness based on the intention index (Fig. S9). Once the index exceeds a predefined threshold, DoF-specific via-points are automatically inserted into the underlying KMP. This enables precise, local corrections without affecting distant trajectory segments.

C. Natural Language Modality

Users adapt skills through voice or text commands processed by a tool-based LLM architecture [5], where the LLM selects and parameterizes predefined functions rather than generating executable code. This maintains strict separation between language understanding and robot control. The IROSA is accessible via an integrated chat window (Fig. S3 in the supplementary material) and supports skill selection and adaptation through natural language. The LLM selects appropriate tools and extracts parameters from the instruction and environmental context. Each tool is pre-validated by the developer with defined parameter bounds and safety constraints, ensuring that only verified operations reach the robot.

Adapt KMP-Based Skills: For KMP-based skill adaptation, the available tools include speed modulation (e.g., "Go slower near the box"), trajectory correction via via-point insertion (e.g., "Move more to the left at the start"), and obstacle avoidance using signed distance field repulsion (e.g., "Avoid the blue box").

Parameterize Ergodic Surface Finishing: We extend the tool-based LLM architecture to ergodic surface finishing [9], which provides a framework for generating trajectories that distribute coverage intensity according to task-specific spatial objectives. The approach learns desired coverage distributions and motion preferences from demonstrations and incorporates tool imprint models for accurate coverage on complex geometries. Because trajectories are generated online, the approach naturally supports human intervention and parameter adaptation. We define tools for start/stop execution, tool velocity, impedance stiffness, and contact force control.

This demonstrates that the tool-based architecture is not tied to a specific skill representation, as shown in Fig. 2 and Fig. S13 in the supplementary material, where the same chat interface controls a surface finishing task instead of a KMP-based motion. The same LLM interprets user intent and selects appropriate tools, while the underlying control method determines how those tools affect the robot behavior.

D. Graphical Interface Modality

A web-based user interface provides visualization, parameter inspection, and adjustment for different skill representations. It features the visualization of the workcell and a virtual robot arm. Additionally, an interaction panel is organized into four functional tabs: *Default Fixtures*, *Data Visualization*, *Model Visualization*, and *Human Intention Detection*. For more general information about our Human Factory Interface (HFI), see Sec. S-II in the supplementary material. The Data Visualization tab (Fig. S5 in the supplementary material) allows inspection of recorded demonstrations, displaying the Cartesian trajectory in the virtual environment. Users can select individual demonstrations and switch between visualization modes including Cartesian position, Cartesian pose with position and orientation, force profile, and joint configuration. The joint configuration view (Fig. S6) renders a robot digital twin at the selected time step, enabling intuitive inspection of the robot pose throughout the demonstration. The Model Visualization tab (Fig. S7) displays the learned KMP mean and covariance, providing a direct view of the skill's underlying data representation. Users can compare the original and updated model side by side, e.g., to verify the effect of via-point insertions. The tab includes an edit mode for interactive trajectory modification and a trajectory control panel for starting execution and monitoring progress. During execution (Fig. S8), a digital twin of the robot moves in sync with the real robot, and the progress bar indicates the current execution state. Users add via-points by drag-and-drop on the trajectory (Fig. S12), and adapt or delete existing ones via a right-click menu (Fig. S11); the adapted trajectory (yellow) is shown against the original (blue) in Fig. S10. The interface provides visual feedback that is difficult to convey through physical or verbal interaction: users can see exactly where via-points will affect the trajectory and verify that the adapted path meets requirements before execution.

E. Complementary Modalities

Each modality offers distinct advantages:

- **Physical interaction** excels at precise spatial corrections. Users directly see and feel the robot's motion and apply corrections intuitively. However, it requires physical proximity and good physical guidance skills, and is limited to local adjustments.
- **Natural language** efficiently expresses high-level semantic modifications ("avoid the obstacle") and references to perceptual features ("move closer to the red box"). It is less suited for precise geometric specifications.
- **Graphical interface** enables methodical parameter adjustment and provides visual confirmation of planned trajectories. Users can see all via-points simultaneously and adjust them with a precision that is difficult to achieve through verbal description or physical interaction.

Users freely switch between modalities based on the task at hand and personal skills. An operator might first use voice commands to add obstacle avoidance, then fine-tune the trajectory via kinesthetic correction, and finally verify changes through the graphical interface.

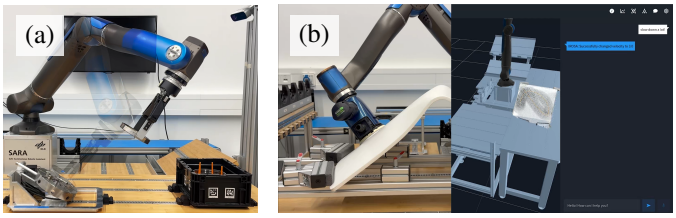


Fig. 2. (a) Bearing ring insertion task: the transparent robot shows original speed, the opaque one shows the adapted trajectory after the voice command "slow down between box and station." (b) Ergodic surface finishing task with the LLM chat interface issuing a similar slow-down command, alongside a virtual twin of the robot workcell.

IV. Evaluation

We validate our framework through live demonstration at the Automatica 2025 trade fair, where visitors could interact with the system using all three modalities. A video of the demonstration is provided in the supplementary materials.

A. Experimental Setup

The framework operates a 7-DoF torque-controlled robot shown in Fig. 1, performing industrial tasks including a bearing ring insertion and surface finishing (Fig. 2). The system runs on industrial hardware with a local LLM backend (Qwen2.5-VL-72B-Instruct) for natural language processing, ensuring data privacy and low latency. Screenshots of the web-based graphical interface are provided in Sec. S-III in the supplementary material.

B. Operational Workflow

The complete workflow consists of the following phases.

a) Skill Acquisition: An operator records demonstrations via kinesthetic teaching, assisted by probabilistic Virtual Fixtures [8] that provide haptic guidance. The fixtures combine dynamical system-based guidance for coarse trajectory following with position-based constraints for precise waypoint passage, reducing teaching effort while ensuring consistent demonstration quality.

b) Skill Execution: The learned KMP generates smooth trajectories adapted to the current object locations. For surface finishing, ergodic control [9] ensures uniform coverage while respecting learned motion preferences. Both representations support online adaptation during execution.

c) Skill Adaptation: When task requirements change (new obstacles, shifted targets, different surface coverage requirements), users adapt the skill through one or more of their preferred modalities, with iterative cycles of execution and adaptation.

C. Qualitative Observations

During the trade fair demonstration, we observed clear patterns in modality preferences among visitors:

- **Fine-tuning**: Users intuitively used physical interaction for small spatial adjustments, benefiting from direct proprioceptive feedback.
- **Semantic modifications**: Voice commands were preferred for high-level changes such as "avoid the red box" or "move higher at the start", which are cumbersome to express through physical interaction.
- **Parameter inspection**: The graphical interface was frequently used to visualize trajectories before and after modifications (Fig. S10 in the supplementary material), and to systematically adjust via-point positions through drag-and-drop (Fig. S12).

V. Discussion

The framework demonstrates that multi-modal interaction makes robot skill adaptation accessible to non-experts, as demonstrated with trade fair visitors. Rather than forcing users to adopt a single interaction paradigm, the system allows selection of the method best suited to each task and user-specific skills: physical interaction for precise local corrections, natural language for semantic-level modifications, and the graphical interface for systematic exploration and verification.

A key design decision is integrating multiple approaches that complement each other throughout the skill lifecycle: Virtual Fixtures [8] assist during demonstration recording, KMPs [6] provide a flexible skill representation supporting via-point adaptation from all modalities, and the tool-based LLM architecture [5] ensures safe natural language processing by constraining the LLM to predefined functions. While the current tool set is bounded, using an LLM rather than conventional keyword-based voice recognition [22] provides tolerance to imprecise or varied phrasing, natural context handling, and easy extension, as new tools can be added without retraining a domain-specific speech model.

Extending the LLM tool architecture to ergodic control applications [9] validates generalizability (Fig. S13 in the supplementary material). The same natural language interface that adapts KMP-based pick-and-place skills (via-point insertion, obstacle avoidance, speed modulation) also controls surface finishing parameters (coverage velocity, contact force, impedance stiffness). This suggests that the tool-based architecture can accommodate diverse skill representations beyond movement primitives.

The natural language interface is currently limited to predefined tool functions; truly open-ended adaptations require code generation approaches that sacrifice safety guarantees. Furthermore, physical interaction modalities require force-torque sensing hardware. An end-effector sensor suffices but limits interaction to the last link, whereas joint torque sensors enable whole-body contact detection. Lastly, Virtual Fixtures assist but do not guarantee demonstration quality, and poor demonstrations still propagate to learned skills.

VI. Conclusion

We presented a framework for interactive robot skill acquisition and adaptation, combining three interaction modalities: physical interaction for precise spatial corrections, natural language for high-level semantic modifications, and a web-based graphical interface for parameter inspection and adjustment. The framework integrates five complementary components that

support each other throughout the skill lifecycle. Live demonstration at Automatica 2025 on a 7-DoF torque-controlled robot validated practical applicability, allowing visitors to naturally switch between modalities based on their needs. Future work will investigate automatic modality suggestion based on task characteristics and conduct a formal user study to quantify modality preferences.

REFERENCES

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces, and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018. [Online]. Available: https://doi.org/10.1016/j.mechatronics.2018.02.009

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot Programming by Demonstration*. Springer, 2008, pp. 1371–1394. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5_60

[3] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 297–330, 2020. [Online]. Available: https://doi.org/10.1146/annurev-control-100819-063206

[4] E. Fiorini, M. Knauer, T. Eiband, M. Iskandar, and J. Silvério, "Interactive learning via physical human feedback using uncertainty-aware energy tanks," *IEEE Robotics and Automation Letters (RA-L)*, 2026, early Access. [Online]. Available: https://ieeexplore.ieee.org/document/11425762

[5] M. Knauer, S. Bustamante, T. Eiband, A. Albu-Schäffer, F. Stulp, and J. Silvério, "IROSA: Interactive robot skill adaptation using natural language," *IEEE Robotics and Automation Letters (RA-L)*, 2026, early Access. [Online]. Available: https://ieeexplore.ieee.org/document/11425760

[6] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *International Journal of Robotics Research (IJRR)*, vol. 38, no. 7, pp. 833–852, 2019. [Online]. Available: https://doi.org/10.1177/0278364919846363

[7] M. Mühlbauer, T. Hulin, B. Weber, S. Calinon, F. Stulp, A. Albu-Schäffer, and J. Silvério, "A probabilistic approach to multi-modal adaptive virtual fixtures," *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, no. 6, pp. 5298–5305, 2024. [Online]. Available: https://doi.org/10.1109/LRA.2024.3384759

[8] M. Mühlbauer, B. Weber, S. Calinon, F. Stulp, A. Albu-Schäffer, and J. Silvério, "A unified framework for probabilistic dynamic-, trajectory- and vision-based virtual fixtures," *arXiv preprint arXiv:2506.10239*, 2025. [Online]. Available: https://arxiv.org/abs/2506.10239

[9] S. Schneyer, K. Nottensteiner, A. Albu-Schäffer, F. Stulp, and J. Silvério, "An ergodic approach to robotic surface finishing with learned motion preferences," *IEEE Transactions on Robotics (T-RO)*, 2025. [Online]. Available: https://doi.org/10.1109/TRO.2025.3641752

[10] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2011. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/7979

[11] S. Alexandrova, Z. Tatlock, and M. Cakmak, "RoboFlow: A flow-based visual programming language for mobile manipulation tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5537–5544. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7139973

[12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500. [Online]. Available: https://doi.org/10.1109/ICRA48891.2023.10160591

[13] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html

[14] Z. Wang, Z. Cheng, H. Zhu, D. Fried, and G. Neubig, "What are tools anyway? a survey from the language model perspective," in *Conference on Language Modeling (COLM)*, 2024. [Online]. Available: https://openreview.net/forum?id=Xh1B90iBSR

[15] A. Maurya, T. Ghosh, A. Nguyen, and R. Prakash, "OVITA: Open-vocabulary interpretable trajectory adaptations," *IEEE Robotics and Automation Letters (RA-L)*, vol. 10, no. 11, pp. 11 054–11 061, 2025. [Online]. Available: https://doi.org/10.1109/LRA.2025.3606309

[16] M. Khoramshahi and A. Billard, "A dynamical system approach for detection and reaction to human guidance in physical human-robot interaction," *Autonomous Robots (AuRo)*, vol. 44, no. 8, pp. 1411–1429, 2020. [Online]. Available: https://doi.org/10.1007/s10514-020-09934-9

[17] L. B. Rosenberg, "Virtual fixtures: Perceptual tools for telerobotic manipulation," in *Proceedings of IEEE Virtual Reality Annual International Symposium (VRAIS)*, 1993, pp. 76–82. [Online]. Available: https://doi.org/10.1109/VRAIS.1993.380795

[18] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4-5, pp. 432–442, 2011. [Online]. Available: https://doi.org/10.1016/j.physd.2010.10.010

[19] S. Ivić, B. Crnković, and I. Mezić, "Ergodicity-based cooperative multiagent area coverage via a potential field," *IEEE Transactions on Cybernetics (TCYB)*, vol. 47, no. 8, pp. 1983–1993, 2017. [Online]. Available: https://doi.org/10.1109/TCYB.2016.2634400

[20] M. Knauer, A. Albu-Schäffer, F. Stulp, and J. Silvério, "Interactive incremental learning of generalizable skills with local trajectory modulation," *IEEE Robotics and Automation Letters (RA-L)*, vol. 10, no. 4, pp. 3398–3405, 2025. [Online]. Available: https://doi.org/10.1109/LRA.2025.3542209

[21] M. Mühlbauer, N. Werner, R. Balachandran, T. Hulin, J. Silvério, F. Stulp, and A. Albu-Schäffer, "Passive variable impedance for shared control," *arXiv preprint arXiv:2604.20557*, 2026. [Online]. Available: https://arxiv.org/abs/2604.20557

[22] K. Zinchenko, C.-Y. Wu, and K.-T. Song, "A study on speech recognition control for a surgical robot," *IEEE Transactions on Industrial Informatics (TII)*, vol. 13, no. 2, pp. 607–615, 2017. [Online]. Available: https://doi.org/10.1109/TII.2016.2625818

[23] F. Schmidt, "Links and nodes: A real-time middleware for distributed robotic systems," 2020, open-source, GPLv3. Documentation: https://links-and-nodes.readthedocs.io. [Online]. Available: https://gitlab.com/links_and_nodes/links_and_nodes

[24] N. Hogan, "Impedance control: An approach to manipulation," in *1984 American Control Conference*. IEEE, 07 1984.

[25] M. Iskandar, C. Ott, A. Albu-Schäffer, B. Siciliano, and A. Dietrich, "Hybrid force-impedance control for fast end-effector motions," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 7, pp. 3931–3938, 2023.

[26] M. Iskandar, O. Eiberger, A. Albu-Schäffer, A. De Luca, and A. Dietrich, "Collision detection, identification, and localization on the DLR SARA robot with sensing redundancy," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3111–3117.

[27] M. J. A. Zeestraten, I. Havoutis, J. a. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on Riemannian manifolds," *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 3, pp. 1240–1247, 2017.

[28] T. Bachmann, O. Eiberger, T. Eiband, F. Lay, P. Angsuratanawech, I. Rodriguez, P. Lehner, F. Stulp, and K. Nottensteiner, "Task-specific reconfiguration of variable workstations using automated planning of workcell layouts," in *ISR Europe 2023; 56th International Symposium on Robotics (ISR)*, 2023, pp. 250–257.

[29] P. M. Schäfer, F. Steinmetz, S. Schneyer, T. Bachmann, T. Eiband, F. S. Lay, A. Padalkar, C. Sürig, F. Stulp, and K. Nottensteiner, "Flexible robotic assembly based on ontological representation of tasks, skills, and resources," in *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, vol. 18, no. 1, 2021, pp. 702–706. [Online]. Available: https://doi.org/10.24963/kr.2021/73

[30] T. Eiband, F. Lay, K. Nottensteiner, and D. Lee, "Unifying skill-based programming and programming by demonstration through ontologies," *Procedia Computer Science*, vol. 232, pp. 595–605, 2024. [Online]. Available: https://doi.org/10.1016/j.procs.2024.01.059

[31] T. Eiband, C. Willibald, I. Tannert, B. Weber, and D. Lee, "Collaborative programming of robotic task decisions and recovery behaviors," *Autonomous Robots (AuRo)*, vol. 47, no. 2, pp. 229–247, 2023. [Online]. Available: https://doi.org/10.1007/s10514-022-10062-9

Supplementary Material

A framework for seamless physical, verbal, and graphical robot skill learning and adaptation

S-I. Implementation Details

A. Software Architecture and Technology Stack

Fig. S1 shows the software architecture and data flow. The system is built on the following technology stack:

Middleware: All components communicate via *links and nodes* (LN) [23], an open-source[1] (GPLv3) real-time-capable middleware for distributed robotic systems. LN is both a process manager with a graphical user interface and a communication framework providing publish-subscribe (via topics) and request-response (via services) communication patterns. A WebSocket bridge translates the LN protocol to JSON-over-WebSocket, enabling browser-based frontends.

MOMO Service Provider: The central orchestrator registers 16 services over LN: 13 motion modulation services (trajectory data retrieval, model visualization, trajectory execution control, via-point management from all three modalities, and human intention detection) and 3 LLM services (query input, answer retrieval, speech-to-text). It publishes real-time status on two topics: trajectory execution progress and LLM notification events.

LLM Backend: Qwen2.5-VL-72B-Instruct is deployed locally via vLLM on GPU hardware. It was already available on our on-premise inference infrastructure and, among the models we tried [5], it selected tools most reliably while supporting OpenAI-compatible function calling, on which the tool-based architecture depends. Its vision-language capability is not exercised in the pipeline reported here, as object positions and semantic labels are passed to the LLM as text, not as images; a vision-language model was nonetheless retained so that visual grounding can be added later without changing the tool interface. The LLM receives tool definitions as typed function schemas (compatible with the OpenAI function calling API) and returns structured tool calls with validated parameters. An example script for running the LLM backend against the IROSA is included in the open-source release (Tab. S-III).

STT: Speech-to-text transcription via OpenAI Whisper, deployed as a FastAPI service.

Robot Handler (corresponding to the Execution Engine in the graphical abstract of the main paper)**:** Impedance-controlled [24] trajectory execution on a 7-DoF torque-controlled robot [25], with force/torque sensing via a momentum-based external observer for physical interaction detection [26].

B. IROSA: LLM Tool Architecture

The IROSA (Interactive Robot Skill Adaptation) architecture defines robot skill adaptations as typed tool functions that the LLM selects and parameterizes. Each tool specifies: a function name, a natural language description, typed parameters with JSON schema, and validated bounds enforced before execution. Tab. S-I lists all tools.

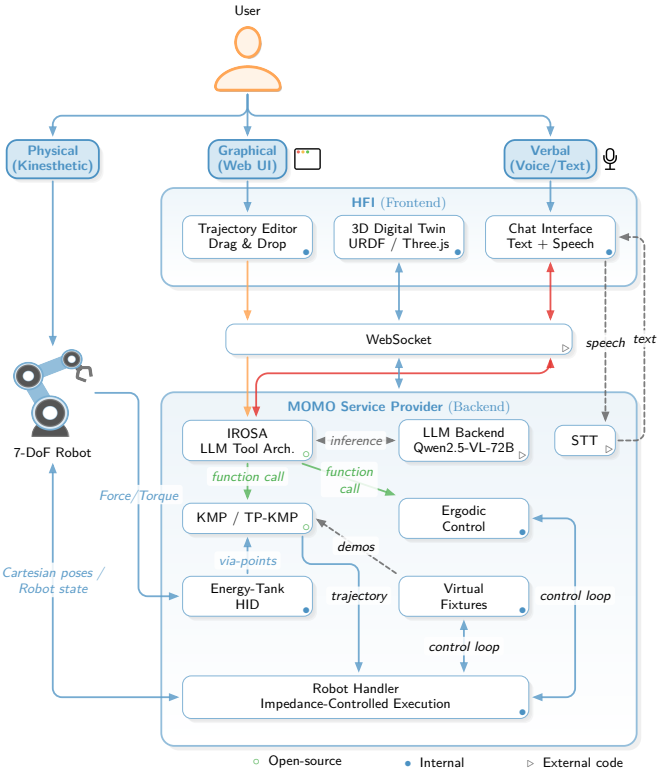


Fig. S1. Software architecture and data flow. The three user modalities (physical, verbal, graphical) connect to the system through distinct paths. The HFI Frontend communicates with the MOMO backend via a WebSocket bridge: the orange path carries trajectory adaptations from the Trajectory Editor, the red path carries bidirectional chat messages to and from the IROSA. The Chat Interface accepts text input directly or forwards speech audio to the STT module for transcription. The IROSA uses an external LLM backend for inference and dispatches function calls to the skill modules (KMP/TP-KMP, Ergodic Control). The Energy-Tank Human Intention Detection translates Force/Torque readings from the robot into via-points for KMP, while Virtual Fixtures provide demonstration data to KMP and maintain a closed control loop with the Robot Handler. KMP generates trajectories for the Robot Handler, which performs impedance-controlled execution. The Ergodic Control module likewise maintains a closed control loop with the Robot Handler. The Robot Handler sends Cartesian poses to the robot, which in turn provides its state back to the Robot Handler. Component ownership is indicated by symbols: ○ open-source, ● internal, ▷ external code.

The LLM receives the tool definitions via the standard function calling API. Given a user command, it returns a structured `tool_call` containing the selected function name and extracted arguments. The MOMO module validates all arguments against the defined bounds before executing the tool on the KMP or ergodic model. This separation ensures that the

[1] https://gitlab.com/links_and_nodes/links_and_nodes

LLM never directly controls the robot—it can only select from pre-validated operations.

New tools can be added by defining a Python function with parameter validation and registering it with a JSON schema description. No LLM retraining is required.

The bounds in Tab. S-I are fixed constants that do not vary per task, and they were determined empirically. For the ergodic tools we started from the parameter values used in the demonstrations (velocity 6, contact force 15 N, stiffness 1000 N/m) and bracketed each with a conservative margin around the range that produced sensible behavior during development; the KMP repulsion radius and speed modulation range were fixed the same way. The bounds are therefore tied to this cell, tooling, and skill set; none of them follow from first principles. Transferring the framework to another setup means re-fitting them once per tool, a configuration change that affects neither the LLM nor its prompts.

TABLE S-I
LLM TOOL SPECIFICATIONS WITH VALIDATED PARAMETER BOUNDS.

Tool	Parameters	Range	Skill
AddViaPoints	time $\in [0,1]$, pos. $[x,y,z]$	–	KMP
AddRepulsion	pos. $[x,y,z]$, radius	$r \leq 1.0$ m	KMP
SlowDown	percentage, t_{start}, t_{end}	$[1,100]\%$	KMP
SpeedUp	percentage, t_{start}, t_{end}	$[1,100]\%$	KMP
SetVelocity	velocity	$[3,16]$	Ergodic
SetForce	force (N)	$[5,30]$	Ergodic
SetStiffness	stiffness (N/m)	$[500,2000]$	Ergodic
SetExecState	"pause" / "resume"	–	Ergodic

C. Reliability, Failure Modes, and Latency

Tool-selection reliability and response latency were quantified separately in [5] on the same robot and task; we summarize the outcome here and refer to that paper for the protocol and baseline comparisons. Depending on the adaptation type, commands were interpreted as intended in 80–100% of cases, with all runs completing the task. The dominant failure mode was over-application: the LLM occasionally added an unrequested speed modulation alongside a correct via-point insertion, while still selecting the right tool.

For the system demonstrated at the trade fair (Tab. S-II), the average end-to-end adaptation latency was 15.4 s, roughly half that of a code-generating baseline on the same local backend. Since the demonstration we have reimplemented the pipeline in C++[2], which reduces the measured end-to-end latency to 2–3 s for natural language commands, including LLM inference, and to about 1 s for physical and graphical adaptations. The bulk of the original latency therefore lay in the surrounding software path rather than in LLM decoding.

Failures are contained by construction, not by model accuracy. Arguments outside the bounds of Tab. S-I are rejected before the tool executes, and every via-point is inspectable and reversible in the graphical interface (Fig. S11), irrespective of the modality that created it.

D. Web-Based GUI Implementation

The graphical interface is built with Vue.js 3 for the component framework, Pinia for reactive state management, and Three.js for 3D visualization. The robot digital twin is rendered from URDF models using the `urdf-loader` library. Communication with the MOMO backend occurs via WebSocket through the LN bridge.

a) Drag-and-Drop Via-Point Editing.: When the user enables edit mode, the interface activates 3D picking on the rendered KMP trajectory. Clicking on a trajectory point and dragging it to a new position triggers the following pipeline:

1) The frontend sends an "add via-point" service call with the point index and new (x, y, z) position via the LN WebSocket bridge,
2) MOMO inserts the via-point into the KMP model with $\gamma = 10^{-8}$,
3) the frontend requests the updated model mean and covariance via a second service call,
4) the trajectory visualization refreshes, showing original (blue) and adapted (yellow) trajectories.

Right-clicking a via-point opens a context menu for adapting (dragging) or deleting it, with the trajectory updating in real time after each modification.

b) LLM Chat Integration.: The ChatBox component sends user text to the MOMO backend via the `set_llm_input_query` service. The frontend subscribes to a notification topic; when the LLM response is ready, it retrieves the answer via `get_llm_answer` and displays it with animated word-by-word rendering. Speech input is handled via a separate speech-to-text service.

c) Real-Time Digital Twin.: During trajectory execution, the frontend subscribes to the execution status topic, receiving the current via-point index and execution state. The robot digital twin updates its joint configuration accordingly, with a progress bar indicating execution progress.

E. Integration: Multi-Modal Via-Point Convergence

A key design insight is that all three modalities converge on the same KMP adaptation mechanism—via-point insertion—through different input paths:

- **Physical interaction**: The energy-tank-based Human Intention Detection (HID) detects intentional corrections per DoF. When the intention index exceeds $h_{\text{th}} = 0.9$, the corrected pose is automatically inserted as a via-point at the corresponding time step. The KMP's uncertainty modulates how long the user must interact before a correction is registered.
- **Natural language**: The LLM selects the `AddViaPoints` tool and extracts positions from spatial references in the command (e.g., "move left at the start" $\rightarrow$ via-point at $t = 0.1$ with shifted y). For obstacle avoidance, the `AddRepulsion` tool computes repulsion via-points from signed distance fields.
- **Graphical interface**: The user directly places via-points via drag-and-drop on the 3D trajectory visualization.

[2]The C++ implementation will be released as open-source software at https://github.com/DLR-RM/pglfd.

After any via-point insertion (regardless of modality), the KMP recomputes its prediction, and the updated trajectory is immediately available to all other modalities. This enables seamless multi-modal workflows, e.g., adding obstacle avoidance via voice, then fine-tuning the result via drag-and-drop.

F. Human Intention Detection (HID)

The energy-tank-based intention detection runs as a C++ real-time service; for algorithmic details, see [4]. The module requires a torque-controlled robot; in our setup, an external observer estimates contact forces along the entire robot body [26] rather than relying solely on a wrist-mounted force/torque sensor, enabling more intuitive physical corrections. Sensor signals are filtered at 400 Hz with dead zones of 7.0 N (force) and 7.0 Nm (torque) for the external observer. The energy tanks are configured per DoF with separate parameters for translational axes (tank size 0.4, energy trigger 0.38, dissipation rate 0.04) and rotational axes (tank size 1.0, energy trigger 0.7, dissipation rate 0.2). When HID is enabled, the robot's collision stop is disabled to permit physical contact.

When the intention index in a specific DoF exceeds the threshold h_{th}, the detected via-point for that DoF is constructed as follows: for each translational axis where interaction was detected, the current end-effector position (transformed from flange to tool frame) replaces the KMP prediction; axes without detected interaction retain the original KMP-predicted position. If any rotational interaction is detected, the full measured orientation overrides the KMP prediction. The resulting 7-DoF pose (position + quaternion) is paired with the corresponding KMP time index and inserted as a via-point with precision $\gamma = 10^{-8}$. After trajectory execution, the Cartesian impedance stiffness is reset to $K_f = 1000\,\text{N/m}$, $K_t = 100\,\text{Nm/rad}$.

G. Virtual Fixtures for Skill Acquisition

The Virtual Fixtures component [7], [8], [21] runs as a Simulink real-time controller parametrized by a Python-based trajectory fixture controller via LN; for algorithmic details, see the cited works. Fig. S4 shows the ellipsoidal fixture visualization used during kinesthetic teaching.

In the experiments, we use GMM-based trajectory fixtures as well as KMP-based velocity fixtures, both learned from demonstrations. Both pipelines share the same preprocessing: pose recordings (position + unit quaternion, stored per time step) are loaded and resampled based on Cartesian distance (threshold 0.001 m).

For trajectory fixtures, the resampled demonstrations are then aligned via dynamic time warping, and a GMM is fitted to the time-augmented data on a product manifold ($\mathbb{R}^1 \times \mathbb{R}^3 \times \mathcal{S}^3$) with the Riemannian EM algorithm [27]. Through GMR, a mean trajectory with associated covariances is extracted, which are inverted to precisions and sent to the real-time controller. The variable impedance formulation [8] derives coupled stiffness matrices from these precisions via precision decomposition, providing position-dependent haptic guidance: high-precision regions are stiff to guide the operator, while low-precision regions are compliant to allow deviation. Passivity of the closed loop is ensured by [21]. We use two trajectory fixtures, one for guiding the ring grasping phase and one for the placement on the measurement station.

For velocity fixtures, positional and rotational guidance are learned independently as detailed in [8]. Velocities are computed from the resampled poses and encoded in GMMs on $\mathbb{R}^3 \times \mathbb{R}^3$ and $\mathbb{R}^3 \times \mathcal{S}^3 \times \mathbb{R}^3$, which are then used to initialize KMPs with an RBF kernel of length scale 0.03. These velocity fixtures coarsely guide the operator towards the placement location.

Fixture configurations are stored as YAML files containing the GMM parameters (manifold type, means, covariances, priors), kernel configuration, and fixture origin. The controller supports up to 10 simultaneously active trajectory and velocity fixtures with independent loading, modification, and origin updates.

H. Ergodic Surface Finishing

The ergodic surface finishing component uses the Spectral Multiscale Coverage (SMC) method for generating coverage trajectories on freeform surfaces, with the ergodic cost evaluated using $K{=}15$ Fourier frequency components per dimension; for algorithmic details (surface parametrization, compliant-disk imprint model, coverage learning from demonstrations), see [9].

An impedance controller with force overlay maintains the desired contact normal force. The LLM tools defined in Tab. S-I set runtime properties on the ergodic controller via a *links and nodes* service call: `SetVelocity` sets the velocity limit (range $[3, 16]$, default 6), `SetForce` sets the contact normal force setpoint (range $[5, 30]$ N, default 15), `SetStiffness` sets the translational surface-tangential stiffness components (range $[500, 2000]$ N/m, default 1000, while the normal component remains fixed at 800 N/m), and `SetExecState` switches the state machine between idle and free-floating mode to pause or resume execution. Since trajectories are generated online, the operator can physically interact during execution: end-effector stiffness is varied with the distance to the surface, allowing the operator to lift the tool and reposition the robot to a new location on the workpiece, after which execution resumes while preserving the full coverage history.

I. Configuration Parameters

Tab. S-II lists the key configuration parameters.

TABLE S-II
KEY CONFIGURATION PARAMETERS OF THE FRAMEWORK COMPONENTS.

Parameter	**Value**	**Component**
GMM components	12	KMP
Sample points N	500	KMP
Kernel (Matérn $\nu = 5/2$), l	0.1	KMP
Regularization λ_1, λ_2	0.1, 1	KMP
Via-point precision γ	10^{-8}	KMP
Intention threshold h_{th}	0.9	HID
Tank size $E_{\max}$ (trans. / rot.)	0.4 / 1.0	HID
Energy trigger E^* (trans. / rot.)	0.38 / 0.7	HID
Dissipation rate P_d (trans. / rot.)	0.04 / 0.2	HID
Reset stiffness K_f / K_t	1000 / 100	HID
Resampling threshold	0.001 m	VF
RBF kernel length scale l_v	0.03	VF
Max. active fixtures	10	VF
LLM model	Qwen2.5-VL-72B-Instruct	LLM
LLM deployment	Local (GPU)	LLM
Avg. response time	15.4 s	LLM

J. Open-Source Resources and Reproducibility

TABLE S-III
OPEN-SOURCE RESOURCES ACCOMPANYING/USED IN THIS PAPER.

Resource	**License**	**Content**
IROSA[3]	MIT	LLM tool arch. + examples
KMP/TP-KMP[4]	MIT	Motion primitives
links and nodes (LN)[1]	GPLv3	Middleware
PGLfD[2]	MIT	C++ reimpl. (planned)

Tab. S-III lists all open-source resources used with this work.

The **IROSA**[3] provides the tool-based LLM architecture including tool definitions, parameter validation, KMP model, and integration with OpenAI-compatible function calling APIs. It includes three example scripts demonstrating speed adaptation, via-point insertion, and obstacle avoidance, each runnable without hardware dependencies. The open-source release replaces the proprietary LLM backend with a standard interface for reproducibility.

KMP/TP-KMP[4] provides the motion primitive implementation with interactive incremental learning, four reproducible experiments, and demonstration data.

Links and nodes [23] is the open-source middleware connecting all components, providing process management, pub/sub topics, and request/response services.

[3]https://github.com/DLR-RM/IROSA
[4]https://github.com/DLR-RM/interactive-incremental-learning

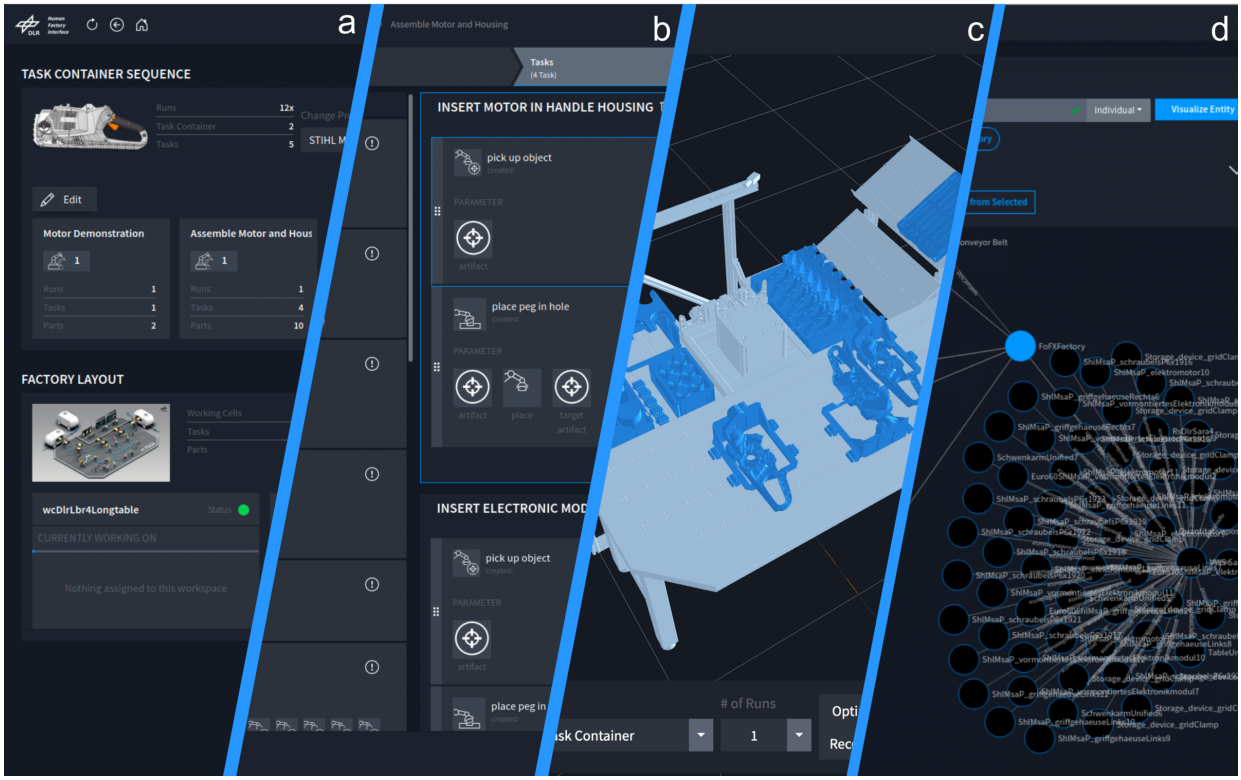


Fig. S2. Overview of the Human Factory Interface consisting of four views: (a) Factory Overview, (b) Task Container Design View, (c) Workcell Viewer, and (d) Ontology Viewer.

S-II. Human Factory Interface

The Human Factory Interface (HFI) is a web-based graphical interface (implementation details in Sec. S-I) that enables operators to monitor, configure, and control robotic workcells, i.e., workspaces comprising one or more robots together with the tools, fixtures, and objects they operate on. In our setup, the HFI manages a *variable workcell* [28], in which fixtures and assets can be reconfigured for different tasks and the layout is registered in the knowledge graph so that the digital twin always reflects the current physical arrangement. The system is backed by a knowledge graph [29] that stores the relationships between workcells, robots, physical assets, and a library of reusable robot skills—parameterizable building blocks such as *pick*, *place*, or *insert* that encapsulate individual robot actions [30]. Non-expert users can compose robot programs by sequencing and parameterizing these skills, either through a graphical interface or by recording demonstrations [30], [31]. As shown in Fig. 1 in the main paper, the HFI observes and receives notifications from the knowledge graph, and communicates directly with the robot for execution and monitoring.

The HFI is organized into four views, as shown in Fig. S2:

a) Factory Overview.: The factory overview (Fig. S2a) displays all registered workcells and their operational states. Operators can monitor which skill is currently executing, pause or stop execution, and investigate warnings or failures.

b) Task Container Design View.: The task container design view (Fig. S2b) allows users to compose task sequences by selecting skills from the library, setting their parameters (e.g., target positions, velocities), and arranging them into executable programs.

c) Workcell Viewer.: The workcell viewer (Fig. S2c) is the primary view used in this work. It provides a 3D visualization of the robotic workcell, rendering the robot and all registered assets as a digital twin. This view supports direct manipulation and has been extended with the skill visualization and interaction capabilities described in the following section.

d) Ontology Viewer.: The ontology viewer (Fig. S2d) visualizes the knowledge graph as an interactive node-edge diagram, showing available resources, active configurations, and the relationships between system components [29].

S-III. MOMO Interface Screenshots

The Motion Modulation Panel, i.e., the graphical interface of the MOMO module (Sec. S-I), extends the HFI Workcell Viewer (Sec. S-II) with skill visualization and interactive adaptation. The following screenshots (Fig. S3–Fig. S13) illustrate its features.

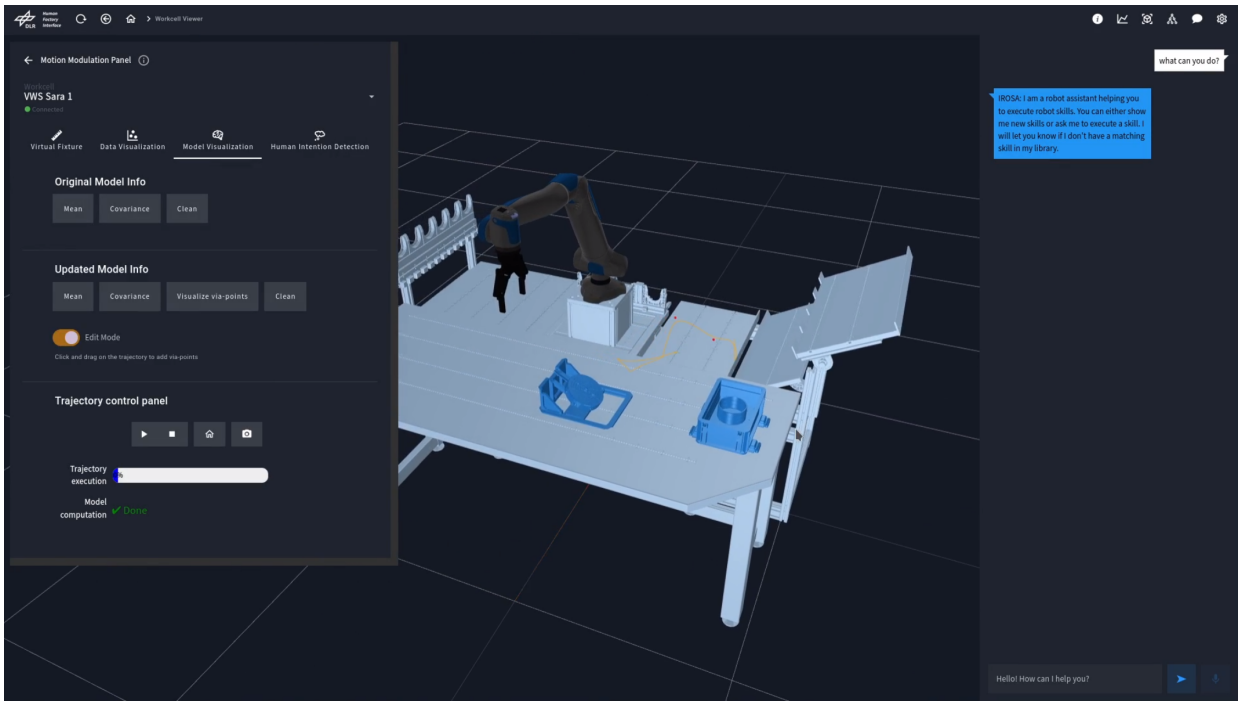


Fig. S3. Overview of the MOMO web-based interface. The IROSA is integrated as a chat window (right), enabling natural language skill adaptation through text and speech. The left panel shows the Model Visualization tab with controls for the original and updated model, edit mode, and trajectory execution. The 3D viewer displays the robot cell with the digital twin.

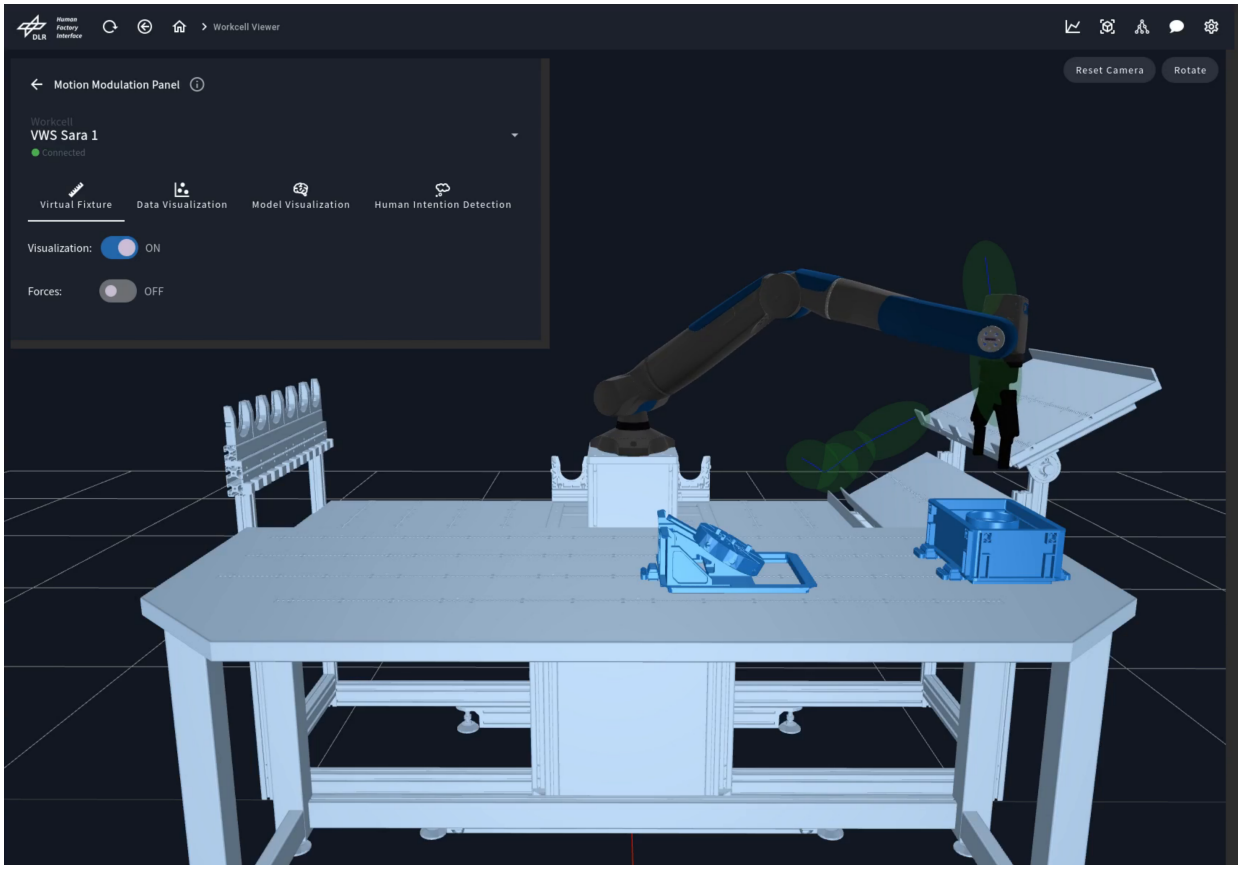


Fig. S4. Default Fixtures tab showing demonstration recording with Virtual Fixtures. The green, semi-transparent ellipsoids overlaid on the trajectory represent the probabilistic Virtual Fixtures that provide haptic guidance to the operator during kinesthetic teaching.

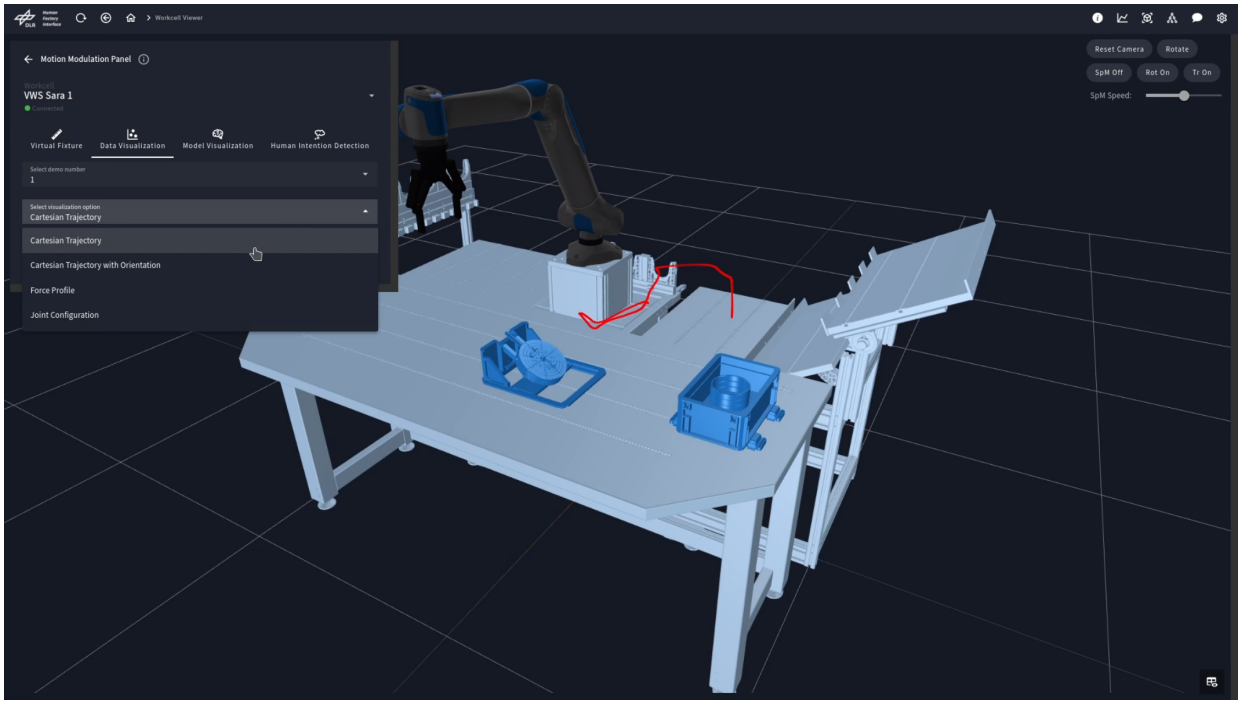


Fig. S5. Data Visualization tab of the web-based interface. A recorded demonstration is selected and its Cartesian trajectory is visualized in red (flange position) projected into the 3D robot cell. The dropdown menu provides additional visualization options: Cartesian Trajectory with Orientation, Force Profile, and Joint Configuration.

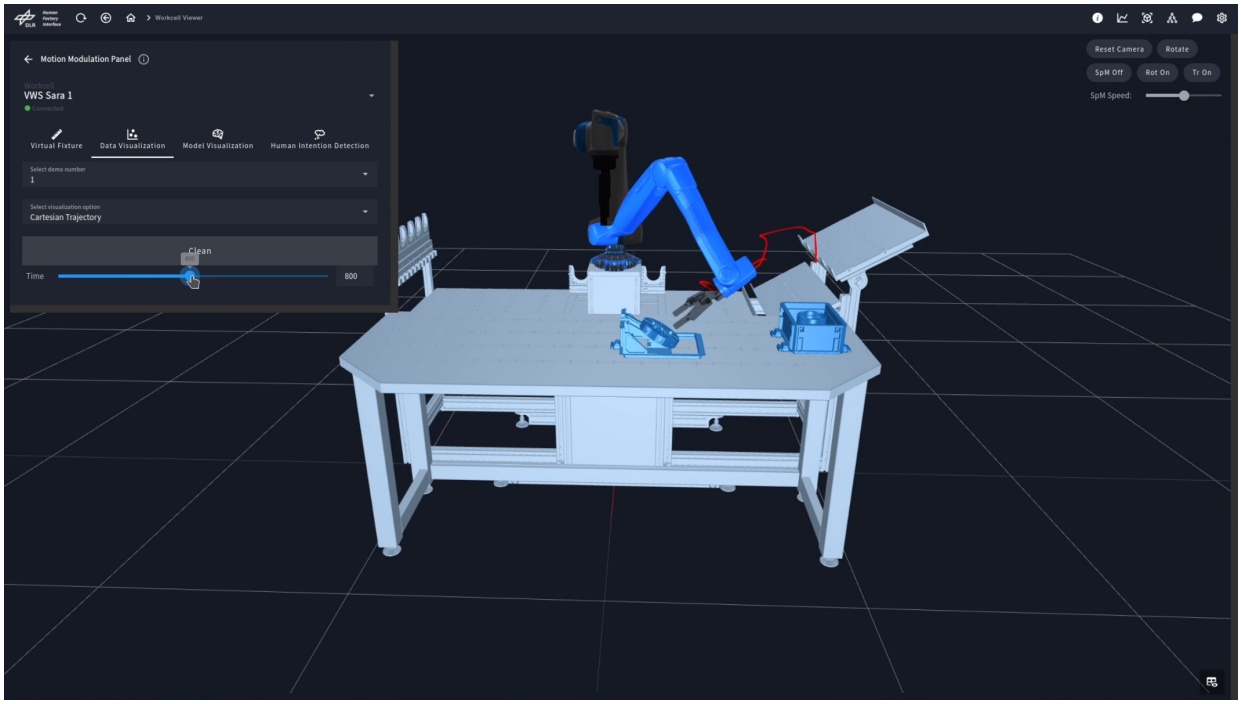


Fig. S6. Data Visualization tab with the Joint Configuration option selected. A blue robot hologram shows the robot pose at the currently selected time step of the demonstration.

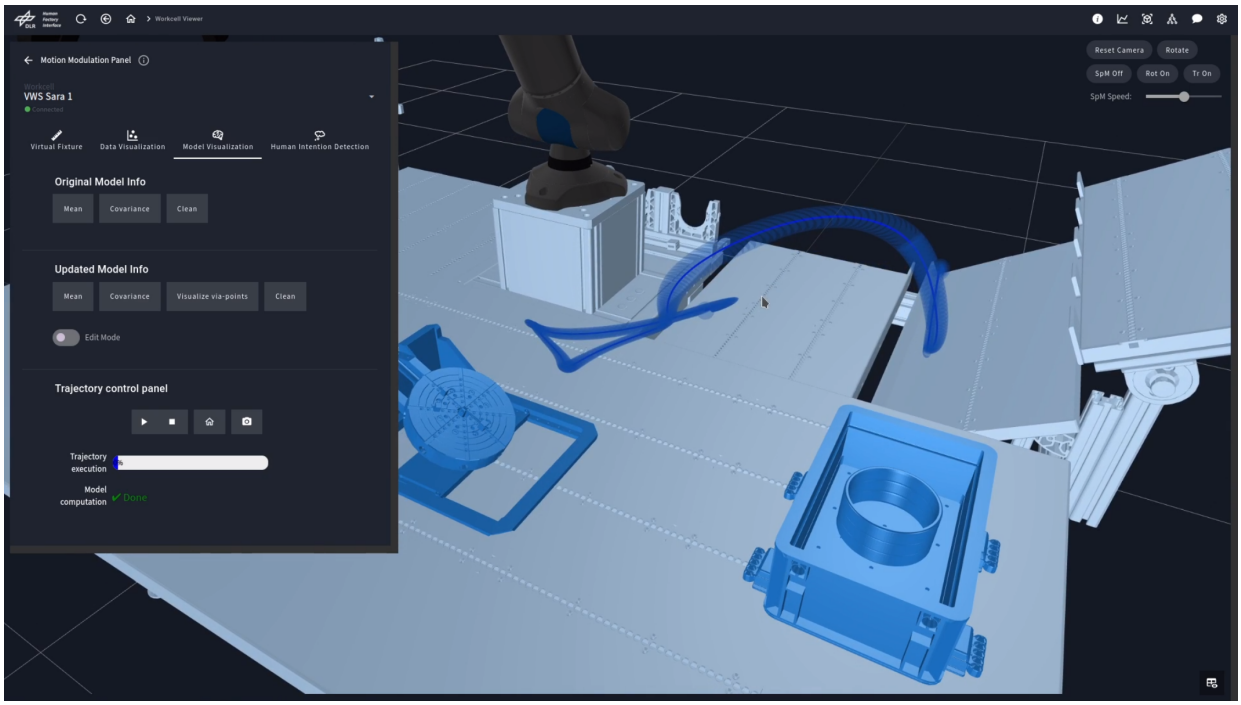


Fig. S7. Model Visualization tab showing the original KMP mean and covariance in blue projected into the robot cell. The panel provides controls for visualizing both the original and updated model (e.g., after via-point insertion), an edit mode toggle for interactive trajectory editing, and a trajectory control panel for starting execution and monitoring progress.

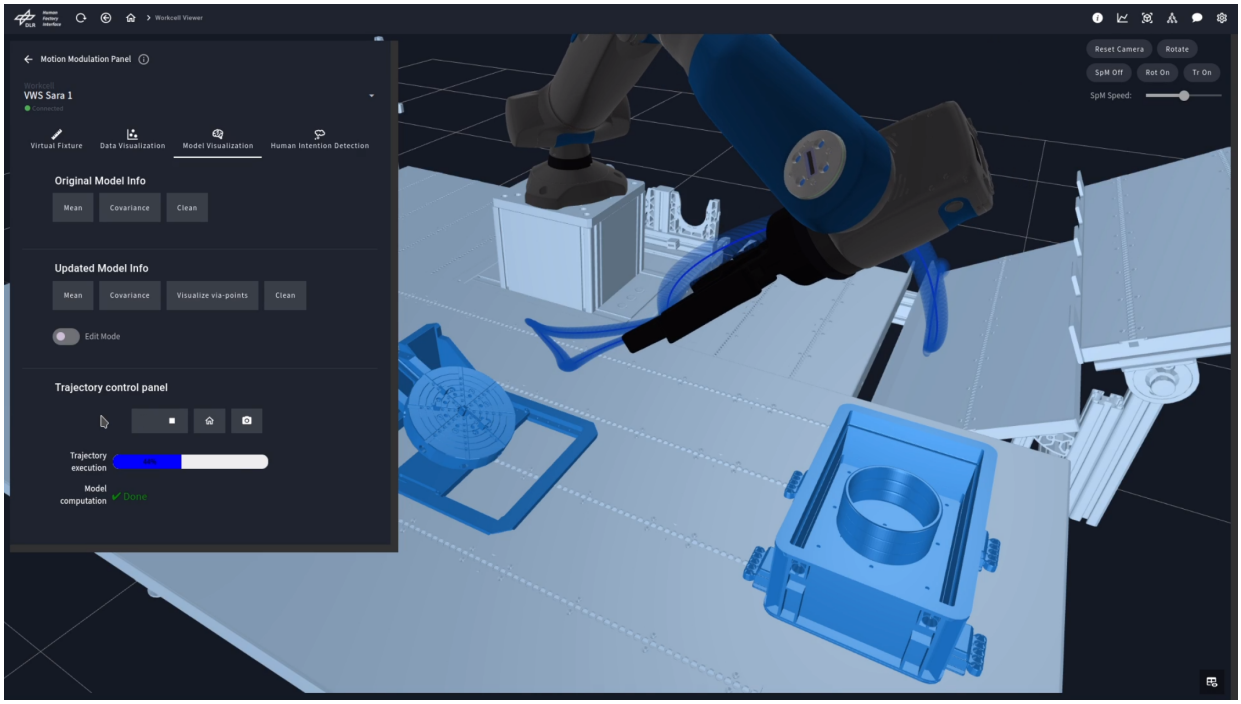


Fig. S8. Model Visualization tab during trajectory execution. The trajectory control panel shows the execution progress, and the digital twin of the robot moves in sync with the real robot.

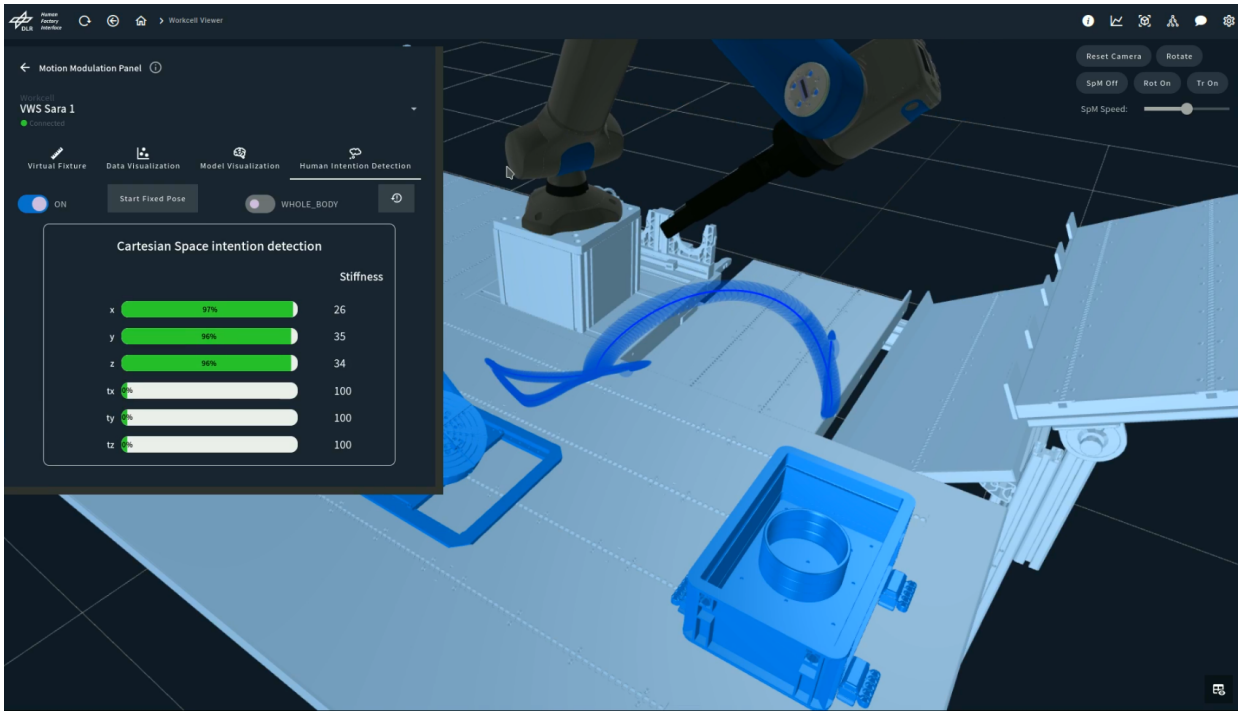


Fig. S9. *Human Intention Detection* tab. When enabled, the interface displays for each Cartesian DoF (x, y, z, θ_x, θ_y, θ_z) the intention index, proportional to the energy tank value, along with the corresponding robot stiffness, which is inversely proportional to the intention index, allowing the operator to monitor physical human-robot interaction in real time.

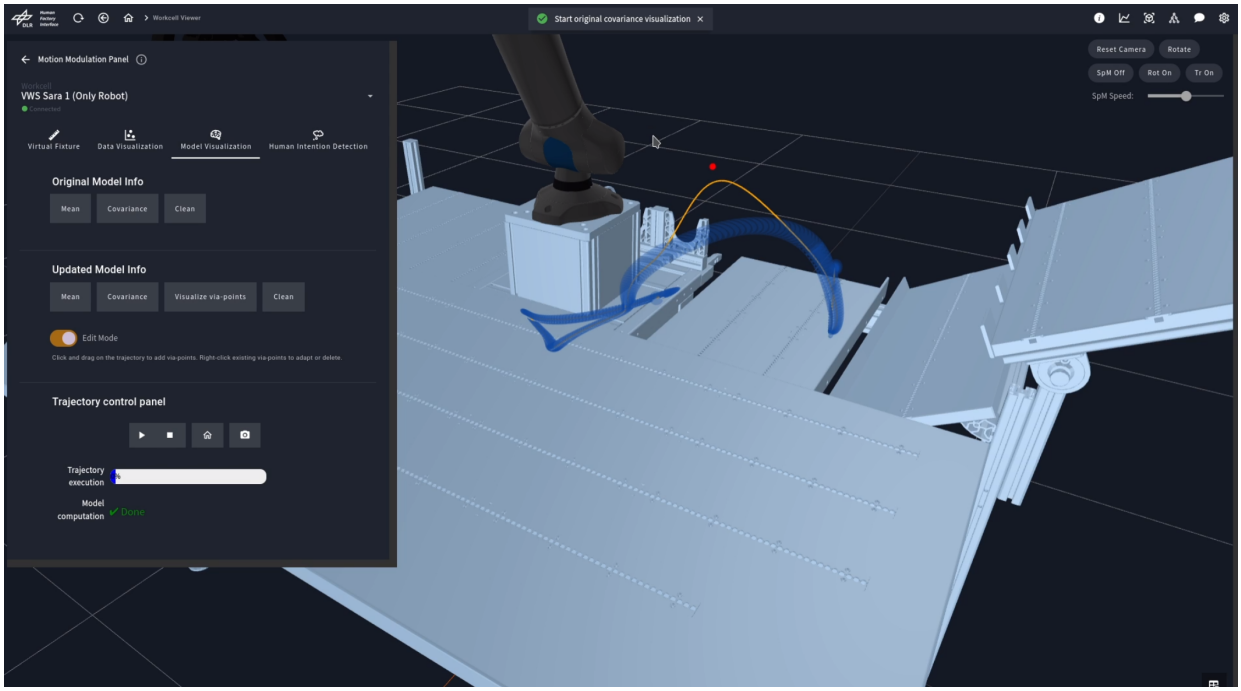


Fig. S10. Model Visualization showing a via-point adaptation. The original KMP mean and covariance are displayed in blue, the adapted mean trajectory in yellow, and the set via-point in red.

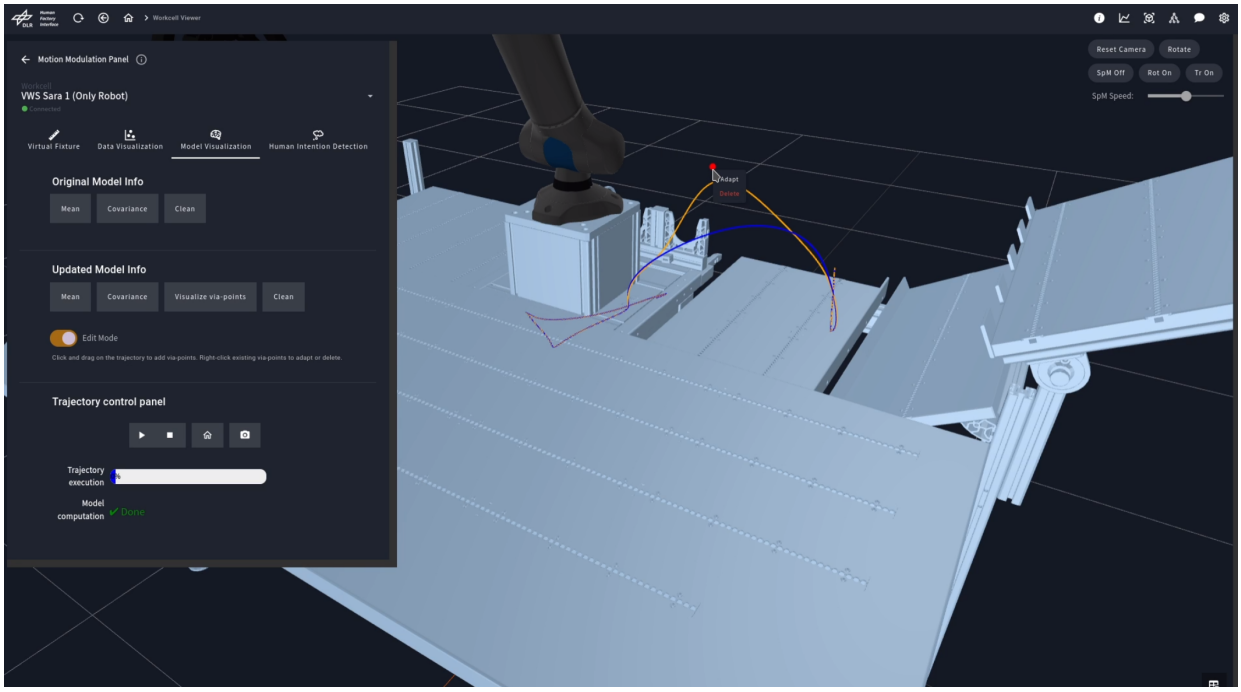


Fig. S11. Context menu for via-point manipulation. Right-clicking on a via-point opens a menu with options to adapt (move with the mouse) or delete the via-point, with the trajectory updating accordingly in real time.

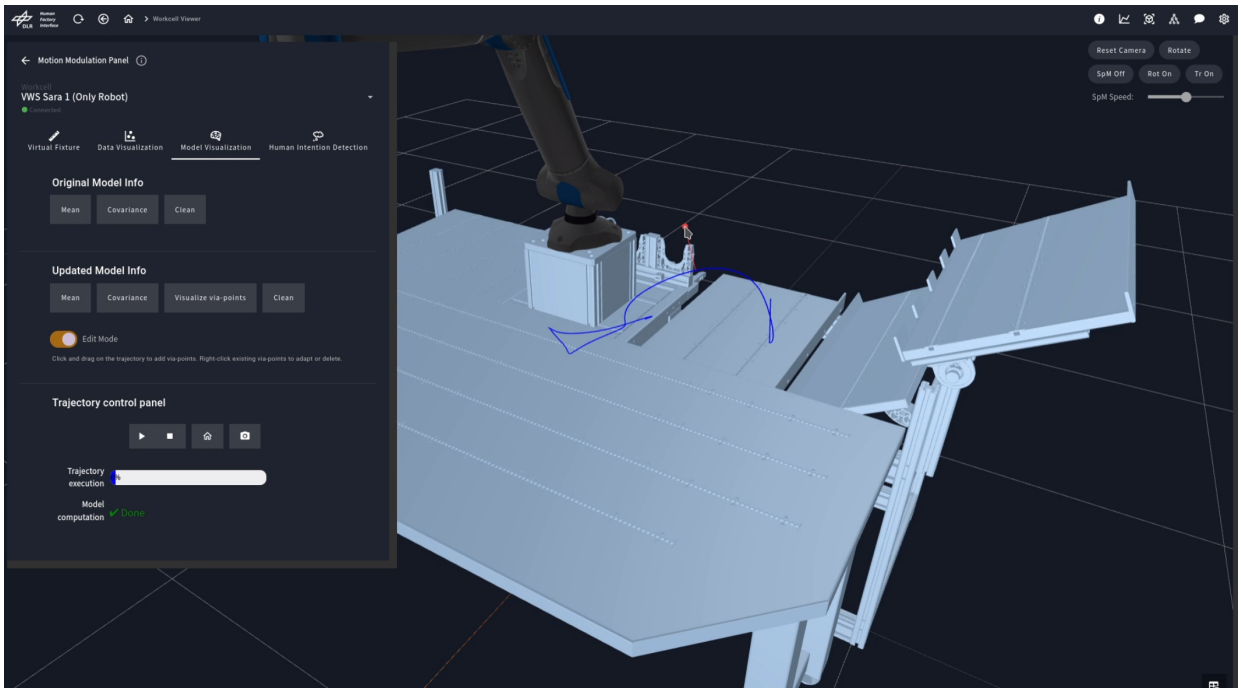


Fig. S12. Drag-and-drop via-point creation. The user clicks on a trajectory point and drags it to a new position, creating a via-point directly through the graphical interface.

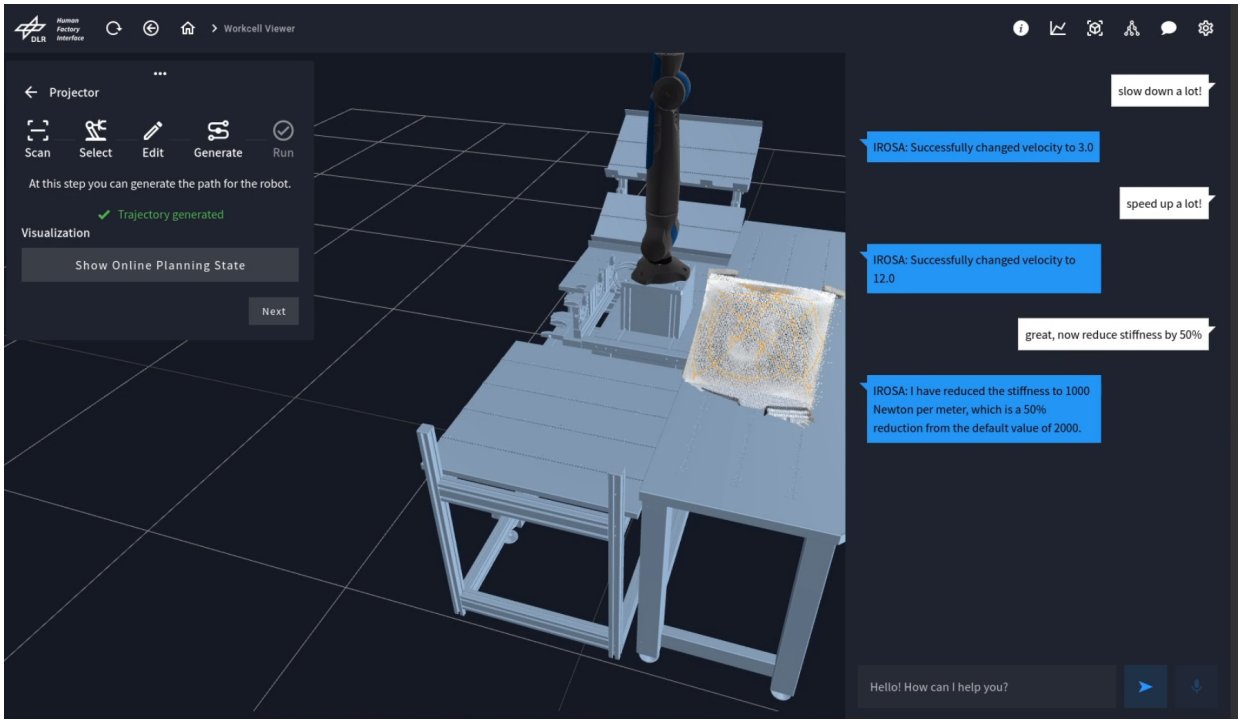


Fig. S13. Ergodic surface finishing skill with integrated IROSA chat interface. This demonstrates that the framework, including the LLM-based natural language adaptation, generalizes beyond KMP-based approaches to other skill representations such as ergodic control. The chat shows example commands for velocity and stiffness adaptation.