WireCraft: A Simulation Benchmark for Industrial DLO Manipulation

Chongyu Zhu[1†] **Ramy ElMallah**[1*] **Hyegang Kim**[1] **Zachary Tang**[1] **Jiachen Rao**[1]
Artem Arutyunov[1] **Seungyeon Ha**[2] **Chi-Guhn Lee**[1]

[1]Department of Mechanical and Industrial Engineering, University of Toronto, Canada
[2]CREFLE Inc., Seongnam, Republic of Korea

Abstract: Deformable Linear Objects (DLOs), such as wires and cables, are central to industrial assembly. Unlike rigid objects, whose state is captured by a 6-DoF pose, DLOs have an infinite-dimensional configuration space and deform continuously under contact with grippers, fixtures, and the workspace, making them a demanding benchmark for general dexterous manipulation. Despite their importance, policy development and comparison remain difficult: existing benchmarks are often tied to specific hardware setups, lack modular and customizable task assets, or study generic deformable-object tasks without the fixtures relevant to real-world industrial wire manipulation. Few benchmarks align simulation, real-world data, and shared evaluation protocols. To bridge this gap, we introduce **WireCraft**, a simulation benchmark for industrial DLO manipulation with configurable difficulty and assets, spanning three task families: connector insertion, clip routing, and channel seating. It supports two complementary DLO physics models, articulated and deformable, and the released trajectories come from both simulation and a physical UR5. We benchmark reinforcement learning (RL), imitation learning (IL), and vision-language-action (VLA) policies under shared metrics. Privileged state-based RL solves a representative setting in each task family with over 82% success, confirming the tasks are well-posed. Image-based policies remain less successful across the evaluated tasks, with ACT and DiT VLA falling below privileged State PPO. On a physical UR5, supplementing real demonstrations with simulated data improves success by 15–65 percentage points in the best-performing mixture for each task. These results indicate that industrial DLO manipulation, though tractable under privileged state, remains an open challenge for current vision-based learning. Our project is open-sourced at https://github.com/WireManip/WireCraft.

1 Introduction

Manipulating Deformable Linear Objects (DLOs) [1], particularly wires, is important for industrial wire-harness assembly [2] and cable management [3, 4]. These settings require the robot to do more than grasp and move a flexible object: it must interact with structured fixtures such as clips, connectors, sockets, and channels [4, 5] while controlling the DLO's deformation [6]. Unlike rigid objects, whose state can be captured by their 3D position and orientation, DLOs possess an infinite-dimensional configuration space and deform continuously [7, 8, 9]. Forces applied at the gripper propagate along the body, causing simultaneous deformation and motion across the DLO [10]. DLOs further exhibit complex contact dynamics, self-occlusion, and deformation uncertainty arising from variation in material stiffness and damping [10, 11]. These properties make policy learning difficult. Compounding the challenge, existing efforts often tie simulation, fixtures, and sensing to

*Equal contribution.
†Corresponding author: cyzhu@mie.utoronto.ca

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

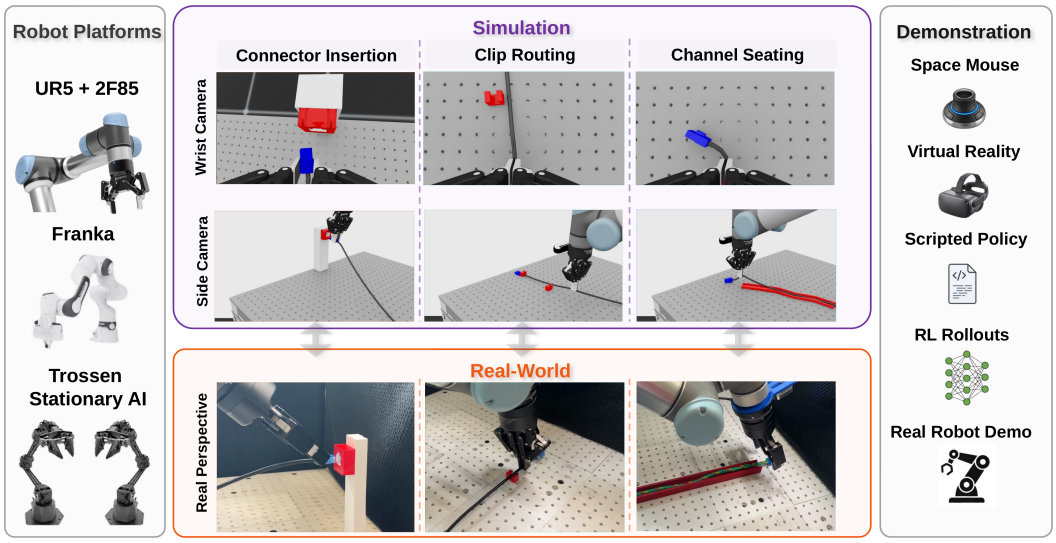


Figure 1: **WireCraft** simulates industrial DLO manipulation across three task families: connector insertion, clip routing, and channel seating with trajectories from multiple sources. Tasks run on the UR5, Franka, and Trossen Stationary AI in simulation, with real-world validation on a UR5.

specific hardware setups, leaving industrial DLO tasks without reusable simulation benchmarks or shared data interfaces.

General robot learning benchmarks such as RLBench [12] and ManiSkill3 [13] are not centered on DLO manipulation and provide limited DLO-specific data. Deformable-object benchmarks such as SoftGym [14], DeformableRavens [15], and DaXBench [16] include DLOs alongside other deformable categories, while DLO-dedicated environments such as ReForm [17] and DLO-Lab [18] target canonical DLO behaviors like shape control and coiling. However, both lack structured industrial fixtures such as connectors, clips, and channels central to assembly tasks. A parallel line of work studies these industrial tasks on physical testbeds [2, 3, 4, 19, 20, 21], but each effort is tied to specific robots, fixtures, cable geometries, and sensing setups, limiting reuse for new model training or sim-to-real research. Physical task boards such as NIST [22] and the ManipulationNet Cable Routing Benchmark [23] standardize real-world evaluation but offer no simulation environment. WireCraft combines configurable insertion, routing, and seating tasks with shared policy interfaces and trajectory data from simulation and physical robots.

To address this gap, we make the following contributions:

- We introduce WireCraft, a unified simulation benchmark for industrial DLO manipulation across connector insertion, clip routing, and channel seating, with configurable task components, standardized policy interfaces, and two complementary DLO physics models trading off speed and fidelity.
- We evaluate representative RL, IL, and VLA policies under a shared evaluation protocol, providing a consistent reference for future work on industrial DLO learning. Simulation baseline evaluation and physical UR5 validation cover all three task families.
- We open-source the benchmark with available trajectory data, data-generation tools, and 3D-printable task-board components. This enables researchers to reproduce the setups, collect demonstrations, and study sim-to-real transfer for industrial DLO manipulation.

2 Related Work

Robot learning and deformable-object benchmarks. General benchmarks have standardized rigid-object manipulation. RLBench [12] provides tabletop tasks with motion-planner-generated

demonstrations, while ManiSkill3 [13] extends this to contact-rich manipulation across 12 task categories. However, neither is centered on DLOs, and neither models fixture-constrained DLO assembly. A second line of work targets deformable-object manipulation: SoftGym [14] provides environments for ropes, cloth, and fluids; PlasticineLab [24] contributes a differentiable soft-body benchmark for deformation; DaXBench [16] unifies rope, cloth, and liquid manipulation under differentiable physics; and DeformableRavens [15] benchmarks goal-conditioned rearrangement of deformables. MoDeSuite [25] introduces mobile-manipulation deformable tasks in Isaac Sim. These benchmarks have driven progress on shape control and free-space rearrangement, but their DLO tasks are not industrially grounded.

DLO-specific simulators and learning environments. A more focused thread targets DLO modeling and control. ReForm [17] introduces an early sandbox of DLO shape-control tasks. Chen et al. [26] integrate Discrete Elastic Rods (DER) [27] into MuJoCo, targeting dynamic DLO simulation rather than benchmarked task suites. DexDLO [28] studies goal-conditioned dexterous manipulation of DLOs. DEFORM [29] and its extension DEFT [30] provide differentiable DER [27] models aimed at perception/control rather than assembly benchmarks. Govoni et al. [31] analyze DLO models under domain randomization across Isaac Sim and Gazebo, focusing on parameter sensitivity. The closest concurrent benchmark is DLO-Lab [18], a differentiable Genesis/Taichi-based simulator paired with DLO tasks and multiple policy-learning baselines. However, DLO-Lab emphasizes generic DLO shape control and material modeling rather than industrial tasks. WireCraft differs by organizing tasks around industrial task families, offering dual wire representations and policy-learning-ready datasets.

Industrial DLO manipulation and physical task boards. A growing body of applied work tackles the three task families WireCraft unifies. For *clip routing*, prior work learns motion primitives for guiding cables through clips [3, 4], proposes spatial representations with fixture-aware planning [5, 32], and combines VLM-based planning with RL for long-horizon tasks [19]. For *connector insertion*, methods generalize across materials and novel connectors via visual flexibility cues or meta-RL [9, 33], and optimize cable following, search, and mating [21, 34]. For *channel seating and constrained placement*, prior work has addressed clamp engagement [2], plugging/clipping/routing in constrained environments [20, 35], dual-arm planning under constraints [36], simulated cable laying [37], and DLO manipulation in benchmark task spaces [38]. Physical task boards provide standardized real-world evaluation: the NIST Assembly Task Boards [22, 39] and ManipulationNet [23] standardize protocols for deformable-part assembly, MOTORCYCLE 1.0 [40] automates bimanual cable routing on a reconfigurable board, and Intrinsic's AIC provides a 3D-printable cable insertion board with Gazebo simulation [41]. However, each system is tied to specific robots, fixtures, sensing, or protocols. WireCraft unifies all three task families under shared simulation, industrially grounded assets, and multi-source demonstrations, making it, to our knowledge, the first unified industrial DLO benchmark.

Learning algorithms for contact-rich and deformable manipulation. Recent progress in robot learning has produced several policy classes relevant to contact-rich manipulation, including on-policy and off-policy reinforcement learning [42, 43], action chunking [44], diffusion-based visuomotor policies [45], transformer-based diffusion backbones [46, 47], and VLA language-conditioned action prediction [48]. Subsequent work has also advanced deformable-object manipulation through geometry-aware RL and equivariant graph policies [49] and through model-based RL with differentiable multiphysics simulation gradients [50]. These methods motivate the need for benchmarks that expose contact-rich deformable manipulation under shared task definitions and evaluation metrics. To this end, we incorporate representative policy families across RL, IL, and VLAs as evaluation baselines rather than exhaustively optimized solvers, establishing WireCraft as a foundation for future specialized algorithms.

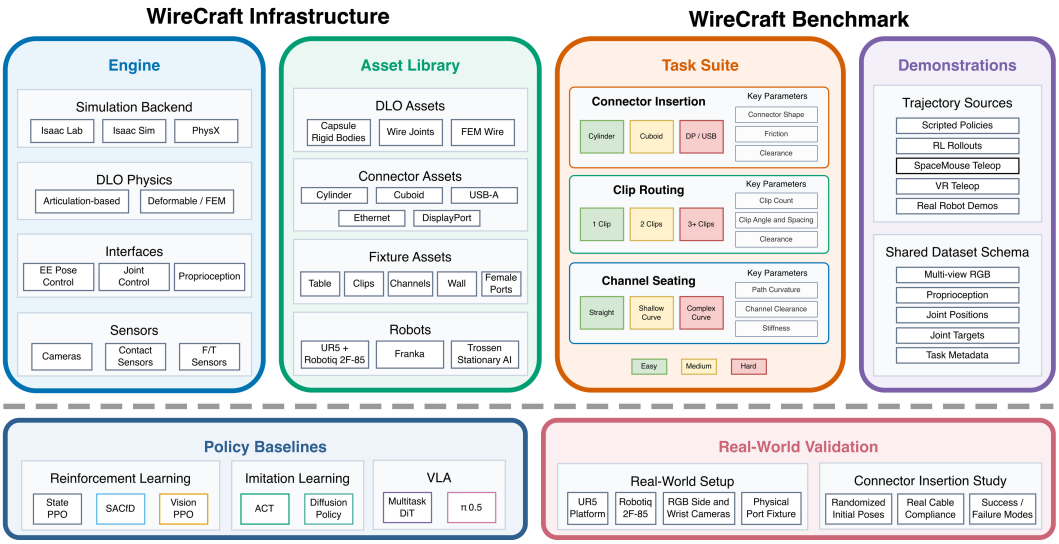


Figure 2: **WireCraft Architecture**, including the WireCraft simulation engine, asset library, task suite, demonstration sources, policy baselines, and real-world validation setup. WireCraft supports articulated and FEM-based deformable wire models, provides configurable industrial DLO manipulation tasks across connector insertion, clip routing, and channel seating, and pairs simulation data with real-world teleoperation trajectories under a shared dataset schema.

3 WireCraft Infrastructure

3.1 WireCraft Engine

Simulation Backend. WireCraft is built on Isaac Lab v2.2.1 [51], an open-source robot learning framework on top of Isaac Sim 4.5 [52]; full version and physics details are in Appendix B. Isaac Sim provides photo-realistic scenes and fast simulation, while Isaac Lab supports GPU-accelerated, multi-instance training for scalable policy learning and trajectory generation.

DLO Physics. WireCraft simulates its two wire representations through complementary physics mechanisms. The articulated wire uses rigid-body dynamics: each segment is a rigid body, and neighboring segments are coupled by 6-DoF joints (PhysX D6). This supports high-throughput trajectory generation due to efficient rigid-body contact but can become unstable under adjacent-segment collision or accumulated contact forces. The deformable wire uses Isaac Sim's deformable body API and PhysX FEM deformable body simulation [53, 54], improving local bending, compression, and contact deformation with higher fidelity, but at higher computational cost. We treat the two wire types as complementary benchmark representations. Detailed wire-model parameters and throughput measurements are reported in Appendix B, and a wire–connector physics visualization is shown in Figure 6. Details of DLO and scene interaction are provided in Appendix B.3.

Sensors. Beyond standard camera RGB data, WireCraft provides built-in interfaces to optional depth channels. To capture physical interactions, the platform integrates contact sensors for precise collision detection and force sensors to monitor grasping force magnitudes and insertion states. Furthermore, real-time robot proprioception, including joint positions and velocities, is natively supported. Observation inputs are selected per baseline, and users can add other sensor types supported by the NVIDIA Isaac ecosystem.

3.2 WireCraft Assets

DLO Assets. The articulated wires are parameterized by length, radius, colors, damping, and stiffness. The deformable wires are simulated via FEM, where Young's modulus, Poisson's ratio, and density govern its elastic response, alongside geometric parameters for length, radius, and mesh

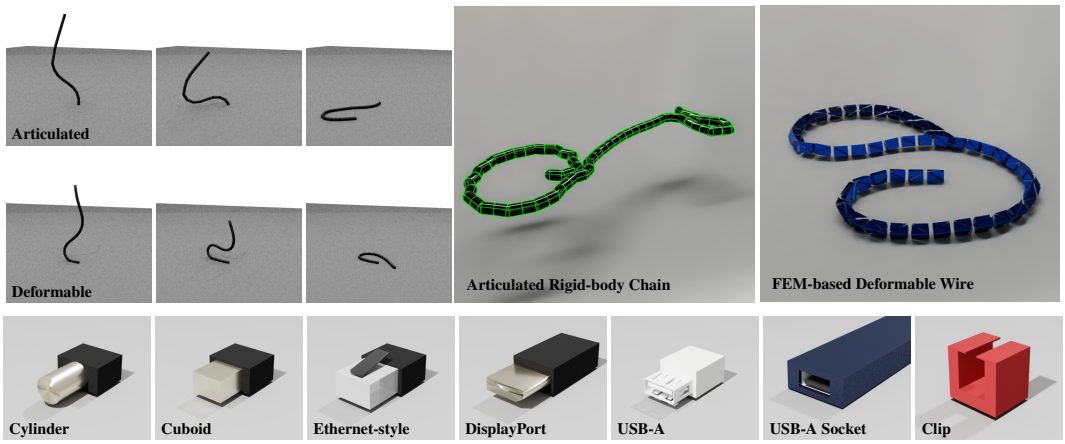


Figure 3: **WireCraft Assets**. *Top:* the two configurable wire representations, articulated and FEM-based deformable, shown as rendered DLOs (left) and their construction-level chain/mesh structure (right). *Bottom:* a subset of representative 3D-printable connectors and task fixtures.

resolution. Both share the same connector and task-component interfaces, so all tasks can be instantiated with either wire type.

Connector Assets. DLO-mounted connectors are 3D-printable and span two families: reference-based connectors modeled after real standards such as USB-A, and geometry-based connectors built from primitives such as cylinders and cuboids. They differ in rotational symmetry, allowing task difficulty to range from rotation-invariant to orientation-sensitive insertion.

Fixture Assets. Task-side fixtures, including clips, channels, and sockets, are likewise 3D-printable. They define the geometric constraints of each task family and remain configurable for different layouts and difficulty levels.

Robot Embodiments. Three simulated embodiments are supported: the UR5, Franka, and Trossen Stationary AI, each with its own gripper and control interface. Tasks can be instantiated across different arms under a unified observation framework, where robot-specific proprioception scales to each robot's degrees of freedom.

4 WireCraft Benchmark

4.1 WireCraft Tasks

We benchmark three industrial task families. **Connector Insertion** performs precision contact-rich insertion with a trailing deformable wire; **Clip Routing** performs fixture-constrained routing under contact and topological constraints; and **Channel Seating** performs continuous shape control along extended geometric constraints.

Task Curriculum. Task difficulty varies through connector geometry, fixture complexity, and path geometry. Connector insertion ranges from a rotation-invariant cylindrical connector to an asymmetric, orientation-sensitive DisplayPort-style connector. Clip routing difficulty scales with the number and placement of clips, while channel seating difficulty is controlled by channel curvature.

Randomization. All tasks share a randomized free-fall wire initialization, where the wire is spawned with varying position and orientation. Task-specific randomization varies the socket position for insertion, clip position and orientation for routing, and channel geometry for seating; detailed ranges are provided in Appendix C.4.

4.2 WireCraft Demonstrations

Data Collection. WireCraft supports four data collection methods that share a common dataset schema. Scripted policies provide scalable, successful demonstrations using privileged simulator state; RL rollouts add closed-loop interaction data and failure-adjacent states; human teleoperation in simulation captures corrective visual strategies for contact-rich manipulation; and real-world UR5 demonstrations provide physical validation. Our data-collection stack builds on an open-source ROS 2 + LeRobot teleoperation stack for WireCraft, with a shared episode recorder for simulation and real-robot data.

Shared Dataset Schema. All sources are serialized into a LeRobot v3.0 dataset format. For the UR5 robot, they include a 7-D state (six joint positions and gripper), a 7-D action (six joint commands and gripper), synchronized wrist and third-person RGB streams at the task control rate, and the language instruction for each task. This shared schema allows IL and VLA policies to be trained from any mixture of these sources without policy-specific task wrappers.

5 Experiments

To establish the benchmark's validity, we evaluate representative RL, IL, and VLA methods on our task suite under common task definitions, randomization, and success criteria. Experiments cover RL, IL, and VLA baselines on deformable-wire connector insertion, and privileged state PPO, ACT, and DiT VLA baselines on clip routing and channel seating.

For connector insertion, **Reach Success** ($\mathrm{SR}_{\mathrm{reach}}$) measures whether the gripper-held connector comes within 2 cm of the socket, and **Insert Success** ($\mathrm{SR}_{\mathrm{insert}}$) measures whether the tip seats in the socket. For clip routing, success requires the wire to be seated in the clips. For channel seating, success requires the wire to lie inside the channel over 80% of its length. Simulation results are reported as mean $\pm$ standard deviation across seeds.

5.1 Baselines on Connector Insertion

On Ethernet-style deformable-wire insertion with 3 mm clearance, we evaluate PPO [42], SAC from Demonstration (SACfD) [43], Vision PPO [42, 55, 56, 57], ACT [44], Diffusion Policy [45], a Diffusion Transformer VLA policy [58], and $\pi_{0.5}$ [48]. IL and VLA policies use the same dataset and evaluation, while state PPO and SACfD use privileged state information. IL/VLA results report the best observed checkpoint among three $\mathrm{SR}_{\mathrm{reach}}$-selected checkpoints (see Appendix F).

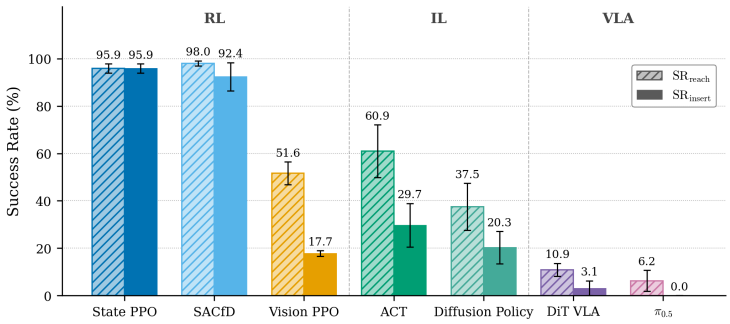


Figure 4: **Ethernet-style Connector Insertion performance.** $\mathrm{SR}_{\mathrm{reach}}$ and $\mathrm{SR}_{\mathrm{insert}}$ on the shared simulated insertion task show privileged RL solves the task reliably, while vision-based baselines exhibit a substantial gap between reaching the socket neighborhood and completing insertion.

Figure 4 shows three trends. First, privileged state-based RL solves the task reliably ($\mathrm{SR}_{\mathrm{insert}} \geq 92\%$ for both PPO and SACfD), establishing that the task is well-posed and that contact-rich alignment is tractable when socket and tail pose are directly observable. Second, every non-privileged baseline exhibits a sizeable reach–insert gap, showing that reaching the socket does not ensure successful

insertion. Vision PPO, however, collapses at the reach stage (51.6%), indicating the bottleneck for vision RL is both contact dynamics and RGB-only state estimation under occlusion and tight clearance. ACT and Diffusion Policy achieve higher contact-rich insertion phase success rates than the other image-input baselines, but both retain a clear gap between reaching the socket and completing insertion. Third, both VLA baselines underperform IL, with $\pi_{0.5}$ reaching only $6.2\%/0.0\%$; we attribute this primarily to a much shorter fine-tuning budget and a frozen vision tower (Appendix F), together with a pre-training distribution containing limited fine-grained contact-rich DLO interaction, rather than to a fundamental limitation of the architecture.

5.2 RL Baselines on Diverse Connector Insertion

We then extend the RL baselines to the full connector suite under the same 3 mm fixture clearance and task interfaces, evaluating privileged-state PPO, SACfD, and Vision PPO in Table 1.

Table 1: RL baseline success rates (%) on the diverse connector insertion tasks.

Connector	Metric	State PPO	SACfD	Vision PPO
Cylinder	$\mathrm{SR}_{\mathrm{reach}}$	95.00 ± 1.25	97.27 ± 1.06	$\mathbf{68.07 \pm 6.02}$
	$\mathrm{SR}_{\mathrm{insert}}$	93.75 ± 1.25	85.55 ± 8.68	$\mathbf{41.49 \pm 1.32}$
Cuboid	$\mathrm{SR}_{\mathrm{reach}}$	$\mathbf{97.08 \pm 1.44}$	96.70 ± 1.06	41.67 ± 4.73
	$\mathrm{SR}_{\mathrm{insert}}$	95.00 ± 3.75	$\mathbf{93.06 \pm 6.46}$	6.25 ± 3.31
Ethernet-style	$\mathrm{SR}_{\mathrm{reach}}$	95.86 ± 1.93	$\mathbf{98.05 \pm 1.05}$	51.63 ± 4.82
	$\mathrm{SR}_{\mathrm{insert}}$	$\mathbf{95.86 \pm 1.93}$	92.40 ± 5.91	17.74 ± 1.21
DisplayPort	$\mathrm{SR}_{\mathrm{reach}}$	95.00 ± 1.25	95.62 ± 10.58	34.01 ± 5.73
	$\mathrm{SR}_{\mathrm{insert}}$	92.92 ± 3.15	77.43 ± 6.17	3.32 ± 1.45

Three observations follow: First, both privileged baselines, PPO and SACfD, are largely insensitive to connector geometry, solving the alignment-sensitive DisplayPort about as well as the simpler connectors ($\mathrm{SR}_{\mathrm{insert}} \geq 92\%$ for PPO; $\geq 77\%$ for SACfD). Second, the ordering is not monotonic in geometric difficulty: the rotation-invariant cylinder, nominally the easiest connector, sits at or below the cuboid and Ethernet-style for both methods. We attribute this tentatively to the shared reward fixing a target socket orientation, which penalizes the cylinder's many equally valid yaw configurations and broadens the set of rewarded behaviors. Third, Vision PPO degrades sharply on every connector, and not monotonically; Ethernet-style is solved more often than the simpler cuboid. One likely explanation is the visual cues each shape provides for yaw estimation: the latch-like feature may give an orientation cue that the smooth cuboid lacks in RGB without depth.

Across connectors, the curriculum degrades vision-based performance while leaving privileged-state performance essentially unaffected, indicating that it stresses the visual observability of orientation rather than control. This supports its use as a difficulty axis (see Appendix E).

5.3 Clip Routing and Channel Seating

We further evaluate State PPO, ACT, and DiT VLA on 1-clip routing and straight-channel seating in Table 2. ACT and DiT VLA results are averaged over three evaluation seeds. With privileged state observations, PPO achieves 95.32% and 82.33% success, respectively. Despite plateauing training losses, image-based ACT and DiT VLA achieve lower success rates, particularly in channel seating, with 18.80% and 15.62% success, respectively. Dataset sizes, training configurations, and checkpoint selection are detailed in Appendix G.

Table 2: Evaluation results on 1-clip routing and straight channel seating tasks in simulation.

Task	Setting	State PPO	ACT	DiT VLA
Clip routing	1-clip	$\mathbf{95.32 \pm 0.93}$	64.06 ± 11.16	39.06 ± 16.16
Channel seating	Straight channel	$\mathbf{82.33 \pm 0.31}$	18.80 ± 7.65	15.62 ± 12.88

6 Real-World Validation

For real-world validation, we evaluate all three WireCraft tasks on a physical UR5 using ACT, which achieved the highest insertion success rate among image-input baselines in Section 5. We first reproduce the physical setup in simulation, then train ACT on four combinations of real demonstrations (R), scripted simulation demonstrations (S), and simulation RL rollouts (P): R, R+S, R+P, and R+S+P. All configurations train for 80k steps, with checkpoints selected by the lowest L1 loss on held-out real demonstrations. Preliminary simulation-only policies showed poor zero-shot transfer, so we focus the quantitative evaluation on mixtures containing real demonstrations.

The hardware evaluation uses a narrower randomization range than simulation. For insertion, the connector is placed within a 3×3 cm region with varying tilt, while the socket remains fixed. For 1-clip routing, a single clip is randomly positioned near the wire on both sides. For straight-channel seating, one wire endpoint remains fixed while the other is placed at varying positions outside the channel. We evaluate each task and data-source combination over 20 real-robot trials, and report different effects of simulation data across tasks in Figure 5.

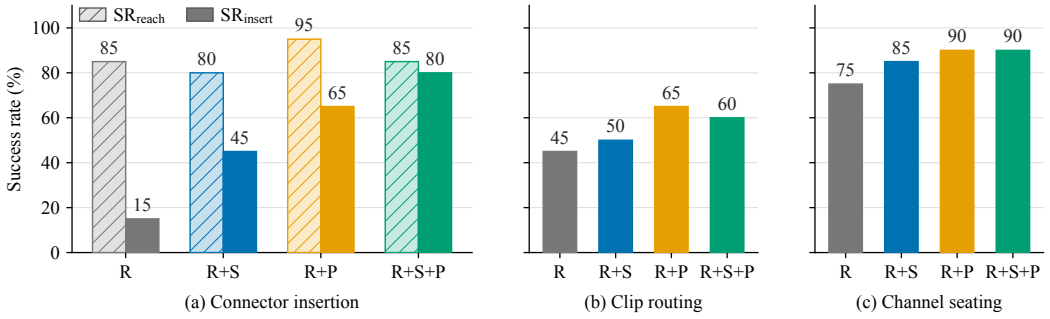


Figure 5: ACT success rates on the physical UR5 for (a) Ethernet-style connector insertion, (b) 1-clip routing, and (c) straight-channel seating. Insertion reports both $\mathrm{SR}_{\mathrm{reach}}$ and $\mathrm{SR}_{\mathrm{insert}}$; routing and seating each report task success.

For insertion, reach success remains between 80% and 95% across data combinations, while insertion success increases from 15% with R to 80% with R+S+P. Higher reach success does not necessarily translate into higher insertion success: R+P reaches more often than R+S+P (95% vs. 85%), but inserts less often (65% vs. 80%). Qualitatively, R+S+P more often produces insertion-compatible grasp poses, whereas R+P sometimes reaches the socket with a grasp that leaves the connector poorly aligned for insertion. The benefit of broader variability of data coverage is therefore associated with grasping the connector in a pose that supports subsequent socket alignment. Successful pickup and reach alone are insufficient.

Clip routing also benefits from simulated data, but the improvement is non-monotonic: success increases from 45% with R to 65% with R+P, while R+S+P achieves 60%. Qualitatively, failures with simulated training data more often place the wire on top of the clip opening instead of beside the clip. These near-misses remain unsuccessful under the binary criterion but indicate closer placement alignment. Clip localization during wire grasping relies on the side-camera view alone, leaving limited visual information for preparing the subsequent placement.

For channel seating, simulated data increase success from 75% with R to 90% with R+P and R+S+P. Most failures occur during the initial grasp; once the wire is successfully grasped, seating is usually completed. Its remaining bottleneck is therefore grasp acquisition, unlike the final alignment and placement failures observed in insertion and clip routing.

Overall, simulated trajectories improve the observed success rates across all three tasks, with task-dependent effects on grasp selection and fixture engagement. The hardware results support the portability of the three task definitions under restricted initialization ranges. Since source diversity and dataset size increase together, these comparisons show the benefit of supplementing real demon-

strations with simulated trajectories, without isolating the effect of diversity alone. Further setup and failure details are provided in Appendix H.

7 Limitations

WireCraft has several limitations. First, policy transfer between the articulated and FEM-based wire models has not been evaluated, and their dynamic deformation and contact response remain quantitatively unmatched. Neither model has been calibrated against the physical wire, so hardware task validation does not establish wire-model accuracy or reliable zero-shot transfer. Second, full force/torque feedback remains unused: privileged insertion uses only saturated fingertip contact magnitudes, while the UR5's mounted force/torque sensor is unread. Simulated contact forces remain physically unvalidated, and the benefits of force/torque feedback and optional depth observations for task completion are untested. Third, hardware evaluation covers all three task families but only selected settings on one UR5, leaving generalization across robots, fixtures, and wire materials untested. Success metrics also do not capture contact loads or wire damage. Finally, the VLA results reflect constrained fine-tuning and do not establish an upper bound on VLA capability; longer training, improved checkpoint selection, and stronger visual adaptation remain future work.

8 Conclusion

We presented WireCraft, a simulation benchmark for industrial DLO manipulation across connector insertion, clip routing, and channel seating. WireCraft provides configurable 3D-printable assets, articulated and FEM-based wire representations, shared policy interfaces, and trajectory data from scripted, RL, and real-world sources, together with teleoperation tools. Our experiments show a clear gap between task feasibility and current vision-based policy performance. Privileged state-based RL solves the evaluated insertion, routing, and seating settings, indicating that the tasks are well posed in simulation. In contrast, vision-based RL, imitation learning, and VLA policies degrade sharply on connector insertion, while ACT and DiT VLA also fall below State PPO on routing and seating. The reach–insert gap shows that reaching the socket neighborhood is not sufficient: fine alignment, occlusion, deformation, and contact-rich seating remain the central bottlenecks. Real-world UR5 validation covers all three task families and shows improved ACT success rates when real demonstrations are supplemented with simulated trajectories.

By releasing WireCraft, we aim to provide a common substrate for studying where industrial DLO policies fail and how they improve, including visual localization, deformation-aware control, contact-rich insertion, fixture-constrained routing, and sim-to-real transfer.

Acknowledgments

This research has been partially funded by the Industrial Technology Innovation Program [P0030285, Autonomous Assembly through AI Agent-Driven Dexterous Manipulation of Flexible and Complex Industrial Components] of the Ministry of Trade, Industry and Energy of the Republic of Korea.

References

[1] J. Zhu, A. Cherubini, C. Dune, D. Navarro-Alarcon, F. Alambeigi, D. Berenson, F. Ficuciello, K. Harada, J. Kober, X. Li, et al. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77, 2022.

[2] X. Zhang, H.-C. Lin, Y. Zhao, and M. Tomizuka. Harnessing with twisting: Single-arm deformable linear object manipulation for industrial harnessing task. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4069–4075. IEEE, 2024.

[3] A. Wilson, H. Jiang, W. Lian, and W. Yuan. Cable routing and assembly using tactile-driven motion primitives. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10408–10414. IEEE, 2023.

[4] J. Luo, C. Xu, X. Geng, G. Feng, K. Fang, L. Tan, S. Schaal, and S. Levine. Multistage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 40: 1476–1491, 2024.

[5] S. Jin, W. Lian, C. Wang, M. Tomizuka, and S. Schaal. Robotic cable routing with spatial representation. *IEEE Robotics and Automation Letters*, 7(2):5687–5694, 2022. doi:10.1109/LRA.2022.3158377.

[6] M. Yu, H. Zhong, and X. Li. Shape control of deformable linear objects with offline and online learning of local linear deformation models. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1337–1343. IEEE, 2022.

[7] F. Gu, Z. Wang, Z. Zhu, J. Ma, Y. Zhou, S. Jiang, and B. He. A survey on robotic manipulation of deformable objects: Recent advances, open challenges and new frontiers. *Neurocomputing*, page 134058, 2026.

[8] W. Zhang, K. Schmeckpeper, P. Chaudhari, and K. Daniilidis. Deformable linear object prediction using locally linear latent dynamics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13503–13509. IEEE, 2021.

[9] M. Li and C. Choi. Learning for deformable linear object insertion leveraging flexibility estimation from visual cues. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, page 5183–5189. IEEE, May 2024. doi:10.1109/icra57147.2024.10610419. URL http://dx.doi.org/10.1109/ICRA57147.2024.10610419.

[10] B. Cao, X. Zang, X. Zhang, Z. Chen, S. Li, and J. Zhao. Shape control of elastic deformable linear objects for robotic cable assembly. *Advanced Intelligent Systems*, 6(7):2300835, 2024.

[11] M. Yan, Y. Zhu, N. Jin, and J. Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE robotics and automation letters*, 5(2):2372–2379, 2020.

[12] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

[13] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, V. N. Rajesh, Y. W. Choi, Y.-R. Chen, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

[14] X. Lin, Y. Wang, J. Olkin, and D. Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 432–448. PMLR, 2021.

[15] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.

[16] S. Chen, Y. Xu, C. Yu, L. Li, X. Ma, Z. Xu, and D. Hsu. Daxbench: Benchmarking deformable object manipulation with differentiable physics. In *The Eleventh International Conference on Learning Representations*, 2023.

[17] R. Laezza, R. Gieselmann, F. T. Pokorny, and Y. Karayiannidis. Reform: A robot learning sandbox for deformable linear object manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4717–4723, 2021. doi:10.1109/ICRA48506.2021.9561766.

[18] J. Cao, Y. Wang, Z. Xiong, C. Lin, Z. Chen, and C. Gan. DLO-Lab: Benchmarking deformable linear object manipulations with differentiable physics. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/63391. Accepted.

[19] M. Li, H. Yu, Y. Huang, Y. Hong, C. Choi, and H. Ye. Hierarchical dlo routing with reinforcement learning and in-context vision-language models, 2025. URL https://arxiv.org/abs/2510.19268.

[20] J. Tanureza, B. Michalak, M. Radke, and K. Haninger. Industrial cabling in constrained environments: a practical approach and current challenges. In *2025 IEEE/SICE International Symposium on System Integration (SII)*, pages 1428–1433. IEEE, 2025.

[21] C. Kienle, B. Alt, F. Schneider, T. Pertlwieser, R. Jäkel, and R. Rayyes. Ai-based framework for robust model-based connector mating in robotic wire harness installation. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 2444–2449. IEEE, 2025.

[22] National Institute of Standards and Technology. Assembly Performance Metrics and Test Methods. https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly, 2018. Accessed: 2026-05-22.

[23] Y. Chen, K. Kimble, E. H. Adelson, T. Asfour, P. Chanrungmaneekul, S. Chitta, Y. Chitambar, Z. Chen, K. Goldberg, D. Kragic, et al. Manipulationnet: An infrastructure for benchmarking real-world robot manipulation with physical skill challenges and embodied multimodal reasoning. *arXiv preprint arXiv:2603.04363*, 2026.

[24] Z. Huang, Y. Hu, T. Du, S. Zhou, H. Su, J. B. Tenenbaum, and C. Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations*, 2021.

[25] Y. Zhang, K. S. Luck, F. Verdoja, V. Kyrki, and J. Pajarinen. Modesuite: Robot learning task suite for benchmarking mobile manipulation with deformable objects. *IEEE Robotics and Automation Letters*, 2026.

[26] Q. J. Chen, T. Bretl, and Q.-C. Pham. Accurate simulation and parameter identification of deformable linear objects using discrete elastic rods in generalized coordinates. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 20454–20460, 2025. doi:10.1109/IROS60139.2025.11247160.

[27] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, and E. Grinspun. Discrete elastic rods. In *ACM Siggraph 2008 Papers*, pages 1–12. 2008.

[28] Z. Sun, J. Zhu, and R. B. Fisher. DexDLO: Learning goal-conditioned dexterous policy for dynamic manipulation of deformable linear objects. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16009–16015. IEEE, 2024. doi:10.1109/ICRA57147.2024.10610754.

[29] Y. Chen, Y. Zhang, Z. Brei, T. Zhang, Y. Chen, J. Wu, and R. Vasudevan. Differentiable discrete elastic rods for real-time modeling of deformable linear objects. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2996–3014. PMLR, 2025. URL https://proceedings.mlr.press/v270/chen25d.html.

[30] Y. Chen, X. Wu, Y. Zong, Y. Chen, A. Li, B. Zhang, and R. Vasudevan. DEFT: Differentiable branched discrete elastic rods for modeling furcated DLOs in real-time. *arXiv preprint arXiv:2502.15037*, 2025. doi:10.48550/arXiv.2502.15037. URL https://arxiv.org/abs/2502.15037.

[31] A. Govoni, N. Zubair, S. Soprani, and G. Palli. Performance analysis of a mass-spring-damper deformable linear object model in robotic simulation frameworks. In *European Robotics Forum 2025*, volume 36 of *Springer Proceedings in Advanced Robotics*, pages 187–192. Springer, 2025. doi:10.1007/978-3-031-89471-8_29.

[32] M. Li, H. Yu, and C. Choi. Routing manipulation of deformable linear object using reinforcement learning and diffusion policy. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–07, 2025. doi:10.1109/ICRA55743.2025.11127451.

[33] T. Z. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine. Offline meta-reinforcement learning for industrial insertion. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6386–6393. IEEE, 2022.

[34] B. Cao, X. Zang, S. Li, X. Zhang, C. le Li, and J. Zhao. Robotic cable following and insertion with tactile sensing. *IEEE Sensors Journal*, 26:6216–6224, 2026. URL https://api.semanticscholar.org/CorpusID:284252279.

[35] H. Zhou, S. Li, Q. Lu, and J. Qian. A practical solution to deformable linear object manipulation: A case study on cable harness connection. In *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 329–333. IEEE, 2020.

[36] M. Yu, K. Lv, C. Wang, Y. Jiang, M. Tomizuka, and X. Li. Generalizable whole-body global manipulation of deformable linear objects by dual-arm robot in 3-d constrained environments. *The International Journal of Robotics Research*, 44(4):607–639, 2025.

[37] W. Li, Z. Zhu, D. Li, Z. Gong, N. Lv, H. Chen, and J. Li. An automated cable laying simulation method by robot for planar routing. In *Proceedings of the 2025 11th Annual International Conference on Network and Information Systems for Computers*, ICNISC '25, page 209–214, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715839. doi:10.1145/3776942.3777006. URL https://doi.org/10.1145/3776942.3777006.

[38] P. Chang, R. Luo, M. Zolotas, and T. Padır. Manipulation of deformable linear objects in benchmark task spaces. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1910–1916, 2022. doi:10.1109/CASE49997.2022.9926677.

[39] K. Kimble, J. Albrecht, M. Zimmerman, and J. Falco. Performance measures to benchmark the grasping, manipulation, and assembly of deformable objects typical to manufacturing applications. *Frontiers in Robotics and AI*, 9:999348, 2022.

[40] O. Azulay, K. Kondap, J. Drake, S. Xie, H. Li, S. Chitta, and K. Goldberg. Motorcycle 1.0: Automating bimanual cable routing around fixtures on the nist task board. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 2636–2641. IEEE, 2025.

[41] Intrinsic. AI for Industry Challenge Toolkit. https://github.com/intrinsic-dev/aic, 2026. Accessed: 2026-10-08.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[44] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems (RSS)*, 2023. doi:10.15607/RSS.2023.XIX.016.

[45] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[46] W. Peebles and S. Xie. Scalable diffusion models with transformers. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182, 2023. doi:10.1109/iccv51070.2023.00387.

[47] Z. Hou, T. Zhang, Y. Xiong, H. Duan, H. Pu, R. Tong, C. Zhao, X. Zhu, Y. Qiao, J. Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7686–7697, 2025.

[48] K. Black, N. Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of the 9th Conference on Robot Learning (CoRL)*, 2025. Oral Presentation.

[49] T. Hoang, H. Le, P. Becker, V. A. Ngo, and G. Neumann. Geometry-aware rl for manipulation of varying shapes and deformable objects. In *International Conference on Learning Representations*, volume 2025, pages 47376–47405, 2025.

[50] E. Xing, V. Luk, and J. Oh. Stabilizing reinforcement learning in differentiable multiphysics simulation. In *International Conference on Learning Representations*, volume 2025, pages 91165–91198, 2025.

[51] M. Mittal et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[52] NVIDIA. Nvidia isaac sim. https://github.com/isaac-sim/IsaacSim, 2026. URL https://docs.isaacsim.omniverse.nvidia.com/latest/index.html.

[53] Isaac Lab Project Developers. Interacting with a Deformable Object. Isaac Lab Documentation, 2026. URL https://isaac-sim.github.io/IsaacLab/main/source/tutorials/01_assets/run_deformable_object.html. Accessed: 2026-05-16.

[54] NVIDIA. Soft Bodies. PhysX Documentation, 2024. URL https://nvidia-omniverse.github.io/PhysX/physx/5.4.0/docs/SoftBodies.html. Accessed: 2026-05-16.

[55] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, 2024.

[56] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022.

[57] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. In *Robotics: Science and Systems*, 2018.

[58] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, S. Palma, P. Kooijmans, M. Aractingi, M. Shukor, D. Aubakirova, et al. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

A Related Work

We expand the related-work discussion from the main paper with a feature-level comparison against prior DLO and deformable-object benchmarks, summarized in Table 4. For each benchmark we mark which of the three industrial task families it covers, what wire physics it uses, which learning baselines it supports, and whether it offers simulated robot evaluation, physical robot validation, and multi-source demonstrations. Most prior benchmarks address only a subset of these dimensions. WireCraft is the only one that spans all three task families with both articulated and deformable wire models, supports RL, IL, and VLA baselines together, and pairs simulated evaluation with physical UR5 validation.

B DLO Physics and Simulation Details

This section expands on the DLO physics summarized in Section 3.1. WireCraft is built on Isaac Sim 4.5 [52] and Isaac Lab v2.2.1 [51], and is additionally verified on Isaac Sim 5.1. We first describe the simulation backend, then the two wire models and their material and contact parameters, the DLO–connector interaction, and finally the computational scaling behavior of each model.

B.1 Simulation Backend

WireCraft runs on Isaac Sim for photo-realistic rendering and physics, and on Isaac Lab for GPU-accelerated, multi-instance parallel simulation used in both policy training and trajectory generation. The default RL configuration uses a PhysX physics timestep of $1/120$ s with a control decimation of 2, giving a 60 Hz control rate and an 8 s episode horizon. We verified WireCraft across multiple GPUs, namely the NVIDIA GeForce RTX 4080 SUPER, the RTX PRO 4500 Blackwell, and the RTX A4000 and RTX 5070Ti.

B.2 DLO Models

WireCraft provides two complementary wire representations for the same benchmark tasks. They are not intended to be parameter-by-parameter calibrated material matches, since the articulated chain and FEM soft body expose different solver-native parameters. Instead, they are aligned at the task level through shared wire endpoints, connector interfaces, fixtures, observation-action schemas, and success criteria. The articulated wire is a chain of rigid capsule segments connected by 6-DoF joints (PhysX D6 joints) whose angular degrees of freedom are driven by passive implicit spring–damper actuators, simulated under rigid-body dynamics. This representation supports high-throughput trajectory generation but can become unstable under adjacent-segment self-contact or accumulated contact forces. The deformable wire is a PhysX FEM soft body: a slender triangulated cylindrical rod (decagonal cross-section, 10 radial × 40 axial subdivisions; length 0.4 m, radius 3 mm) that PhysX voxelizes into a hexahedral FEM simulation mesh (resolution 40 along the major axis). This captures local bending, compression, and contact deformation at higher fidelity but greater cost. The parameter names reflect each model's native representation and should not be read as physically equivalent material settings.

Contact and material parameters. Beyond the per-wire settings in Table 3, contact behavior is governed by the connector, gripper, and table materials. The rigid connectors use a mass of 0.01 kg and isotropic friction 2.0 (static and dynamic, with a `max` friction-combine mode); the gripper inner-finger pads use friction 5.0 to provide a firm grasp on the connector and wire; and the table and scene default material uses friction 1.0. The articulated wire uses restitution 0.0, while the deformable wire exposes no restitution term. These are fixed defaults; the connector-insertion data pipeline additionally randomizes the connector planar position and orientation and the socket clearance for demonstration diversity, as detailed in Appendix C.4.

Table 3: **Representative wire model parameters.** Values reflect each model's native representation and should be interpreted as solver-specific benchmark settings rather than physically equivalent material parameters. Articulated wire values correspond to the committed task-environment configuration.

Setting	FEM-based Deformable Wire	Articulated Wire
Representation	Isaac Sim deformable body volume (PhysX FEM)	Rigid cylinder chain with D6 joints
Total length	0.4 m	0.4 m
Segments / resolution	40 segments; hexahedral resolution 40	12 rigid segments; $\sim$0.033 m per segment
Radius / density	0.003 m; 800.0	0.004 m; 2000.0
Friction	Dynamic 5.0; static does not support deformable body	Static 1.0; dynamic 1.0
Material / joint parameters	Young's modulus 5.0×10^7; Poisson's ratio 0.49; vertex-velocity damping 3.0	Joint stiffness 5.0; damping 5.0; joint limit 0.5 rad
Solver / contact settings	32 position iterations; contact offset 0.001; rest offset 0.0	32 position iterations; 4 velocity iterations; max depenetration velocity 1.0
Collision / body settings	Collision simplification enabled; sleep threshold 0.01	Self-collision enabled; max linear/angular velocity 5.0/60.0

B.3 DLO–Connector Interaction

Each connector is coupled to its DLO at the endpoints. For the deformable wire, each rigid connector is bound to the soft body through a PhysX auto-attachment (`PhysxPhysicsAttachment` with deformable-vertex attachments enabled and a vertex overlap offset of 0.015 m), applied at both ends; forces transfer through the attached deformable vertices during grasping, routing, and insertion. For the articulated wire, the endpoint connector is rigidly welded to the adjacent segment with a fixed joint. Figure 6 visualizes the resulting DLO–Connector behavior for both representations across representative simulation frames.

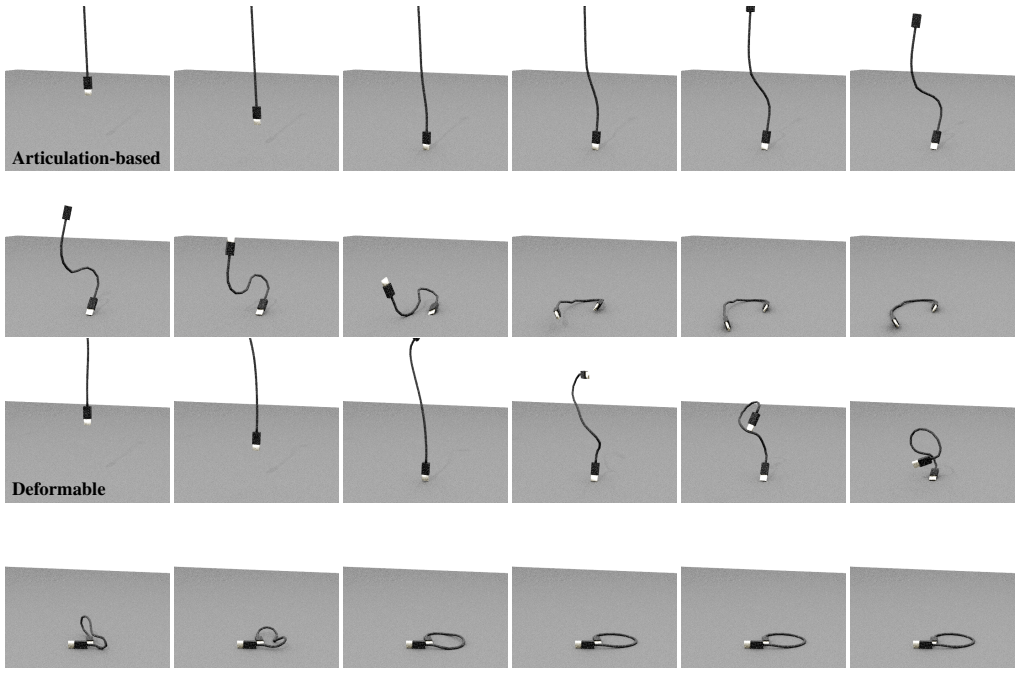


Figure 6: **DLO–Connector Physics Demonstration.** Visualization of articulated and deformable wire concatenated with a DisplayPort connector across representative simulation frames.

Table 4: **Comparison with related DLO and deformable-object benchmarks.** WireCraft combines industrial DLO task families, articulated and FEM-based deformable wire simulation, RL/IL/VLA baselines, simulated-robot evaluation, and physical UR5 validation within a unified benchmark framework. *Insertion*, *Routing*, and *Seating* refer to end/part insertion, fixture-guided routing, and placement into extended grooves or channels, respectively. *Multi-Tier* indicates multiple task levels, task families, or difficulty settings; *Multi-Source* indicates demonstrations or trajectories from multiple sources under a shared schema. Deformable cells report the specified physics representation when available. ✓: supported. ✗: not supported. –: not applicable. Benchmarks are sorted by publication year.

Benchmark	*Insertion*	*Routing*	*Seating*	*Multi-Tier*	*Articulated Wire*	*Deformable Wire*	*RL Baseline*	*IL Baseline*	*VLA Baseline*	*Simulated Robot*	*Physical Robot*	*Multi-Source*
RLBench [12] (2020)	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
Zhou et al. [35] (2020)	✓	✗	✗	✗	–	–	✗	✗	✗	✗	✓	✗
DeformableRavens [15] (2021)	✗	✗	✓	✓	✓	✓ FEM	✗	✓	✗	✓	✓	✗
PlasticineLab [24] (2021)	✗	✓	✗	✓	✗	✓ MPM	✓	✗	✗	✓	✗	✗
ReForm [17] (2021)	✓	✓	✗	✓	✓	✓ AGX/LEM	✓	✗	✗	✓	✗	✗
SoftGym [14] (2021)	✗	✗	✗	✓	✗	✓ PBD	✓	✗	✗	✓	✓	✗
NIST ATB [39] (2022)	✓	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
Jin et al. [5] (2022)	✓	✓	✗	✓	–	–	✗	✓	✗	✗	✓	✗
Zhao et al. [33] (2022)	✓	✗	✗	✓	–	–	✓	✓	✗	✗	✓	✓
Chang et al. [38] (2022)	✓	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
DaXBench [16] (2023)	✗	✗	✗	✓	✗	✓ MPM/MS	✓	✓	✗	✓	✓	✗
Wilson et al. [3] (2023)	✓	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
Luo et al. [4] (2024)	✓	✓	✗	✓	–	–	✗	✓	✗	✗	✓	✗
DexDLO [28] (2024)	✗	✗	✗	✓	✓	✓ ART	✓	✗	✗	✓	✗	✗
Li and Choi [9] (2024)	✓	✗	✗	✓	✗	✓ PBD	✓	✗	✗	✓	✓	✗
Zhang et al. [2] (2024)	✓	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
Chen et al. [26] (2025)	✗	✗	✗	✗	✓	✓ DER^{H}	✗	✗	✗	✓	✓	✗
DEFORM [29] (2025)	✗	✗	✗	✗	✗	✓ DER	✗	✗	✗	✓	✓	✗
Yu et al. [36] (2025)	✗	✗	✗	✓	✗	✓ DER	✗	✗	✗	✓	✓	✗
ManiSkill3 [13] (2025)	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓
DEFT [30] (2025)	✓	✗	✗	✓	✗	✓ B-DER	✗	✗	✗	✗	✓	✗
Govoni et al. [31] (2025)	✗	✗	✗	✗	✗	✓ MSD	✗	✗	✗	✓	✗	✗
Li et al. [32] (2025)	✓	✓	✗	✓	✗	✓ PBD	✓	✓	✗	✓	✓	✗
Li et al. [19] (2025)	✓	✓	✗	✓	✗	✓ Particle	✓	✗	✗	✓	✓	✗
Kienle et al. [21] (2025)	✓	✗	✗	✓	–	–	✗	✗	✗	✗	✓	✗
Tanureza et al. [20] (2025)	✓	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
Li et al. [37] (2025)	✗	✓	✗	✗	✗	✓ MSD	✗	✗	✗	✓	✗	✗
Azulay et al. [40] (2025)	✗	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
MoDeSuite [25] (2026)	✗	✗	✗	✓	✗	✓ FEM/PBD	✓	✓	✗	✓	✓	✗
DLO-Lab [18] (2026)	✓	✓	✗	✓	✗	✓ Diff. DER	✓	✗	✗	✓	✓	✗
Cao et al. [34] (2026)	✗	✓	✗	✓	–	–	✗	✗	✗	✗	✓	✗
WireCraft (Ours) (2026)	✓	✓	✓	✓	✓	✓ FEM	✓	✓	✓	✓	✓	✓

B.4 Computational Scaling and Throughput

We benchmarked the deformable and articulated wire models in matched DLO-only scenes with ground contact enabled and endpoint connectors disabled. Each run used the $1/120$ s physics timestep without rendering and applied a 5 s warmup followed by a 60 s measured rollout per configuration. The sweep covered 64, 128, 256, 512, 1024, 2048, 4096, and 8192 parallel environments, stopping each model after its first failed point; the articulated wire was additionally probed at 16,384, where setup failed. The deformable curve uses the configuration with 32 solver position iterations. Because the two models expose different physical and numerical parameters, we compare their computational scaling under matched external task conditions rather than treating them as material-calibrated equivalents.

As shown in Figure 7, which is measured on an NVIDIA RTX A4000 GPU, the articulated model scales successfully to 8192 parallel environments, while the deformable model succeeds through 2048 environments and fails at 4096. The deformable wire is faster at the smallest environment counts, but the articulated wire reaches substantially higher aggregate throughput, whereas the deformable wire shows much weaker throughput scaling before reaching its limit.

The deformable limit is consistent with the per-scene cap of 4095 deformable-volume actors enforced by the PhysX constant `PX_MAX_NB_DEFORMABLE_VOLUME`, which originates from the 12-bit field used to pack the (deformable-volume id, tetrahedron id) pair into a single 32-bit narrow-phase index. In practice, this hard ceiling is not the only constraint: GPU memory for deformable-volume contact buffers and the solver heap (`PxGpuDynamicsMemoryConfig`) is statically pre-allocated, does not grow dynamically, and may bind first depending on mesh resolution, contact density, and available device memory [54]. Because neither constraint is exposed as a runtime knob in Isaac Sim, we treat the deformable model as our higher-fidelity, lower-throughput representation and use the articulated model for high-throughput RL training and bulk trajectory generation; the deformable model is retained for contact-sensitive evaluation and physically detailed trajectory rollouts. Absolute throughput should be read in the context of each model's discretization and solver settings.

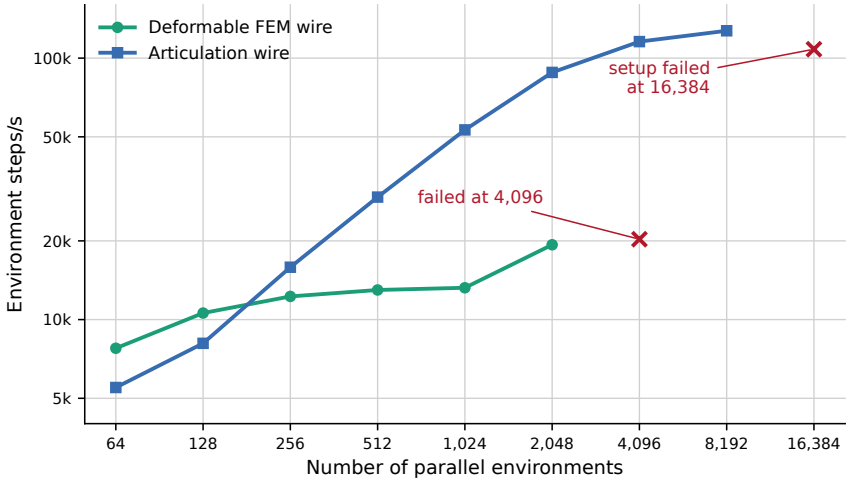


Figure 7: **Scaling behavior of the two WireCraft wire models.** Both models were evaluated in matched headless DLO-only scenes with a $1/120$ s physics timestep, 5 s warmup, 60 s measured rollout. Absolute throughput depends on discretization and solver settings; the comparison is intended to show scaling behavior of the selected benchmark configurations. The articulation model scales to 8192 successful environments, while the deformable solver-32 model succeeds through 2048 environments and fails at 4096 owing to PhysX deformable-volume limits.

C Task Design

C.1 Connector Insertion

Connector insertion requires the robot to grasp the rigid connector at the end of a deformable wire and insert it into a matching socket. The task combines local peg-in-hole precision with global deformation effects: the trailing wire introduces drag, gravity-induced shape changes, and contact disturbances that perturb the connector during final alignment.

The task curriculum is set by connector and socket geometry, which is fixed per task rather than randomized across episodes. WireCraft registers five connector/socket geometries that span a range of rotational symmetries and alignment sensitivities. The cylinder connector requires the held connector's long axis to align with the socket axis but is rotationally symmetric about that axis, making it the easiest variant. The cuboid connector extends the cylinder requirement and additionally requires alignment of the rotation about the long axis. The Ethernet-style connector adds a latch-like geometry and serves as our primary insertion task; the DisplayPort connector has an asymmetric shell and is the most alignment-sensitive; and the USB-A connector provides a reference-standard geometry. Among these, the cylinder, cuboid, Ethernet-style, and DisplayPort connectors are used in the benchmark evaluations (Table 1); the USB-A connector is released as an additional asset but is not included in the reported experiments. A second difficulty axis is the socket clearance, which controls the gap between connector and socket; a smaller clearance yields a tighter, harder insertion.

C.2 Clip Routing

Clip routing requires the robot to route a DLO through one or more structured U-slot clips mounted on the workspace. The task emphasizes fixture interaction and topological constraints: the DLO must be guided into the clip openings while maintaining contact stability and avoiding incomplete seating or slippage.

The task curriculum is set by the number of clips, fixed per task. Difficulty scales from single-clip routing to multi-clip sequences of up to three clips, where additional clips lengthen the multi-step route and require intermediate re-grasps. Each clip's x position is fixed along the routing direction, while its y position and yaw are randomized per episode. A secondary difficulty axis is the clip opening: the top-gap and lip retention can be adjusted, and a sufficiently wide opening removes the lip and reduces the clip to an unconstrained U-slot.

C.3 Channel Seating

Channel seating requires the robot to lay a DLO into a geometric channel so that the DLO body conforms to the target path. Unlike connector insertion, where success depends on a local terminal pose, channel seating evaluates extended shape control over the entire DLO body.

The task curriculum is set by channel geometry, fixed per task. Each channel is constructed from five segments, and difficulty is controlled by the random relative rotation applied at each segment-to-segment joint: the straight channel uses zero inter-segment rotation and tests simple alignment along a linear path, while the curved channel applies a per-joint random bend of up to $\pm 20^\circ$, producing a non-linear path that requires continuous path-following.

C.4 Randomization

Each task applies per-episode randomization at reset, on top of the fixed per-task curriculum described above. Connector insertion randomizes the wire-attached connector and the socket; clip routing randomizes the clips; and channel seating randomizes the wire-attached connector and, optionally, the channel shape. The full set of ranges is listed in Table 5. The reported insertion ranges correspond to the RL training and evaluation configuration; the scripted and teleoperation data-collection pipelines use a different connector planar range, which we treat as dataset-generation settings rather than the benchmark task randomization.

Table 5: Per-episode randomization for the WireCraft task families, grouped by the randomized component. The connector is attached to the DLO terminal and randomized before a short free-fall settling drop. Ranges are reported for the RL training and evaluation configuration.

Component	Parameter	Default range / description
Connector	Spawn height	6–11 cm above the table, then free-fall settling
	Roll / pitch	Uniform $[-18^\circ, 18^\circ]$
	Yaw	Uniform $[-36^\circ, 36^\circ]$
	X offset	Uniform ± 5 cm
	Y offset	Uniform ± 15 cm
Socket	X position	Fixed along the insertion axis
	Y position	Randomized within a 28 cm range
	Z position	Randomized within a 15 cm range
Clips	X position	Fixed along the routing line
	Y position	Uniform within ± 6 cm of the nominal lane
	Yaw	Uniform $[-15^\circ, 15^\circ]$
Channels	Position	Fixed
	Shape	5 segments; each adds a signed bend of up to $\pm 20^\circ$ to the running heading

C.5 Control Modes

WireCraft exposes three end-effector control modes, selectable per task, so that users can match the action space to the task's requirements. *Joint control* commands joint-position targets directly. *Cartesian control* commands a 6-DoF end-effector delta pose, resolved to joint targets by a differential-IK controller. *Cartesian control with fixed orientation* commands only the 3-DoF translation while holding the end-effector orientation fixed, which suits tasks where a constant tool pose is desired. Connector insertion and clip routing use full Cartesian control, since both require reorienting the held object; channel seating uses the fixed-orientation mode to keep the connector level while laying the DLO into the channel. All three modes share the same gripper command and downstream joint-target interface, so a task can switch modes without changing the policy's action head.

D Scripted Policy

All three task families share a common scripted-policy framework: a per-environment vectorized finite-state machine that, within each phase, reads the current end-effector and fixture poses, expresses a target end-effector pose in the robot base frame, and emits a Cartesian delta-pose action obtained by clamping the pose error toward that target. Each policy selects the control mode best suited to its task (see Appendix C.5). A phase advances once the end-effector reaches its target within a position and orientation tolerance and dwells for a minimum number of control steps. At the grasp phase the policy snapshots the relevant poses, so that every post-grasp phase servos toward a stable reference that is unaffected by the deformation introduced once the wire is held. The three policies differ in what they grasp, in their task-specific sub-skills, and in the perturbations they inject to diversify demonstrations, as summarized in Table 6.

D.1 Insertion

The insertion policy grasps the connector at its tail and seats it into the socket. After grasping and lifting, it moves to a pre-insertion pose, solves for the gripper orientation that aligns the held connector with the socket under a tightened orientation tolerance, drives the connector into the socket, and releases. To produce demonstrations of failure and recovery, with probability 0.3 the policy injects a pick-and-drop perturbation, lifting and releasing the connector mid-air before re-approaching

Table 6: Comparison of the three scripted policies. All share the vectorized finite-state-machine framework and a Cartesian delta-pose action; they differ in their grasp target, control dimensionality, core sub-skill, and the perturbations injected for demonstration diversity.

	Insertion	**Clip routing**	**Channel seating**
Grasp target	Connector tail	DLO body node near the clip	Connector
Action	7-DoF (pose + gripper)	7-DoF (pose + gripper)	7-DoF with fixed end-effector orientation
Core sub-skill	Align and drive into the socket	Over-stretch, then push the DLO into the slot	Lift and slowly place into the channel
Reference frame	Socket pose	Clip pose with a frozen anchor-to-clip direction	Channel center and length
Success	Connector seated in the socket	DLO node inside the clip slot	DLO body laid within the channel over 80%
Injected perturbation	Pick-and-drop (30%) and failure retry	Pre-stretch with a frozen target direction	Delayed grasp and slowed placement

and regrasping, and a failure-triggered retry returns to the approach phase if the insertion has not succeeded after the hold phase, for up to two attempts. Per-episode randomization of the approach pitch and the tail grasp offset, together with phase-local slowdowns around the pick and insertion, further diversify the demonstrations.

D.2 Clip Routing

The clip-routing policy grasps the deformable wire itself at the node nearest the clip, rather than the connector, since the connector is wider than the clip mouth and cannot enter the slot. The grasp node is located by proximity and the grasp orientation is set to the local wire tangent. After descending to a fixed table-relative height and closing on the wire, the policy lifts and rotates the held wire to the clip's slot axis. It then over-stretches the wire by translating past the placement target along the anchor-to-clip direction, pulls back to the clip front to release excess tension, and pushes the held wire down past the clip lip so that it seats into the slot under a controlled descent. Success requires a wire node to lie inside the clip slot. The anchor-to-clip direction is frozen at grasp time because the wire's far endpoint is a dynamic body that drifts under tension, which would otherwise cause the placement target to move every step.

D.3 Channel Seating

The channel-seating policy grasps the connector and lays the wire into the channel so that its body follows the target path. Unlike the other two policies, which use full Cartesian control, it uses the fixed-orientation Cartesian mode with a much finer action scale to keep the connector level and produce smooth, non-snagging motion. It grasps the connector gently, holding the gripper open for a settling delay before closing, then lifts vertically and settles. It then descends along the channel path at a strongly reduced speed, seating the wire at the channel-center height before releasing. Following the path rather than descending at a single waypoint lets the same policy handle both straight and curved channels.

D.4 Scripted Demonstrations

See Figure 8.

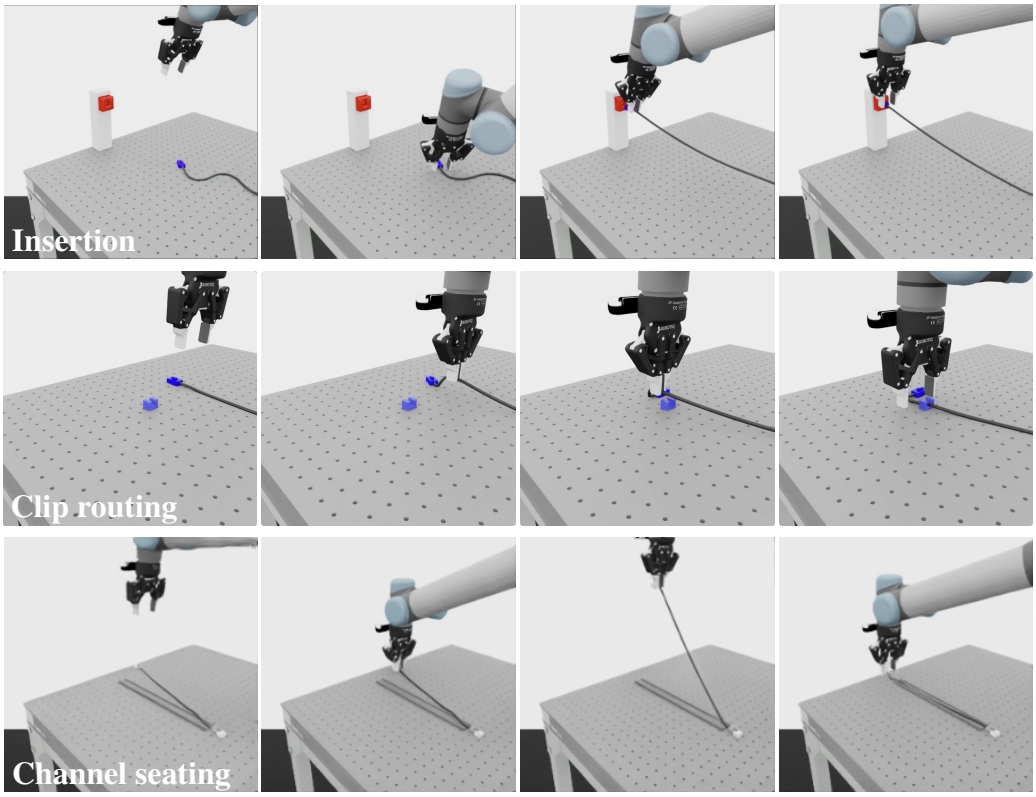


Figure 8: **Scripted demonstration rollouts.** Each row shows a scripted policy executing one task family across representative phases, progressing from the initial state through grasp and manipulation to task completion.

E RL Experiment Detail

E.1 State-based RL

We detail the privileged RL configuration on connector insertion, our primary RL setting. On insertion, the two privileged baselines, PPO and SACfD, share the same observation, action, and reward, and differ only in the learning algorithm. Clip routing and channel seating reuse the same PPO algorithm, network, and hyperparameters without per-task tuning, under their own task-specific observations and rewards, and we do not detail them further here. All RL results in Section 5 are reported as mean $\pm$ standard deviation across 3 seeds.

Observation. The insertion observation is a 42-dimensional privileged state vector (`observation_space=42`) : arm joint positions (6) and gripper position (1); arm joint velocities (6) and gripper velocity (1); end-effector position (3) and quaternion (4); left/right fingertip contact (2); the connector-to-seated-target vector (3); connector position (3) and quaternion (4); socket position (3); seated-target position (3); and the insertion-gate and axial/lateral seating-error scalars (3).

Action and control. The insertion action is a 6-DoF Cartesian end-effector delta resolved to joint targets by a differential-IK controller, plus one gripper command. Actions are clipped to $[-1, 1]$ and scaled by 0.01 in RL setting. Simulation runs at $\Delta t = 1/120$ s with decimation 2, giving a 60 Hz policy rate; episodes last 8 s.

Reward. The insertion reward is a shaped sum of reach (15), grasp (25), bring (40), two-point YZ alignment (20), a roll-gated insertion-depth high-water ratchet (6000), socket/roll/EE-orientation

terms (10 each), and a sparse seated bonus (1000), against action and table/gripper penalties. PPO and SACfD use this identical reward; PPO applies empirical observation normalization, while SACfD normalizes the state with a running standard scaler. Insertion results across connectors are in Table 1.

Networks. PPO [42] uses an actor–critic MLP with hidden sizes $[256, 128, 64]$ and ELU activations for both actor and critic. SACfD [43] uses a tanh-squashed Gaussian actor ($[256, 256]$, ELU) with twin Q-critics ($[256, 256]$, ELU). PPO hyperparameters are listed in Table 7 and SACfD in Table 8.

Table 7: State-based PPO hyperparameters. Vision PPO (Sec. E.2) inherits these except where noted. `num_envs` is set at launch with no committed default.

Parameter	Value
Actor / critic MLP	$[256, 128, 64]$, ELU
Clip range ϵ	0.2
Entropy coef.	0.01
Value-loss coef.	1.0
GAE λ	0.95
Discount γ	0.99
Learning rate	5×10^{-4}, adaptive (KL^*=0.01)
Steps per env (rollout)	128
Minibatches / epochs	8 / 5
Max grad norm	1.0
Max iterations	20000
Obs. normalization	empirical (on)

Table 8: SACfD hyperparameters on connector insertion tasks.

Parameter	Value
Actor / critic MLP	$[256, 256]$, ELU
Discount γ	0.99
Target smoothing τ	0.005
Actor / critic LR	3×10^{-4} / 3×10^{-4}
Batch size	256
Gradient steps / env step	1
Replay capacity	$8000 \times N_{\text{env}}$
Warmup (learning starts)	5000
Entropy temp. α	learned, target -7, α_0=0.1
Reward shaping scale	0.01
State preproc.	RunningStandardScaler
Total timesteps	10^6
Demonstration augmentation (SACfD)	
Demo set	$\sim$150k transitions (scripted)
Demo replay ratio	0.5
BC weight (anneal)	$2.5 \rightarrow 0.5$ over $[80\text{k}, 250\text{k}]$ steps

E.2 Vision PPO

Design rationale. Training a visual policy from scratch with online RL is sample-inefficient: the encoder must learn useful features purely from the RL reward, which is slow and wasteful in interaction. We therefore decouple representation learning from control. The visual encoder is first supervised by behavior cloning on the corresponding state policy's rollouts, and only then refined with online PPO, which substantially speeds convergence. For the encoder we use a pretrained DINOv2 ViT backbone [55, 56] with a learned attention-pooling head rather than a CNN trained from scratch, since a frozen self-supervised backbone provides transferable features and keeps the online

phase cheap; the attention pool lets the policy aggregate task-relevant regions across the camera views.

Architecture. Vision PPO replaces the privileged actor input with two 224×224 RGB streams, a wrist camera and a fixed third-person side camera, plus a 14-d proprioceptive vector of arm joint positions and velocities together with the gripper position and velocity. It is an *asymmetric* actor–critic [57]: the actor sees images and proprioception, while the critic retains the full 42-d privileged state. The visual actor is a DINOv2 ViT-S/14 backbone with a cross-attention pooling head (8 learned queries per camera, 16 total; query dimension 64, hidden 512) that produces the 7-d action mean. The encoder is behavior-cloned from the state policy's rollouts before PPO; during PPO the backbone and attention pool are frozen, and only the final projection head, the action $\log \sigma$, and the critic are trained. The critic is warm-started from the converged State-PPO critic and remains trainable.

Hyperparameters. Vision PPO inherits all PPO settings in Table 7 except for a tighter trust region (clip 0.1, target-KL 0.005), no entropy bonus, a down-weighted value loss (0.3), a lower learning rate (1×10^{-4}), a small initial action noise ($\sigma = 0.05$), 64 parallel environments, and 600 fine-tuning iterations from the behavior-cloned initialization, with observation normalization disabled. Training was run on an NVIDIA RTX PRO 4500 Blackwell GPU. Per-connector results are in Table 1.

E.3 RL Rollout Demonstrations

See Figure 9.

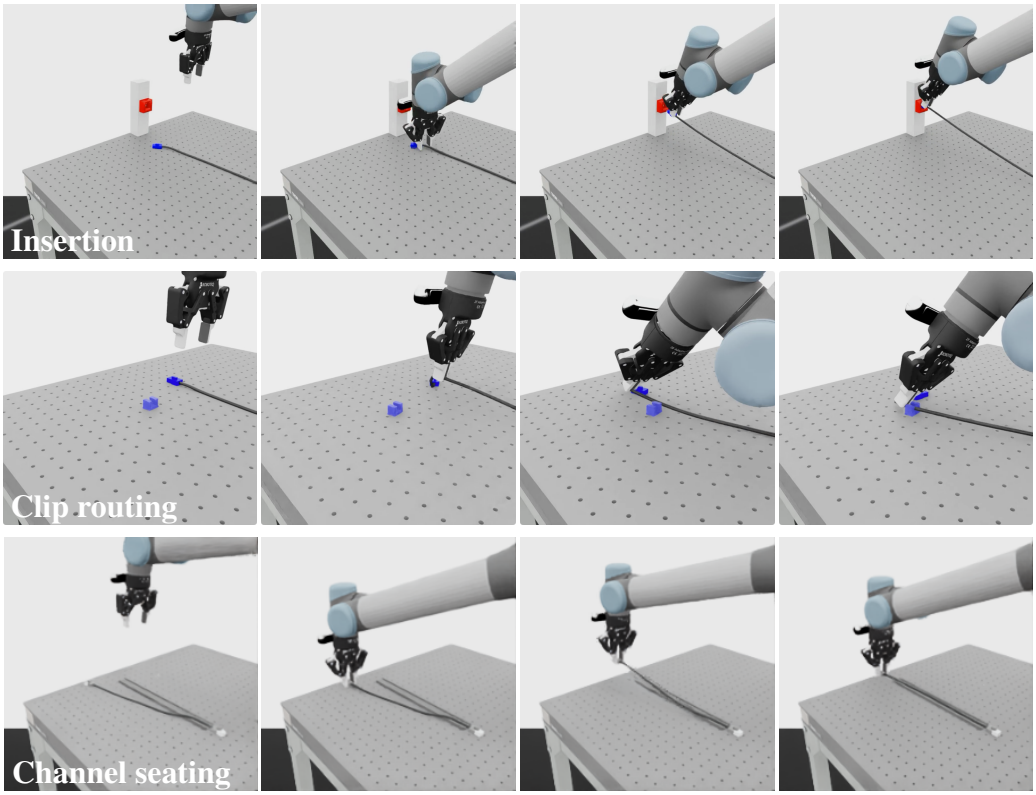


Figure 9: **RL rollout demonstrations.** Each row shows a trained State PPO policy rolled out on one task family across representative phases. The clip-routing row additionally illustrates an error and subsequent recovery.

F Connector-Insertion IL/VLA Experimental Details

F.1 Dataset

All four IL/VLA policies are trained on a shared dataset of 1,000 scripted connector-insertion trajectories collected in the same Isaac Lab task used for evaluation. Each trajectory contains two RGB image streams, one from a third-person workspace camera and one from a wrist-mounted camera, together with UR5 joint positions, gripper state, and joint-space actions. The action representation contains six arm commands and one gripper command. The shared dataset and action/state encoding allow all policies to be evaluated under the same observation and control interface.

F.2 Policy Training Configurations

See Table 9.

Table 9: Training configurations for the IL/VLA connector-insertion baselines. All policies use the same dataset and action/state encoding.

Policy	Model / conditioning	Action prediction	Training
ACT	ResNet-18 visual encoder with transformer action-chunking policy	Chunk size 100; 50 executed actions per inference call	100k gradient steps; batch size 64; learning rate 10^{-5}
Diffusion Policy	ResNet-18 visual encoders with a temporal U-Net action denoiser	Horizon 16; 2 observation steps; 8 action steps	200k gradient steps; batch size 64; learning rate 10^{-4}
Diffusion Transformer VLA	Frozen CLIP ViT-B/16 image and text conditioning with a DiT-style action denoiser	Horizon 32; 24 action steps	300k gradient steps; batch size 32; learning rate 2×10^{-5}
$\pi_{0.5}$	Initialized from `lerobot/pi05_base`; vision tower frozen; action expert and language adapter trainable	25 action steps	50k gradient steps; batch size 32; learning rate 10^{-5}

F.3 Checkpoint Selection

Each policy was checkpointed periodically during training and evaluated using the original reach-success criterion. We retained the top three checkpoints for each policy according to this reach success, rather than selecting checkpoints using insert success after the final evaluation. This avoids choosing checkpoints based on the stricter test metric while still allowing us to report how the best training-time checkpoints behave under both reach and insert success criteria. Because checkpoints are selected by reach success on the evaluation rollouts themselves rather than on a held-out validation set, the reported figures are an optimistic, best-observed estimate under our training budget and should not be read as validation-based model selection; insert success is taken from the same checkpoint and is therefore not separately optimized.

F.4 Evaluation Protocol

For each retained checkpoint (See Table 10), we evaluate four random seeds with 16 rollouts per seed, for a total of 64 rollouts per checkpoint (see Table 11). Each rollout records both reach and insert success from the same trajectory, avoiding separate evaluation runs for the two criteria. Episodes use a fixed horizon of 800 simulator steps. We report mean and standard deviation across seeds.

Table 10: Available and retained checkpoints for the connector-insertion IL/VLA baselines. Retained checkpoints are the top three checkpoints per policy according to online reach success during training.

Policy	Number of available checkpoints	Retained checkpoints
ACT	20	25k, 40k, 100k
Diffusion Policy	20	140k, 190k, 200k
Diffusion Transformer VLA	23	110k, 180k, 200k
$\pi_{0.5}$	5	10k, 20k, 30k

Table 11: Connector-insertion results for the retained checkpoints selected by reach success. Each entry reports mean ± standard deviation over four seeds, with 16 rollouts per seed.

Policy	Checkpoint	$\mathrm{SR}_{\mathrm{reach}}$ (%)	$\mathrm{SR}_{\mathrm{insert}}$ (%)
ACT	25k	43.75 ± 11.69	3.13 ± 3.13
ACT	40k	54.69 ± 17.88	9.38 ± 6.99
ACT	100k	$\mathbf{60.94 \pm 11.16}$	$\mathbf{29.69 \pm 9.24}$
Diffusion Policy	140k	29.69 ± 18.42	17.19 ± 13.53
Diffusion Policy	190k	35.94 ± 9.24	17.19 ± 6.81
Diffusion Policy	200k	$\mathbf{37.50 \pm 9.88}$	$\mathbf{20.31 \pm 6.81}$
Diffusion Transformer VLA	110k	9.38 ± 6.99	3.13 ± 3.13
Diffusion Transformer VLA	180k	4.69 ± 5.18	1.56 ± 2.71
Diffusion Transformer VLA	200k	$\mathbf{10.94 \pm 2.71}$	$\mathbf{3.13 \pm 3.13}$
$\pi_{0.5}$	10k	0.00 ± 0.00	0.00 ± 0.00
$\pi_{0.5}$	20k	$\mathbf{6.25 \pm 4.42}$	$\mathbf{0.00 \pm 0.00}$
$\pi_{0.5}$	30k	1.56 ± 2.71	0.00 ± 0.00

G Clip Routing and Channel Seating Experimental Details

Datasets. ACT and DiT VLA use the same scripted demonstration dataset for each task: 1,600 trajectories for 1-clip routing and 1,000 trajectories for straight-channel seating.

Training. For both tasks, ACT is trained for 100k gradient steps with batch size 64 and learning rate 10^{-5}; DiT VLA is trained for 300k gradient steps with batch size 32 and learning rate 2×10^{-5}. All four training runs use seed 1000. ACT uses a ResNet-18 visual encoder, predicts 100-action chunks, and executes 50 actions per inference call. DiT VLA uses CLIP ViT-B/16 image and text conditioning, two observation steps, a 32-step prediction horizon, and 24 executed actions per inference call.

Checkpoint selection and evaluation. We evaluate the final training checkpoint for each task–policy pair, at 100k steps for ACT and 300k steps for DiT VLA. For ACT and DiT VLA, results are reported as mean ± standard deviation across three evaluation seeds.

H Real-World Validation

We evaluate ACT on a physical UR5 across connector insertion, 1-clip routing, and straight-channel seating. For each task, we compare four training-data combinations: real demonstrations (R), scripted simulation demonstrations (S), and simulation RL rollouts (P), combined as R, R+S, R+P, and R+S+P. Preliminary trials showed poor zero-shot performance for simulation-only policies, so we restrict the full hardware evaluation to data combinations containing real demonstrations. The comparison examines whether simulated trajectories supplement real demonstrations under shared task definitions and policy interfaces.

H.1 Setup

Physical setup. The platform uses a UR5 arm with a Robotiq 2F-85 gripper and two RealSense D405 cameras providing side and wrist RGB views. The fixtures include a 3D-printed Ethernet-style connector and socket, a single clip, and a straight channel. Connector insertion and clip routing use a 6 mm-diameter HDMI cable, while channel seating uses a 10 mm-diameter green cord with red and yellow flecks. A force–torque sensor is mounted between the arm flange and gripper, but its readings are not used by ACT. We reproduce its end-effector offset in simulation and match the task-board appearance and camera poses to the physical setup. ACT observes RGB images and robot proprioception, but not depth and force/torque inputs, and executes action commands at 30 Hz.

Task initialization. Hardware trials use narrower initialization ranges than simulation. For insertion, the wire-side connector is placed within a 3×3 cm region with varying tilt, while the socket

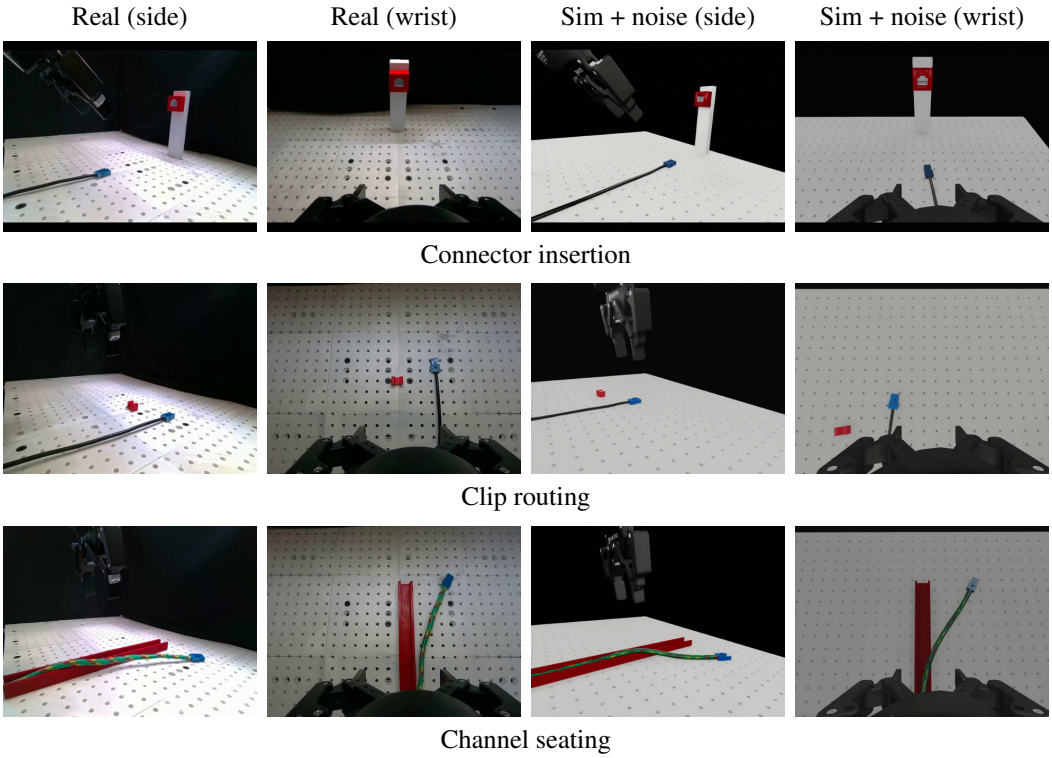


Figure 10: **Real vs. noise-augmented simulation observations across three tasks.** Rows show connector insertion, clip routing, and channel seating; columns show the side and wrist views from real demonstrations and scripted simulation trajectories. Brightness, contrast, and Gaussian-noise augmentation are applied only to the simulated frames using the training preset.

remains fixed. For routing, a single clip is placed on either side of the wire, 2–5 cm away. For seating, one wire endpoint is fixed, while the other is placed 2–5 cm from the channel in a direction perpendicular to its longitudinal axis.

Bridging the sim-to-real visual gap. We match the simulated side and wrist views to the physical cameras and augment simulation images during training. Brightness and contrast factors are sampled independently from $[0.85, 1.15]$, and Gaussian-noise standard deviation is sampled from $[0, 0.015]$ for pixels in $[0, 1]$. These augmentations apply only to simulation episodes; real images and evaluation inputs are not augmented. Figure 10 shows observations from both domains across the three tasks.

H.2 Protocol

Data sources. Real demonstrations are collected on the physical UR5, while scripted demonstrations and RL rollouts are generated in the matched simulation environments. All sources share the same RGB, joint-state, and action schema. Table 12 reports the dataset sizes and training/validation episode counts. We hold out every tenth episode within each source, giving approximately 10% validation episodes. The remaining episodes are used for training.

Training. All configurations train ACT for 80k gradient steps with batch size 16, learning rate 10^{-5}, and training seed 1000. ACT uses an ImageNet-initialized ResNet-18 visual backbone, predicts 100-action chunks, and executes 50 actions per inference call. Inputs comprise two RGB views and a seven-dimensional joint/gripper state; outputs are six arm-joint targets and one gripper target.

Table 12: Trajectory data for the three hardware tasks by source. Episodes and frames report total dataset sizes; train and validation columns report episode counts. All sources are recorded at 30 FPS under the shared schema.

Task	Source	Episodes	Frames	Train	Validation
Insertion	Real-world (R)	50	30,018	45	5
	Scripted (S)	200	121,890	180	20
	RL rollout (P)	200	27,394	180	20
1-clip routing	Real-world (R)	51	30,634	46	5
	Scripted (S)	200	125,304	180	20
	RL rollout (P)	200	53,226	180	20
Straight-channel seating	Real-world (R)	50	30,033	45	5
	Scripted (S)	200	114,954	180	20
	RL rollout (P)	200	56,511	180	20

Checkpoint selection. Checkpoints are saved every 5k steps and selected by the lowest masked L1 action-prediction loss over the predicted chunks in normalized action space. Selection uses only the five held-out real episodes for each task, including for mixed-source policies. In the order R, R+S, R+P, and R+S+P, selected checkpoints are $40/80/55/80$k for insertion, $55/80/80/70$k for routing, and $80/80/75/80$k for seating. Thus, configurations share the training budget and selection rule, but not the selected training step. Normalization statistics for the real-only datasets include the held-out episodes.

H.3 Results and Analysis

Representative failure modes are shown in Figure 11.

Connector insertion. Observed failures include unsuccessful pickup and reaching without seating. The gripper sometimes closes beside the connector or fails to secure it during lifting. After successful pickup, the connector may reach the socket but remain unseated because of its position or orientation within the grasp. A grasp sufficient for transport may therefore be unsuitable for insertion.

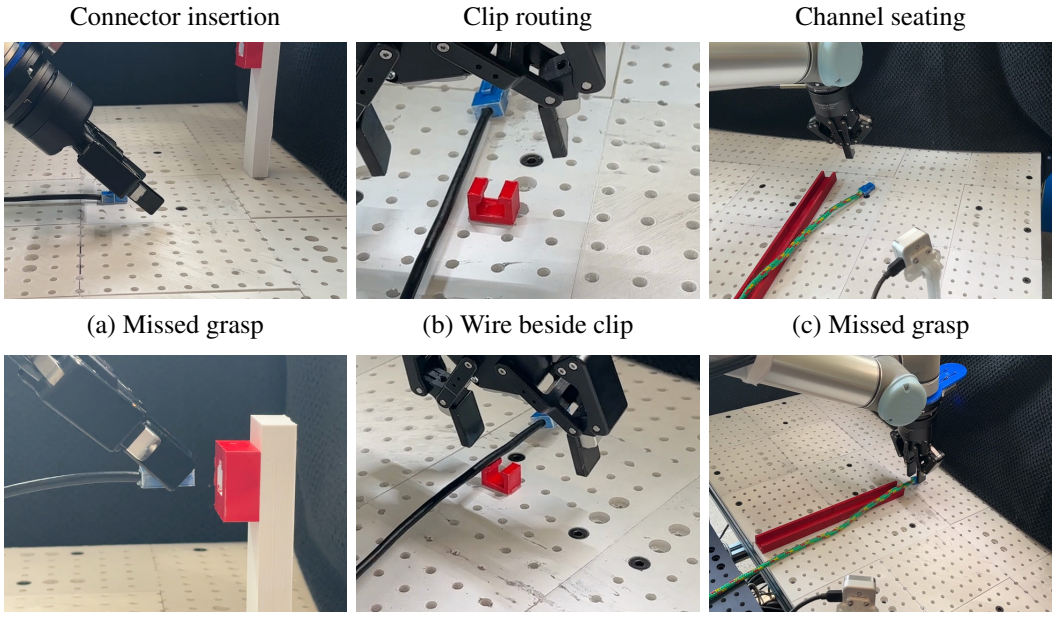
Clip routing. The wire may be released beside the clip opening or rest on the clip rim without entering the slot. In these cases, the gripper reaches the fixture, but the wire remains incorrectly positioned relative to the opening.

Channel seating. Missed grasps leave the free endpoint on the board. After successful pickup, moving the endpoint near the channel can still leave part of the wire outside it; successful seating requires placing the wire body within the channel.

The characteristics of the three trajectory sources are summarized in Table 13.

Table 13: Characteristics of the three trajectory sources used for hardware policy training.

Source	**Trajectory characteristics**
R	Human demonstrations contain pauses and repeated fine adjustments during alignment. Their limited diversity can restrict generalization.
S	Scripted trajectories use action noise during generation to broaden motion coverage. Execution is relatively slow and regular, providing varied examples of controlled task completion.
P	Task-optimized RL rollouts produce fast, aggressive motions. For this dataset collection, execution speed was reduced to 50% of the original policy speed.



(a) Missed grasp (b) Wire beside clip (c) Missed grasp

(d) Unseated connector (e) Wire on clip rim (f) Wire outside channel

Figure 11: **Representative hardware failures in insertion (left), clip routing (middle), and channel seating (right).** Panels show (a,c) missed grasps, (b) wire misplacement beside the clip, (d) reaching without connector seating, (e) wire resting on the clip rim, and (f) wire placement outside the channel.

Limitations. The hardware study covers three tasks on one UR5 with restricted initialization ranges and 20 trials per configuration. Indistinguishable difference in count of successful trials cannot lead to a definitive ranking of data mixtures. Source composition and dataset size vary together, so these comparisons measure data supplementation without isolating source diversity from data quantity.

I Teleoperation Data Collection

WireCraft teleoperation collects demonstrations by having a human operator drive the UR5 end-effector in Isaac Sim through a 6-DoF input device, with episodes recorded as supervised trajectories for IL and VLA training. Operator input is mapped to an end-effector delta pose plus a binary gripper command,

$$a_{\text{cart}} = [\Delta x, \Delta y, \Delta z, \Delta r_x, \Delta r_y, \Delta r_z, g],$$

which is resolved by differential IK into a UR5 joint-position target. Trajectories are stored under the same observation–action schema as scripted demonstrations, so the two sources can be mixed without modifying the policy interface. The key behavioral difference is that scripted policies produce smooth, phase-structured rollouts from privileged simulation state, whereas teleoperation produces visually conditioned rollouts that include pauses, corrections, and re-grasps useful for contact-rich DLO manipulation.

Control interfaces. We support three input modes, all sharing the same Cartesian-delta action interface: (i) a 3Dconnexion SpaceMouse with on-screen camera views (default); (ii) Meta Quest VR with retargeted wrist motion and a pinch-to-grip gesture, useful when fixed cameras provide poor depth cues; and (iii) a combined mode using the SpaceMouse for control and the VR headset for visualization. All modes record only simulator observations and IK-resolved joint targets; raw button events and hand-tracking signals are not stored.

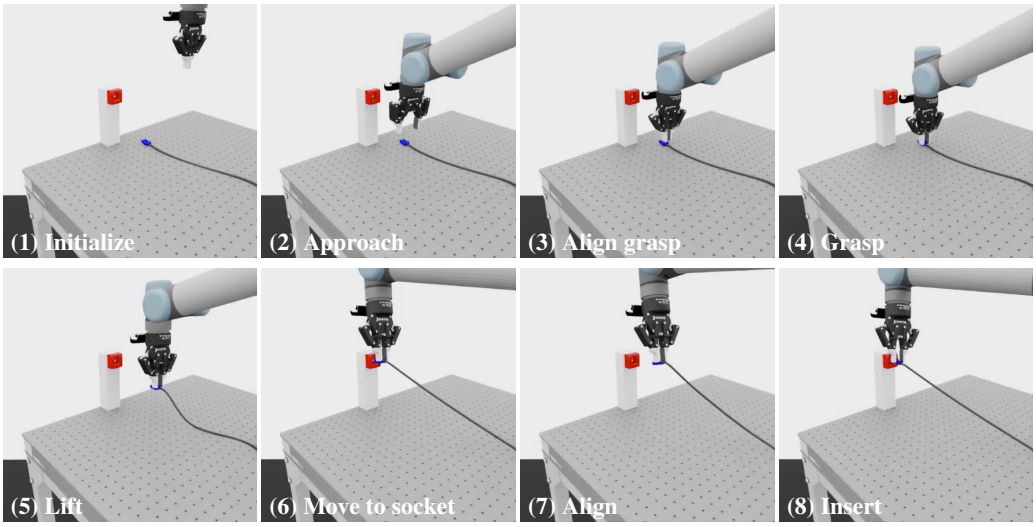


Figure 12: **A teleoperated connector-insertion trajectory in simulation.** A human operator drives the UR5 end-effector through eight representative stages: (1) initialization with the connector resting on the table; (2) approaching the connector; (3) aligning the gripper above it; (4) grasping the connector, typically after several attempts, since the operator must re-position and re-close the gripper to secure a firm grasp; (5) lifting the connector clear of the surface; (6) moving toward the socket while pulling the trailing wire taut to reduce its disturbance during insertion; (7) aligning the connector with the socket opening; and (8) seating it, often requiring repeated insertion attempts to correct small misalignments. Because the operator has no haptic feedback and only limited visual cues for fine alignment, both grasping and insertion rely on trial-and-error retries, making teleoperated episodes substantially longer than scripted or policy rollouts.

Recorded schema. The teleoperation pipeline follows the shared WireCraft trajectory schema, storing synchronized third-person and wrist RGB streams (`observation.images.cam_side2`, `observation.images.cam_color`), UR5 joint and gripper state (`observation.state`), end-effector pose (`observation.cartesian_state`), the applied joint-position target (`action`), `timestamp`, `episode_index`, and a `task` label. The recorded `action` is the IK-resolved joint target rather than the raw operator Cartesian delta, allowing teleoperated trajectories to use the same action representation as scripted and policy-generated data. In this paper, we release the teleoperation interface and recording pipeline, but do not include simulated teleoperation trajectories in the reported training datasets. Systematic teleoperation-data collection, filtering, and use for policy training are left to future work.