Few-Shot Learning of Tool-Use Skills with Proximity and Tactile Sensing

Author Names Omitted for Anonymous Review.

***Abstract*— Teaching robots to use tools is challenging due to simultaneous robot–tool and tool–environment contacts. Tactile and proximity sensors are crucial for detecting these interactions; however, learning tool manipulation with these sensors remains difficult given limited real-world data and a large sim-to-real gap. We propose a few-shot tool-use skill transfer framework that leverages multimodal sensing by pre-training a base policy in simulation to capture contact states common to tool-use skills, and fine-tuning it with a small number of human demonstrations in the real-world target domain. We validate our approach on surface-following tasks with diverse tools using the Franka Emika arm, demonstrating improved contact recognition and rapid skill acquisition.**

I. INTRODUCTION

Tools extend the ability of robots to manipulate objects and the environment [1–6]. This study focuses on manipulating grasped tools, rather than fixing tools to the end-effector as in most prior work [2–5]. The ability to manipulate grasped tools allows robots to use tools designed for humans and to seamlessly swap between them for different tasks.

Manipulation of grasped tools involves two types of contact: intrinsic contact (between robot fingers and the tool) and extrinsic contact (between the tool and the environment)[7]. While tactile sensors determine contact forces, contact events, local geometries, and material properties when they are in contact with the object or the environment [8], proximity sensors identify local geometries without contact [9]. Despite these capabilities, there are currently no sensors capable of directly measuring extrinsic contact. The challenge is to use multiple indirect sensor sources to identify interactions between tools and environments.

Furthermore, accurate models and simulations of deformable tools are often unavailable. Several recent works control extrinsic contacts between a tool [10] or object [11, 12] and the environment, but they are limited to rigid objects on flat surfaces. Learning from Demonstration (LfD) [13] provides a viable approach, however, collecting sufficient demonstrations is time-consuming and limited by tool and environment variations. Van der Merwe *et al.* and Zhang *et al.* propose learning contact states from interaction data [14, 15], enabling manipulation of deformable tools on non-flat surfaces, however, these approaches, are limited to the manipulation of the tools used during training.

We, therefore, propose a multimodal few-shot tool-use skill transfer framework that pre-trains a base policy in simulation and fine-tunes it with a few human demonstrations. The key idea is to leverage inexpensive simulated data while bridging the gap to the real target domain with human demonstrations. While the proposed framework is a general multimodal approach, we focus on the combined use of tactile and proximity sensors for tool-use skill acquisition.

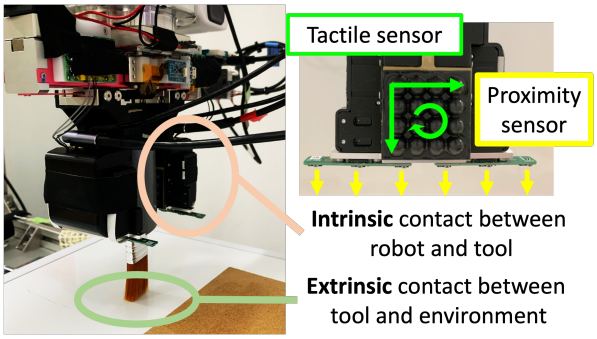


Fig. 1: Tool manipulation using tactile and proximity sensors, involving two points of contact.

In summary, our contributions are:

- addressing the challenge of learning deformable tool-use skills involving intrinsic and extrinsic contact,
- proposing a data-efficient approach combining tactile and proximity sensing with pre-training in simulation and few-shot fine-tuning with human demonstrations,
- validating the proposed method on surface-following tasks with diverse tools on a physical robot, without fixing the tool to the end-effector.

II. PROBLEM FORMULATION

We address the problem of learning tool-use skills involving two locations of contact: one between the robot and the tool, and the other between the tool and the environment, as illustrated in Fig. 1. We consider a scenario where the physical and geometric properties of the tool and surface are unknown a priori. Additionally, as the desirable contact force between the tool and the environment depends on the task, the robot needs to learn this force from demonstrations of the target task. In this study, we focus on surface-following tasks such as painting with a brush and sweeping with a broom.

III. METHOD

To enable learning of new tool-use skills from a limited number of demonstrations, we propose a few-shot tool-use skill transfer framework. This framework involves pre-training the base policy using primitive motions in simulation and fine-tuning it with human demonstrations of downstream tasks using the target tool in the real world to bridge the gap.

A. Pre-training base policy

We first train the base policy π_b by leveraging a large amount of data $D_{prim} = \{\tau_1, \cdots \tau_N\}$, where $\tau =$

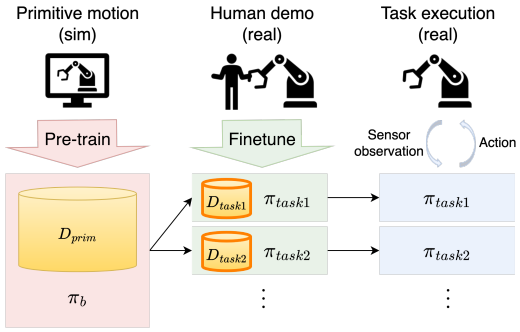


Fig. 2: Few-shot tool-use skill transfer framework.

$\{(x_0, u_0), (x_1, u_1), \ldots, (x_T, u_T)\}$ collected through primitive motions in simulation. Here, N denotes the number of trajectories, and T represents the number of timesteps in each trajectory τ. By performing this set of primitive motions repeatedly in simple simulated environments with varying conditions, we expect the base policy π_b to recognise contact states between the tool and the environment from multi-modal sensor observations and to capture motions common in tool-use skills, transferable across tasks. We highlight that every new task with a new tool, environment, or task goal shares this single base policy, as illustrated in Fig. 2.

For this, we adopt the Sequence-to-Sequence (Seq2Seq) model [16–18]. The model employs an encoder-decoder structure using Long Short-Term Memory (LSTM), as illustrated in Fig. 3. This encoder-decoder structure offers key advantages for our framework. The encoder acts as a feature extraction module, capturing latent representations of contact dynamics, decoupled from motion generation, enabling the reuse of the feature extraction module across tasks. Then, we can fine-tune the decoder, a motion generation module, for task-specific motion generation. To achieve this, we train the model on primitive data by minimising a combined state and action prediction loss (see Appendix A for details).

B. Few-shot fine-tuning using demonstration

Subsequently, we fine-tune the pre-trained base policy π_b to obtain the fine-tuned policy π_{task} using a small amount of demonstration data D_{task} specific to each new downstream task. This step allows adaptation to differences in tool properties, environments, percepts, and task goals, bridging the gap from pre-trained to the target downstream task domain.

To prevent overfitting with limited demonstrations, we fine-tune only the fully connected (FC) layer of the decoder, keeping all other layers in both the encoder and decoder frozen. Therefore, while the primary role of the decoder during pre-training is motion generation, we expect the updated weights of ω_{dec} to adapt not only to differences in task goals and motions but also to domain differences, such as tool properties and other gaps between the pre-trained and target task domains. We achieve this adaptation by incorporating both state prediction and action prediction

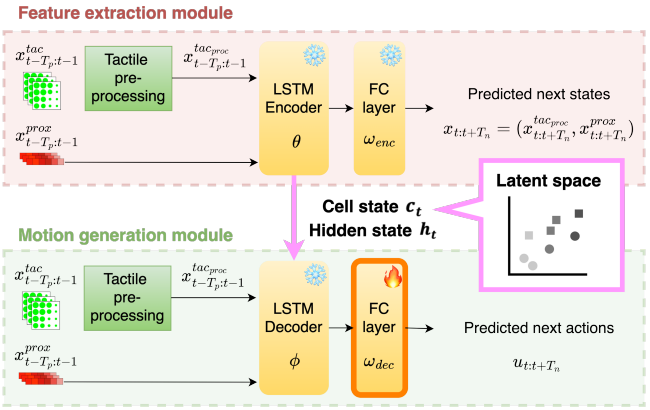


Fig. 3: Seq2Seq architecture of the tool-use skill policy. The feature extraction module captures the contact relationship between the robot, tool, and environment by predicting next states. The motion generation module predicts the next desired actions to perform the task.

losses, as shown in LfD loss

$$\hat{\omega_{dec}} = \arg\min_{\omega_{dec}} L\left(x_{t:t+T_n}, u_{t:t+T_n}\right) \quad (1)$$
$$= E_{MSE}\left(\hat{x}_{t:t+T_n}, x^*_{t:t+T_n}\right) + \beta E_{MSE}\left(\hat{u}_{t:t+T_n}, u^*_{t:t+T_n}\right).$$

C. Task execution

At test time, the robot predicts the next desired actions $\hat{u}_{t:t+T_n}$ based on past sensor observations $x_{t-T_p:t-1}$ using the fine-tuned policy π_{task} to perform the downstream task. At each timestep, the robot executes only the action for the next step while predicting actions for the next T_n steps.

IV. EXPERIMENTAL SETUP

We conducted our experimental evaluation on surface-following tasks using tools with varying geometric and physical properties, in simulation and on a physical setup.

Fig. 4 depicts the experimental setup and sensor configurations. Inside each fingertip, we attach tactile sensors [19] with 4 by 4 elastic hemispheres placed on 12 by 12 pressure sensor nodes, providing the 3-axis force estimates at each hemisphere by measuring its distortion. In simulation, we use 3-axis force sensors arranged in a corresponding 4 by 4 configuration as an approximate match. For both simulated and real tactile observations, we pre-process 3-axis forces from each fingertip, denoted as x^{tac}, into shear forces $x^{shear} \in \mathbb{R}^2$ and translational and rotational slip $x^{slip} \in \mathbb{R}^3$, following the theory of translational and rotational slip [19]. Additionally, we place six Time Of Flight proximity sensors on the right fingertip, oriented to look down at the environment. We provide details of the tasks, system design, data collection, and model training in Appendix B.

V. RESULTS AND DISCUSSION

We evaluate the ability of the proposed few-shot tool-use skill transfer framework to adapt the motion for new tools and target tasks with only a few demonstrations. We provide details of the baselines and evaluation metrics in Appendix C.

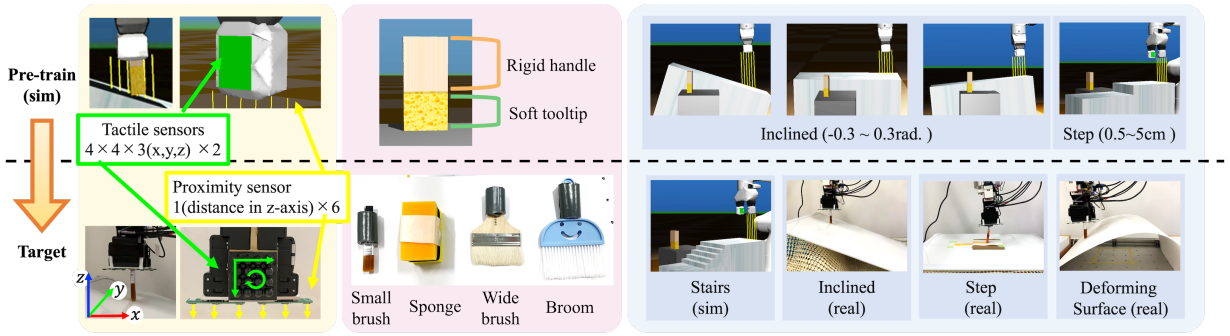


Fig. 4: Experimental setup.

A. *Does the tool-use skill policy, fine-tuned with demonstrations, improve the performance of downstream tasks?*

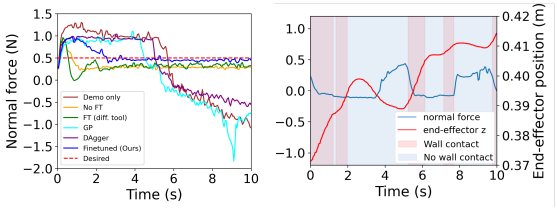
First, we evaluated our approach in two test cases using the simulated environments "Inclined" and "Stairs". For the inclined surface, the task involved wiping an inclined surface with a desired force of 0.5N, which differed from the 0.3N used during pre-training. In the "Stairs" environment, the task was to wipe a surface with stairs instead of a single step seen in the pre-training phase.

In both environments, "Finetuned" achieved the highest task performance compared to the baselines ("Demo only" and "No FT") as well as the alternative methods "GP" and "DAgger," as depicted in Table I. Fig. 5a illustrates that "Demo only" failed to achieve the desired force due to insufficient demonstrations. In contrast, "Finetuned" successfully achieved the new desired contact force, different from the desired force used in pre-training. Fig. 5b shows the end-effector position in the z-direction as well as the contact force in the normal direction while wiping stairs using the fine-tuned policy. The red shading indicates the contact of the tool tip with the wall, and the blue indicates the flat region (*i.e.,* no contact of the tool tip with the wall). We observed that the fine-tuned policy detects the contact of the tool tip with the wall, leading to successful lifting of the tool, and returns to the desired contact force (*i.e.,* 0.3N in this case) when reaching the flat region. We also evaluated the generalisability of the policy fine-tuned on a different tool instance with a longer handle (5cm vs. 3cm) and softer tip (stiffness 10 vs. 30) (FT (diff. tool) in Table I); while this policy outperformed the baselines, the one fine-tuned on the target tool achieved the lowest error, highlighting the importance of task-specific fine-tuning.

Second, we evaluated our approach in two real-world environments: "Inclined" and "Step", using four different tools. In both environments, the robot needs to adapt its motions to real tools with various properties that are inherently different from those used in simulation, while also accounting for the sim2real gap in sensing. Additionally, the robot needs to achieve a new target contact force required for each task, as demonstrated by a human. Table IIa and Table IIb display the Root Mean Squared Error (RMSE) of the inclination and the contact force, respectively, for four downstream tasks. In all cases, "Finetuned" achieved a significant reduction in RMSE

Env.	Evaluation Criteria	Demo only	No FT	FT (diff. tool)	GP	DAgger	Finetuned
Inclined	RMSE of slope (rad) ↓	0.073 ± 0.002	0.041 ± 0.006	0.038 ± 0.003	0.095 ± 0.007	0.072 ± 0.005	0.040 ± 0.005
	RMSE of force (N) ↓	0.68 ± 0.06	0.22 ± 0.01	0.17 ± 0.02	0.61 ± 0.05	0.62 ± 0.07	**0.11** ± 0.02
Stairs	Wiped area (%) ↑	45.0 ± 3.50	49.7 ± 2.31	53.1 ± 2.98	43.3 ± 4.01	51.1 ± 3.80	**59.2** ± 4.31

TABLE I: Comparison of RMSE when transferring the pre-trained tool-use skill to a new task (*i.e.,* with a new target force) and to a more complex environment (*i.e.,* stairs).



(a) Inclined (b) Stairs

Fig. 5: End-effector motion and contact force profile.

compared to the baselines. We report task completion results for each task in Appendix C.3. Furthermore, for the task of painting a surface with a step, "Finetuned" notably increased the wiped area, as evidenced by the results in Table IIc.

B. *How do proximity and tactile sensing contribute to tool manipulation?*

We investigated the contributions of each sensor modality and configuration to surface-following tasks, as summarized in Fig. 6 and Fig. 7.

In simulation, all modalities perform similarly in following the desired inclination. However, proximity sensing alone struggles to achieve the desired force on inclined surfaces and to detect tool-tip contact on stairs, where tactile information is essential. Additionally, the sensor configurations capturing only intrinsic contact, as shown in Fig. 7(a)(1)-(3) performed poorly across tasks, as indicated in Fig. 7(b)–(d). Combining an array of proximity sensors and shear force from tactile sensors which allows to capture both intrinsic and extrinsic contact (Fig. 7(a)(4)) resulted in the lowest RMSE of contact force on inclined surfaces and the largest wiped area on stairs. These results underscore the importance of our sensor configurations and learning framework combining both sensing to implicitly capture both intrinsic and extrinsic contact for achieving the desired tool-environment contact.

Tool	Demo only		No FT		Fintuned	
	x	σ	x	σ	x	σ
Small brush	0.41	0.16	0.25	0.07	**0.17**	0.02
Sponge	0.30	0.12	0.29	0.01	**0.17**	0.03
Wide brush	0.18	0.04	0.37	0.12	**0.10**	0.02
Broom	0.84	0.33	0.66	0.37	**0.12**	0.05

(a) RMSE of the inclination followed by the end-effector (↓).

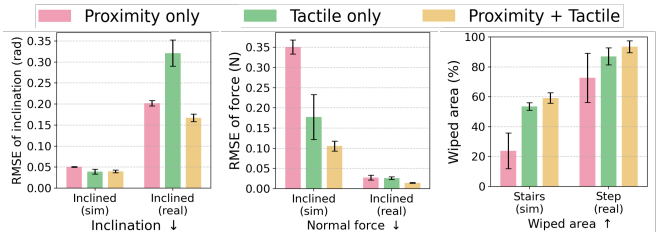
Tool	Demo only		No FT		Fintuned	
	x	σ	x	σ	x	σ
Small brush	0.041	0.006	0.031	0.011	**0.014**	0.002
Sponge	0.065	0.034	0.051	0.015	**0.017**	0.004
Wide brush	0.017	0.008	0.038	0.007	**0.009**	0.004
Broom	0.034	0.022	0.057	0.047	**0.020**	0.009

(b) RMSE of the normal contact force (↓).

Env.	Demo only		No FT		Fintuned	
	x	σ	x	σ	x	σ
Step	72.74	16.45	87.07	5.61	**93.54**	4.01

(c) Wiped area (↑).

TABLE II: Comparison of task performance using different tools on "Inclined" and "Steps" surfaces on a physical setup.



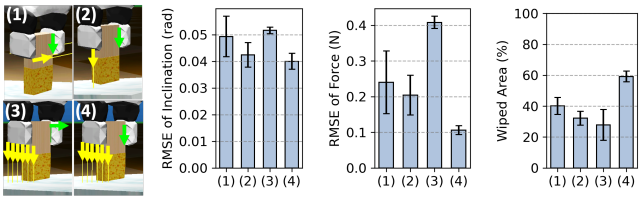
(a) Inclination ↓ (b) Normal force F_z ↓ (c) Wiped area ↑

Fig. 6: Contribution of each modality.

In real-world environments, combining proximity and tactile sensing significantly improved task performance in both inclined and step environments compared to using either sensor alone. While tactile sensing outperformed proximity sensing in simulation, their performances were closer in real-world scenarios, suggesting a larger sim-to-real gap and higher sensor noise for tactile sensors in a physical setup.

C. How adaptive and robust are the learnt tool-use skills?

Finally, we qualitatively evaluated the online adaptability of the learnt tool-use policy to changing environments. We apply t-SNE to the LSTM encoder's cell state $\mathbf{c_t} \in \mathbb{R}^{100}$ at each timestep t for both training and testing data, with the fixed hyperparameters: perplexity = 30, learning rate = 200, number of iterations = 1000, and using Euclidean distance as the metric. Fig. 8(1) and Fig. 8(3) show the t-SNE-transformed cell states, with grey dashed lines illustrating the transition of cell states in response to changing surface inclination from -0.25 radians (down) to 0.0 radians (flat) and to 0.25 radians (up) in simulation and real settings, respectively. Similarly, the model updated cell states when detecting tool-tip contact with walls in the stairs environment in simulation (see Fig. 8(2)) and in the step environment in physical setups (see Fig. 8(4)). This analysis, which shows that our policy can quickly update the contact state encoded in the model's cell states based on sensing feedback, supports



(a) Configurations. (b) Inclination ↓ (c) Normal force F_z ↓ (d) Wiped area↑

Fig. 7: Comparison of different sensor configurations. : (1) proximity sensors inside the finger, (2) a single proximity sensor, (3) normal force only, (4) proposed shear and slip force with an array of proximity sensors.

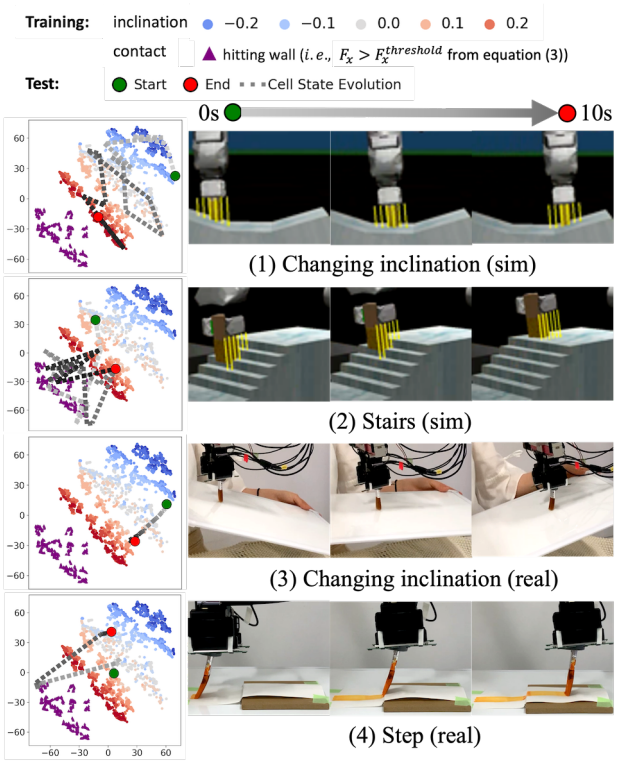


Fig. 8: The t-SNE visualisation of cell state evolution in the feature extraction module over time and the adaptation of robot motion on the fly in simulated and physical setups.

our quantitative results on successful adaptation when hitting the wall in the "Stairs" and "Step" environments in Table I and Table II. See Appendix C.4 for additional results on online adaptation to surface changes.

VI. CONCLUSIONS

We present a few-shot tool-use transfer framework that leverages inexpensive and diverse simulation data to pre-train a base policy, followed by fine-tuning with few human demonstrations to bridge the domain gap. Experiments show that our approach enables the manipulation of various tools with limited demonstrations, leveraging pre-trained representations of high-dimensional tactile and proximity sensing. Future work includes pre-training with diverse tools in simulation, incorporating modalities such as vision and language, and applying them to large-scale foundation models.

REFERENCES

[1] M. Qin et al., "Robot tool use: A survey," *Front. Robot. AI.*, vol. 9, Jan. 2023.

[2] M. Y. Aoyama et al., "Few-shot learning of force-based motions from demonstration through pre-training of haptic representation," in *ICRA*[1], 2024.

[3] N. Saito et al., "How to select and use tools? : Active perception of target objects using multimodal deep learning," *RA-L*[2], vol. 6, no. 2, pp. 2517–2524, 2021.

[4] P. Sundaresan et al., "Learning visuo-haptic skewering strategies for robot-assisted feeding," in *6th CoRL*[3], 2022.

[5] Y. Wi et al., "VIRDO++: Real-world, visuo-tactile dynamics and perception of deformable objects," in *6th CoRL*[3], 2022.

[6] K. Yamane et al., "Soft and rigid object grasping with cross-structure hand using bilateral control-based imitation learning," *RA-L*[2], 2023.

[7] D. Ma et al., "Extrinsic contact sensing with relative-motion tracking from distributed tactile measurements," *ICRA*[1], 2021.

[8] Q. Li et al., "A review of tactile information: Perception and action through touch," *T-RO*[??], vol. 36, no. 6, pp. 1619–1634, 2020.

[9] S. E. Navarro et al., "Proximity perception in human-centered robotics: A survey on sensing systems and applications," *T-RO*[??], vol. 38, no. 3, pp. 1599–1620, 2022.

[10] Y. Shirai et al., "Tactile tool manipulation," in *ICRA*[1], 2023.

[11] S. Kim et al., "Simultaneous tactile estimation and control of extrinsic contact," in *ICRA*[1], 2023.

[12] A. Bronars et al., "Texterity: Tactile extrinsic dexterity," in *ICRA*[1], 2024.

[13] A. Billard et al., "Robot programming by demonstration," *Springer Handbook of Robotics*, B. Siciliano et al., Eds., pp. 1371–1394, 2008.

[14] M. V. der Merwe et al., "Learning the dynamics of compliant tool-environment interaction for visuo-tactile contact servoing," in *6th CoRL*[3], 2022.

[15] H. Zhang et al., "Interaction control for tool manipulation on deformable objects using tactile feedback," *RA-L*[2], vol. 8, no. 5, pp. 2700–2707, 2023.

[16] I. Sutskever et al., "Sequence to sequence learning with neural networks," *NeurIPS*[4], vol. 27, 2014.

[17] K. Cho et al., "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[18] K. Kutsuzawa et al., "Sequence-to-sequence model for trajectory planning of nonprehensile manipulation including contact model," *RA-L*[2], vol. 3, no. 4, pp. 3606–3613, 2018.

[19] T. Narita et al., "Theoretical derivation and realization of adaptive grasping based on rotational incipient slip detection," in *ICRA*[1], 2020.

[20] E. Todorov et al., "Mujoco: A physics engine for model-based control," in *IROS*[??], 2012.

[21] K. Zakka et al., *MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo*, 2022.

[22] N. Srivastava et al., "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[23] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[24] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *NeurIPS*[4], 1988.

[25] M. Al-Shedivat et al., "Learning scalable deep kernels with recurrent structure," *J. Mach. Learn. Res.*, vol. 18, no. 82, pp. 1–37, 2017.

[26] S. Ross et al., "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. of the 14th Int. Conf. Artif. Intell. Stat.*, 2011.

APPENDIX

A. Method

1) Seq2seq model

The encoder serves as a feature extraction module. This module encodes multi-modal sensor observations x from the past T_p timesteps into latent representations consisting of cell and hidden states by predicting the next T_n states $\hat{x}_{t:t+T_n}$. In our case, with tactile and proximity sensors, the sensor observations x include $x_{tac}^{t-T_p}$ and $x_{prox}^{t-T_p}$. The decoder functions as the motion generation module, predicting the desired actions $u_{t:t+T_n}^{prim}$ in specified in primitive motions for the next T_n timesteps. We condition this motion generation on the encoded contact states, which are passed from the encoder's cell and hidden states to the decoder.

2) Pre-training loss

We train the model on primitive data by minimising the pre-training loss

$$\hat{\theta}, \hat{\phi}, \hat{\omega_{enc}}, \hat{\omega_{dec}} = \underset{\theta,\phi,\omega_{enc},\omega_{dec}}{\arg\min} \; L\left(x_{t:t+T_n}, u_{t:t+T_n}\right) \tag{2}$$
$$= E_{MSE}\left(\hat{x}_{t:t+T_n}, x_{t:t+T_n}^{obs}\right) + \beta E_{MSE}\left(\hat{u}_{t:t+T_n}, u_{t:t+T_n}^{prim}\right)$$

where θ and ω_{enc} denote the parameters of the LSTM layer and fully connected layer in the encoder, respectively, and ϕ and ω_{dec} represent the parameters of the LSTM layer and fully connected layer in the decoder of the seq2seq model, respectively, as shown in Fig. 3. The first term of the loss function is the state prediction loss, which computes the Mean Squared Error E_{MSE} between the observed states $x_{t:t+T_n}^{obs}$ and the predicted states $\hat{x}_{t:t+T_n}$ for the next T_n time steps to train the feature extraction module. The second term of the loss function is the action prediction loss, which computes the Mean Squared Error E_{MSE} between the actions $u_{t:t+T_n}^{prim}$ observed during the execution of primitive motions and the predicted actions $\hat{u}_{t:t+T_n}$ for the next T_n time steps to train the motion generation module. β is the weight coefficient that balances the two loss terms.

B. Experimental setup

1) Tool-use tasks

In simulation, we designed a brush with a 3cm rigid handle and a soft tooltip (2×3×3 capsules, 2.5cm spacing, stiffness 30 using MuJoCo's composite API) for pre-training. We then validated our proposed framework in real environments using four different tools: a small brush, a wide brush, a sponge, and a broom on a physical setup as depicted in Fig. 4. Each tool had a soft material around the handle to ensure a stable grasp. For the surfaces, we prepared two basic environments: "Inclined," which is a flat surface inclined at $\psi = [-0.3, 0.3]$ radians, and "Step" which has a step with a height of $h = [0.5, 5.0]$ cm in simulation. We tested in the 'Inclined' and 'Step' environments in the real world. Additionally, we tested more complex environments, including 'Stairs' with multiple consecutive steps in simulation and 'Deforming Surface' in the real world.

2) System design and robot setup

We utilise a 7 Degrees of Freedom (DoF) Franka Emika Panda robotic arm with a position controlled 2-finger parallel gripper attached to its end-effector for both simulation and real robot experiments. For the simulation setup, we utilise the Mujoco physics engine [20, 21], configured to replicate the physical robot setup. At the start of each trial, the robot grasps the tool handle near its centre, with a position

[1]IEEE International Conference on Robotics and Automation.

[2]IEEE Robotics and Automation Letters.

[3]Annual Conference on Robot Learning.

[4]Advances in neural information processing systems.

shift sampled from $\mathcal{N}(0,1)$ cm, by closing the gripper until the normal force exceeds 5N, and then maintains a fixed gripper width throughout the trial. The base and fine-tuned policies output end-effector velocities in the x and z directions, as shown in Eq. (3), which we convert to end-effector positions and then map to joint configurations using inverse kinematics to control the robot. We provide demonstrations by kinaesthetically moving the robot arm in gravity compensation mode.

3) Tactile and proximity sensors

Inside each fingertip, we attach tactile sensors [19] with 4 by 4 elastic hemispheres placed on 12 by 12 pressure sensor nodes. The tactile sensors estimate the 3-axis force at each hemisphere by measuring its distortion. In simulation, we use 3-axis force sensors arranged in a corresponding 4 by 4 configuration as an approximate match. For both simulated and real tactile observations, we pre-process 3-axis forces from each fingertip, denoted as x^{tac}, into shear forces $x^{shear} \in \mathbb{R}^2$ and translational and rotational slip $x^{slip} \in \mathbb{R}^3$, following the theory of translational and rotational slip [19]. This tactile representation, $x^{tac_{proc}} = \left(x^{shear}, x^{slip}\right)$, provides a compact and meaningful low-dimensional tactile feature that reduces the sim-to-real gap and enhances robustness against variations in raw sensor values caused by slight changes in grasp location or sensor noise. Additionally, we place six Time Of Flight (ToF) proximity sensors on the right fingertip, oriented to look down at the environment. Each sensor measures the distance between itself and the closest object as a scalar value.

4) Pre-training data collection via primitive motions

First, we collect primitive data D_{prim} for the surface-following task in two basic environments, "Inclined" and "Step", in simulation. For the surface-following task, we define primitive motions along the x-axis as $u_x = u_x^{const}$, and along the z-axis as

$$u_z = \begin{cases} u_z^{up} & \text{if } F_x > F_x^{threshold} \text{ (hitting wall)} \\ u_z^{down} & \text{elif } F_z < 0.0 \text{ (out of contact)} \\ \alpha(F_z^{target} - F_z) & \text{otherwise (in contact)} \end{cases} \tag{3}$$

where we define the axis coordinates in the task frame, as illustrated in Fig. 4. The robot moves its end-effector at a constant velocity u_x^{const} in the horizontal direction. When it detects a collision with a wall, it moves the end-effector up by u_z^{up}; if out of contact, it moves the end-effector down by u_z^{down} to reestablish contact. Otherwise, it adjusts the velocity of the end-effector in the normal direction u_z to achieve the specified target force F_z^{target}, without tool slippage, given the current normal force F_z and the scaling factor α. In our experiment, we set parameters to $u_x^{const} = 0.3\,\mathrm{cm/s}$, $u_z^{up} = 0.5\,\mathrm{cm/s}$, $u_z^{down} = -0.5\,\mathrm{cm/s}$, $F_x^{threshold} = 0.5$ N, $F_z^{target} = 0.3$ N, and $\alpha = 0.1$. We collected 11 trajectories in the "Inclined" environment and 10 trajectories in the "Step" environment. For each trial, we record the end-effector positions, tactile data ($4 \times 4 \times 3$ axes for 2 fingertips), and proximity data (6 dim.) at a frequency of 20 Hz over a duration of 10 seconds.

5) Demonstration data collection

We collect demonstration data D_{task} to fine-tune the base policy π_b for learning the target task motion. For the simulated experiments, we collect 3 demonstration trajectories each for (1) the new target force where $F_z^{target} = 0.5N$ and (2) the stairs environment. For the physical robot experiments, we collect 3 demonstration trajectories for each downstream task, using different tools. We record the end-effector positions, tactile data ($4 \times 4 \times 3$ axes for 2 fingertips), and proximity data (6 dim.) at a frequency of 20 Hz for 10 seconds.

6) Seq2seq model training

The seq2seq model consists of 1 LSTM layer and 1 fully connected layer in the encoder, and 1 LSTM layer and 1 fully connected layer in the decoder. We apply dropout [22] to the fully connected layers following the LSTM layers in both the encoder and decoder to prevent over-fitting during pre-training, with a dropout rate of 0.2. We set the dimensions of the cell state and hidden state to $c \in \mathbb{R}^{100}$ and $h \in \mathbb{R}^{100}$, respectively. We employ stochastic gradient descent [23] to update the weights of the neural network. We normalise all end-effector position data, shear force and slip data, and proximity data to the range $[-0.9, 0.9]$.

First, we pre-train the seq2seq model using primitive data as detailed in Section III-A. We train the model for 2000 epochs at a learning rate of 0.001, utilising a loss function defined in Eq. (2) with $\beta = 0.1$. It is important to note that the collection of primitive data and the pre-training of the base policy are only required once. This single pre-trained model serves as the base policy for all downstream tasks. After pre-training the base policy, we freeze the weights of all layers in the encoder and the LSTM layer in the decoder. For each task, we fine-tune the fully connected layer in the decoder on demonstration data for 300 epochs at a learning rate of 0.005 using the loss function Eq. (1).

C. Results and Discussion

1) Baseline

We compared our proposed approach, denoted as 'Fine-tuned,' with a behaviour cloning baseline, "Demo only" [24], which uses a policy trained solely on demonstration data without any pre-training in simulation, and an ablated baseline, "No FT," which uses the pre-trained policy without fine-tuning in the target domain. We also included two alternative methods used in settings with limited supervised data: Gaussian Processes (GP) with LSTM [25], to incorporate temporal information for fair comparison, trained on demonstration data without pre-training; and Dataset Aggregation (DAgger) [26], where we trained on pre-training data in the first iteration, and on aggregated data—including additional fine-tuning demonstrations as corrections—in the second. In all approaches, we collected three demonstrations and empirically tuned hyperparameters to minimise the training loss.

2) Evaluation metrics

For the "Inclined" environment in Fig. 4, we computed: 1) the RMSE between the ground truth and the followed

Evaluation Metric	Simulation			Physical setup		
	Demo only	**No FT**	**Finetuned**	**Demo only**	**No FT**	**Finetuned**
RMSE of slope (rad) ↓	0.082 ± 0.007	0.052 ± 0.005	**0.042** ± 0.006	0.76 ± 0.21	0.30 ± 0.10	**0.22** ± 0.05
RMSE of force (N) ↓	0.72 ± 0.05	0.29 ± 0.05	**0.14** ± 0.03	0.052 ± 0.006	0.035 ± 0.002	**0.017** ± 0.004

TABLE III: Task performance on online changing inclination from -0.25 rad. (down) to 0.0 rad. (flat) and to 0.25 rad. (up).

inclination, and 2) the RMSE between the desired contact force and the applied force to evaluate how effectively the tool followed the surface geometry while applying the desired force. We collected demonstration data for test cases in addition to the training data to obtain the desired contact force for evaluation. For surfaces with steps (i.e., "Step" and "Stairs" in Fig. 4), the RMSE metrics were unsuitable due to intentional tool lifting when hitting a step. Thus, we used wiped area (%)—the contact area relative to the total surface area—as the evaluation metric. Since demonstrations aim to maximise the wiped area by quickly reestablishing contact after lifting the tool, this metric reflects both imitation accuracy and task completeness.

3) Task completion

In the "small brush" painting task on a physical setup, "Finetuned" painted $93.5 \pm 1.68\%$ of the surface, outperforming "No FT" ($87.1 \pm 1.61\%$) and "Demo Only" ($72.7 \pm 4.75\%$). In the "broom" task, it swept 8.0 ± 1.0 dust pieces out of 10, compared to 2.7 ± 1.53 for "No FT" and 6.3 ± 0.58 for "Demo Only".

4) Online adaptation

Table III shows the quantitative evaluation when online adapting to changing inclination of the surface. Furthermore, when the surface deformed as the robot wiped the paper, as shown in the rightmost target environment in Fig. 4, the robot successfully adapted its motion to the deforming surface.