Supplementary Material of Touch2Robot

Here we lay down the details of the data collection, training, and testing process. More technical details are given here to illustrate our method and implementations better.

A. Additional Method Details

This section provides details on tactile processing, task definitions and rewards, and model training.

A. Tactile Processing and Semantic Alignment

Touch2Robot represents human, simulated-robot, and real-robot tactile observations in a shared binary semantic contact space. We first process each sensing modality independently and then align corresponding human and robot contact regions for retargeting and evaluation.

The human glove provides 256 taxel readings, which are grouped into semantic fingertip and palm regions according to the glove layout in Fig. 10. Regional tactile responses are binarized using recording-specific thresholds, with separate thresholds for fingertip and palm regions to account for their different signal ranges. For cross-embodiment alignment with LEAP, we use four fingertip regions corresponding to the thumb, index, middle, and ring fingers, together with four palm regions. The remaining human fingertip region has no robot counterpart and is excluded from contact comparison.

For simulated robot contact, we define eight semantic contact regions corresponding to the LEAP tactile layout: four fingertip regions and four palm regions, as shown in Fig. 11. Each robot region is paired with its corresponding human semantic region. The thumb, index, middle, and ring fingertip sensors correspond to human regions $T1$–$T4$, respectively. The four palm sensors correspond to human regions $P1$–$P4$. We set the simulated force threshold to 0.5 N for both fingertip and palm regions.

For real robot contacts, we record the raw tactile readings together with their timestamps. We set a channel specific threshold to handle varying baselines and noise levels. For the fingertip TwinTac channels, thresholds are obtained from an unloaded calibration recording. The FSR stream uses a zero threshold in its native output units.

B. Task Environments and PPO Training

We evaluate Touch2Robot on four contact-rich manipulation tasks *Pick*, *Rotate*, *Wipe* and *Drawer*. Fig. 12 shows the four LEAP Hand environments in Mujoco. The PPO training hyperparameters are summarized in Table VII.

C. Retargeter Distillation

We construct paired human robot sequences from teacher trajectories, with 30 trajectories for each task. We add Gaussian noise $\varepsilon \sim \mathcal{N}(0, 0.1^2)$ to the joint positions. The same joint noise are applied to both the human input and the robot teacher target, preserving their frame-wise correspondence, while the object geometry and scene configuration remain unchanged. We collect 30 teacher trajectories per task and augment them to a total of 400 training rollouts.

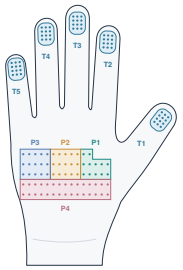

Fig. 10: Human tactile-region.

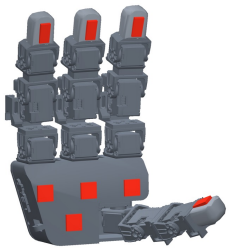

Fig. 11: Leap tactile-region.

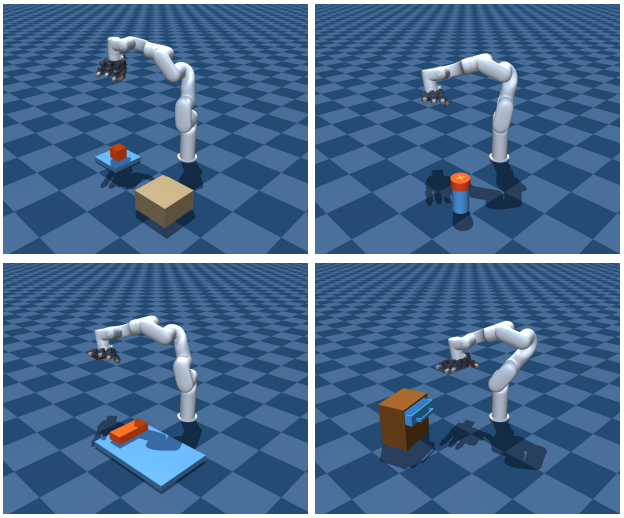

Fig. 12: Simulated task settings.

At each timestep, the student uses a causal Transformer to fuse a recent history of raw human motion retargeting, object poses, tactile preferences. The geometry feature is produced by a PointNet style encoder. The student is supervised using the teacher's robot joint targets and binary contact labels. The architecture and training settings are summarized in Table VIII.

D. Diffusion Policy Training

For each task, we train downstream policies on robot demonstrations obtained using each compared data-collection method. Each demonstration contains synchronized RGB observations from two camera views, robot joint states, tactile readings, and joint-target commands. We use the LeRobot implementation of Diffusion Policy and optimize the standard noise-prediction objective. At deployment, the policy predicts a sequence of joint targets and executes the first 16 actions before replanning from the latest observations. The main training settings are summarized in Table IX; all remaining architecture

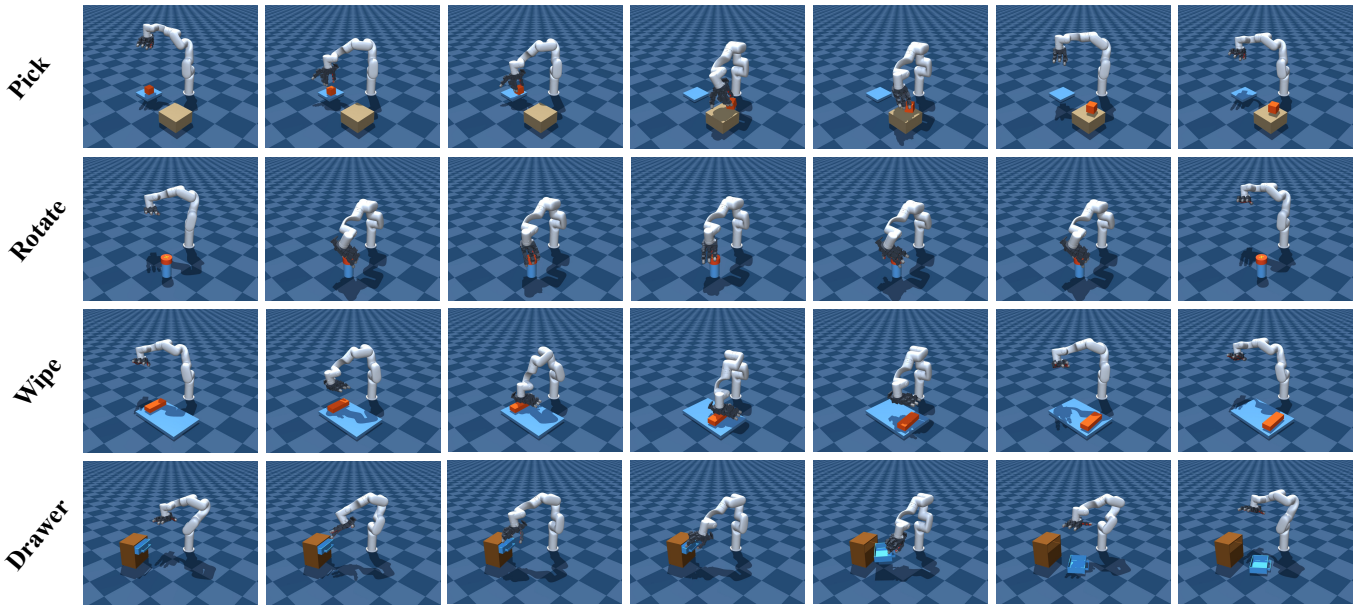

Fig. 13: **Task execution sequences.** Example rollouts for Pick-and-Place, Rotate, Wipe, and Drawer.

TABLE VII: PPO training hyperparameters.

Hyperparameter	Value
Parallel environments	200
Rollout length	512
Number of mini-batches	4
Optimization epochs	5
Hidden dimensions	$[1024, 1024, 512]$
Activation	ELU
Learning rate	3×10^{-4}
PPO clip range	0.2
Maximum gradient norm	1.0
Discount factor γ	0.96
GAE parameter λ_{GAE}	0.95
Initial action-noise std.	0.8
Desired KL divergence	0.016
Entropy coefficient	0

TABLE VIII: Unified retargeter training hyperparameters.

Hyperparameter	Value
History length H	16
Geometry feature dimension	32
Transformer layers / heads	4 / 8
Tactile input dropout	0.25
Optimizer	AdamW
Batch size	128
Learning rate	3×10^{-4}

and diffusion settings follow the default configuration.

B. Baseline Implementation Details

This section describes how the retargeting baselines are adapted to our human demonstration data and LEAP-Hand evaluation setting. Unless otherwise specified, all methods use the same human motion, object trajectories, robot model, and simulation environment. Human tactile measurements are provided only to Touch2Robot.

TABLE IX: Diffusion Policy settings.

Parameter	Value
Sampling frequency	20 Hz
RGB views / resolution	2 / 224×224
Observation steps	5
Prediction horizon	24
Exec. steps	16
Optimizer	Adam
Learning rate	10^{-4}
Weight decay	10^{-6}
Batch size	64
Training updates	50,000

A. Retargeting Baselines

Dex Retargeting. We use Dex Retargeting as a purely kinematic baseline. The MANUS hand motion is first transformed into the robot reference frame, after which the human wrist pose and fingertip geometry are mapped to the LEAP Hand through frame-wise constrained optimization. Robot joint limits are enforced during optimization, and the solution from the previous frame is used to initialize the next frame for temporal continuity. The resulting LEAP joint trajectory is used directly as the retargeted robot motion. This baseline does not use object dynamics or human tactile measurements.

ConTrack. ConTrack requires robot-side hand-motion, object-motion, and geometric contact references. We therefore preprocess each human demonstration into the corresponding LEAP-Hand representation. Human hand motion and object poses are first transformed into a common metric coordinate frame and temporally aligned. We then solve sequential, joint-

TABLE X: **User-study questionnaire.**

ID	Dimension	Question
Q1	Contact Awareness	I could identify which parts of the robot hand were in contact with the object.
Q2	Adaptation Confidence	I felt confident in adjusting my movements when the robot hand's contact was not as intended.
Q3	Feedback Usefulness	The interface feedback helped me judge whether the demonstration was readily for execution by the robot.
Q4	Ease of Use	I could use the interface feedback without noticeably interfering with my natural movements.

limit-constrained IK to obtain a LEAP joint reference from the human wrist pose and fingertip positions.

To construct the contact-style reference required by ConTrack, we extract human–object proximity from the reconstructed MANO hand. We divide the fingers into 15 segments and mark a segment as contacting when any of its vertices lies inside the object or within 5 mm of its surface. The closest object-surface point is recorded in the object-local frame and associated with the corresponding LEAP finger link. The resulting LEAP joint trajectory, demonstrated object trajectory, and geometry-derived contact references are then used to train ConTrack in our simulator.

We retain ConTrack's original adaptive task–style optimization, but replace its original demonstration source with the above references derived from our human data. Importantly, ConTrack does not receive the measured tactile glove signals used by Touch2Robot; its contact supervision is obtained only from hand–object geometry.

DexMachina. DexMachina is adapted from its original functional-retargeting formulation to our single-hand LEAP setting. We use the same demonstrated object trajectory as the task reference and construct the robot-side kinematic reference from the human motion using the same coordinate alignment and IK procedure described above. Demonstration-derived contact targets are obtained geometrically from the reconstructed human hand and object, rather than from the tactile glove.

We retain DexMachina's task-tracking, imitation, and contact objectives, together with its virtual-object-assistance curriculum. Terms associated with the second hand in the original bimanual formulation are removed for our single-hand tasks. The virtual assistance is progressively reduced during training until the object evolves entirely under robot–object dynamics.

C. Additional Evaluation Details

This section provides additional details on the evaluation procedures used in the main paper. We first define the contact-based metrics for tactile fidelity and preference alignment, and then describe the contact-onset metric and user-study protocol.

A. Contact Precision, Recall, F1, and False-Positive Rate

We compare binary contact sequences defined over the shared semantic regions. Let $y_{t,c} \in \{0,1\}$ denote the reference contact state and $\hat{y}_{t,c} \in \{0,1\}$ the evaluated contact state, where t is the time index and c is the semantic region. For tactile fidelity, $y_{t,c}$ is the measured real-robot contact and $\hat{y}_{t,c}$ is the reconstructed target-hand contact. For preference alignment, $y_{t,c}$ is the aligned human tactile preference and $\hat{y}_{t,c}$ is the reconstructed target-hand contact.

We count true positives (TP) when both sequences indicate contact, false positives (FP) when only the evaluated sequence indicates contact, false negatives (FN) when only the reference indicates contact, and true negatives (TN) when neither indicates contact. We aggregate these counts over all time–region pairs before computing the metrics.

Precision and recall are

$$\text{Precision} = \frac{TP}{TP+FP}, \qquad \text{Recall} = \frac{TP}{TP+FN}. \tag{11}$$

We compute F1 and the false-positive rate as

$$\text{F1} = \frac{2TP}{2TP+FP+FN}, \qquad \text{FPR} = \frac{FP}{FP+TN}. \tag{12}$$

B. Contact-Onset Error

We align contact sequences using the recorded replay timestamps and their mapping to reference-trajectory time. For each trajectory and semantic region, we compare the contact-onset times. Let t_i^{ref} and t_i^{eval} denote the corresponding onset times in seconds. A trajectory–region pair is considered valid when both sequences contain a contact onset. For N valid pairs, the mean contact-onset error is

$$E_{\text{onset}} = \frac{1000}{N} \sum_{i=1}^{N} \left| t_i^{\text{eval}} - t_i^{\text{ref}} \right|. \tag{13}$$

The factor 1000 converts seconds to milliseconds. Missing contacts are captured by the contact-overlap metrics and are excluded from the onset error.

C. User Study Protocol and Questionnaire

We conduct a within-subject comparison between Visual Feedback and Touch2Robot. Visual Feedback displays the retargeted robot hand without contact information, whereas Touch2Robot additionally visualizes reconstructed target-hand contacts. Participants perform the Rotation task under both conditions following a short familiarization session. Condition order is counterbalanced across participants, while initial states and demonstration-collection time are matched. Participants complete the questionnaire in Table X immediately after each condition.

Ten participants rate each item on a seven-point Likert scale. Statistical significance is evaluated on the original ratings using paired two-sided Wilcoxon signed-rank tests with Holm correction.