APPENDIX A
SIMULATION ENVIRONMENT AND SOFT-HAND MODELING

This section describes the simulation environment used for retargeting and evaluation. The simulator is designed to capture the dominant deformation modes and contact interactions of the soft robotic fingers while maintaining numerical stability and computational efficiency. Rather than explicitly modeling pneumatic fluid dynamics, we adopt a hybrid rigid–soft abstraction that approximates pressure-induced bending through internal actuation torques.

A. Soft Finger Model

Each soft robotic finger is modeled as a deformable body using a stable Neo-Hookean material formulation [9]. Material parameters are chosen to approximate the compliance of the physical elastomer while ensuring stable simulation under sustained contact-rich interaction.

B. Rigid Spine Approximation

To enable controllable deformation, each finger contains an internal rigid spine composed of a chain of capsule-shaped rigid bodies aligned along the finger's longitudinal axis (Fig. 3). Adjacent capsules are connected via spherical joints, forming an articulated backbone that supports smooth bending under applied torques. Joint stiffness and damping are tuned to suppress high-frequency oscillations while preserving compliant motion.

C. Soft–Rigid Coupling

The rigid spine is coupled to the surrounding deformable mesh using distributed soft-to-rigid spring constraints. For each spine capsule, nearby FEM nodes within a fixed radius are connected via virtual springs. This distributed coupling acts as a tendon-like mechanism, transmitting spine motion to the soft body while avoiding localized stress concentrations and producing smooth, physically plausible deformation.

D. Actuation Abstraction

In the physical system, finger motion is generated through differential pressurization of three pneumatic chambers. In simulation, pneumatic actuation is abstracted by applying external torques to the rotational degrees of freedom of the spine capsules. Torques are applied about two orthogonal bending axes for all spine segments except the fixed base, inducing articulated bending that approximates the dominant deformation modes observed under pneumatic actuation. Explicit modeling of pressurized air dynamics is omitted to improve numerical stability and simulation throughput.

E. Boundary Conditions

Boundary conditions are applied at both the soft and rigid levels to anchor the hand. Soft-body nodes near the base of each finger are fixed in world coordinates, and the base capsule of each spine has all six degrees of freedom constrained. This prevents global drift and localizes deformation to the intended regions of the finger.

F. Collision Handling

Collision interactions are selectively enabled to balance physical realism and numerical stability. Self-collision within individual fingers and collisions between a spine and its enclosing soft body are disabled. Collisions between fingers and manipulated objects, as well as between spines of different fingers, are enabled to support realistic multi-finger contact during manipulation.

Multiple fingers are instantiated by arranging identical finger–spine assemblies around a circular base attached to the end-effector. Each finger is assigned a distinct collision group to allow fine-grained control over inter-finger interactions.

G. Simulation Parameters

Table IV summarizes representative simulation parameters used across experiments. When exact physical measurements are unavailable, parameters are chosen to fall within commonly used ranges for soft-body manipulation.

APPENDIX B
SURFACE GEOMETRY AND DISTANCE COMPUTATION

This section describes the intrinsic surface distance computation used throughout `SoftAct` to propagate contact influence along the hand surface rather than through free space.

A. Hand Surface Representation

The hand surface is represented as a triangular mesh modeled as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where vertices correspond to mesh vertices and edges connect adjacent triangles.

Given a 3D point x on the hand surface, it is projected to the nearest mesh vertex

$$\hat{v}(x) = \arg\min_{v_i \in \mathcal{V}} \|x - v_i\|_2.$$

TABLE IV: Simulation parameters (representative values).

Parameter	Value
Young's modulus	127 kPa
Poisson's ratio	0.48
Material density	1000 kg/m^3
Spine capsule length	10 mm
Number of spine segments	8
Joint stiffness	0.2 N·m/rad
Joint damping	0.02 N·m·s/rad
Soft–rigid spring stiffness	200 N/m
Simulation timestep	1 ms
Contact friction coefficient	0.6

B. Geodesic Distance

The geodesic distance between two surface points x and y is defined as

$$d_{\text{geo}}(x, y) = \min_{P \in \mathcal{P}(\hat{v}(x), \hat{v}(y))} \sum_{(i,j) \in P} \|v_i - v_j\|_2,$$

where $\mathcal{P}(\cdot, \cdot)$ denotes the set of all vertex paths on the mesh graph. Shortest paths are computed using Dijkstra's algorithm with Euclidean edge weights.

This formulation ensures that contact influence respects surface curvature and articulated geometry, which is particularly important for highly curved and deformable structures such as soft fingers.

APPENDIX C
OFFLINE FORCE-BALANCED FINGER ASSIGNMENT

This section corresponds to **Stage 1** of `SoftAct` (Sec. III), which computes a fixed, force-balanced mapping between human and robot fingers prior to execution.

A. Contact Representation

At each timestep t, demonstrated contacts are represented as a set

$$\mathcal{C}_t = \{(c, f)\},$$

where c denotes a contact location on the hand surface and f is the corresponding contact force vector.

B. Geodesic Force Diffusion

Contact forces are diffused over the hand surface using a geodesic heat kernel. For mesh vertex v at timestep t, the heat value is defined as

$$h_v^t = \sum_{(c,f) \in \mathcal{C}_t} \|f\|_2 \exp(-\lambda\, d_{\text{geo}}(v, c)),$$

where λ controls the spatial decay of influence.

C. Per-Finger Load Estimation

Heat values are accumulated over time and summed over vertices associated with each skeleton finger region to obtain the total contact load

$$F_r = \sum_t \sum_{v \in \mathcal{R}_r} h_v^t,$$

where $\mathcal{R}_r$ denotes the mesh region associated with skeleton finger r. When segmentation is unavailable, contacts are attributed to the nearest skeleton finger based on bone proximity.

D. Load-Balanced Allocation

Given N_f available robot fingers, allocation is performed by solving

$$\min_{\{n_r\}} \max_r \frac{F_r}{n_r} \quad \text{s.t.} \quad \sum_r n_r = N_f,$$

where n_r denotes the number of robot fingers assigned to skeleton finger r.

E. Workspace-Aware Finger Matching

For each skeleton fingertip, we compute its demonstrated trajectory envelope in the end-effector (EE)–local frame. The envelope is aligned to the robot EE xy-plane via plane fitting followed by Procrustes alignment. Each robot finger workspace $\mathcal{W}_i$ is estimated through random pressure sampling.

The assignment cost between robot finger i and skeleton fingertip r is defined as

$$C_{ir} = \|\mu_i - \mu_r\|_2 + \beta\, \mathbb{I}[\mathcal{W}_i \cap \mathcal{W}_r = \varnothing]\,.$$

The optimal assignment is solved using the Hungarian algorithm, yielding a fixed mapping $\pi(i)$ that remains constant throughout execution.

APPENDIX D
ONLINE CONTACT-INFORMED FINGERTIP REFINEMENT

This section corresponds to **Stage 2** of `SoftAct` (Sec. III), which performs online refinement of fingertip targets during execution.

A. Geodesic Contact Weighting

For each demonstrated contact c_j^t, we compute its geodesic distance to the current fingertip position s_i^t,

$$d_{ij}^t = d_{\text{geo}}(s_i^t, c_j^t), \quad w_{ij}^t = \|f_j^t\|_2 \exp\big(-\lambda\, d_{ij}^t\big)\,.$$

B. Fingertip Adjustment Rule

The adjustment vector is computed as

$$\delta_i^t = \frac{\sum_j w_{ij}^t (c_j^t - s_i^t)}{\sum_j w_{ij}^t + \epsilon}, \qquad \|\delta_i^t\|_2 \le \delta_{\max}.$$

The corrected fingertip target is tracked using Jacobian-based torque control in the end-effector frame.

APPENDIX E
SOFTACT RETARGETING ALGORITHM

We summarize the full two-stage force-aware retargeting procedure used by `SoftAct`.

Algorithm 1 Force-Balanced Contact-Centric Retargeting

Require: Demonstration frames $\{(B_t, V_t, \mathcal{C}_t)\}_{t=1}^{T}$
Require: Robot state $(p_{ee}^t, R_{ee}^t, \{s_i^t\}_{i=1}^{N_f})$
Ensure: Robot actions $\{a_t\}_{t=1}^{T}$

```
Stage 1: Offline force-balanced finger assignment
Compute geodesic distances on the hand surface
Accumulate contact force magnitudes per human finger
Allocate robot fingers to minimize maximum per-finger load
Solve workspace-aware assignment to obtain mapping π(i)
Stage 2: Online retargeting with contact refinement
for t = 1 to T do
    Compute baseline EE target (p̂_ee^t, R̂_ee^t)
    for each soft finger i do
        Compute geodesic-weighted contact influence
        Compute fingertip adjustment δ_i^t
        Track skeleton fingertip π(i) with adjusted target
    end for
    Execute robot action a_t
end for
```

APPENDIX F
TRAINING DETAILS: SOFT FINGER DIFFUSION POLICY

This section describes the training setup for the low-dimensional diffusion policy used to execute retargeted soft-finger trajectories.

A. Policy Overview

We train a conditional diffusion policy that predicts short-horizon soft-finger actions given recent robot observations. The policy operates in a low-dimensional action space and is conditioned on proprioceptive state and task-relevant features provided as a global conditioning signal.

B. Sequence Structure

Training is performed on fixed-length trajectory segments with a horizon of 16 steps. Each training sample includes the most recent 2 observation steps and predicts the next 8 action steps. No action latency is modeled, and past actions are not explicitly included in the observation stream.

C. Model Architecture

The policy backbone is a one-dimensional conditional UNet with three downsampling stages. The diffusion timestep embedding dimension is 256, with channel widths of $\{256, 512, 1024\}$ and kernel size 5. Group normalization with 8 groups is used throughout. Conditioning is applied globally using concatenated observations across the temporal context window.

D. Diffusion Process

We employ a DDPM-style diffusion process with 100 diffusion steps during training and inference. The noise schedule follows a cosine-based schedule with variance clipping enabled. The model is trained to predict noise residuals (ϵ-prediction).

E. Optimization and Training Protocol

Training uses the AdamW optimizer with learning rate 1×10^{-4}, $\beta = (0.95, 0.999)$, and weight decay 1×10^{-6}. A cosine learning-rate schedule with 500 warmup steps is applied. Models are trained for 3000 epochs with batch size 256. An exponential moving average (EMA) of model parameters is maintained throughout training and used for evaluation.

F. Inference

At test time, action sequences are generated using 100 diffusion denoising steps and executed in a receding-horizon fashion.

TABLE V: Diffusion policy training hyperparameters.

Parameter	Value
Horizon	16
Observation steps	2
Action steps	8
Diffusion steps	100
Batch size	256
Learning rate	1×10^{-4}
EMA decay	0.999
Training epochs	3000

APPENDIX G
LOW-LEVEL CONTROLLER TRAINING DATA

We visualize a downsampled dataset used for the low level soft robot finger controller used for real-world deployment.

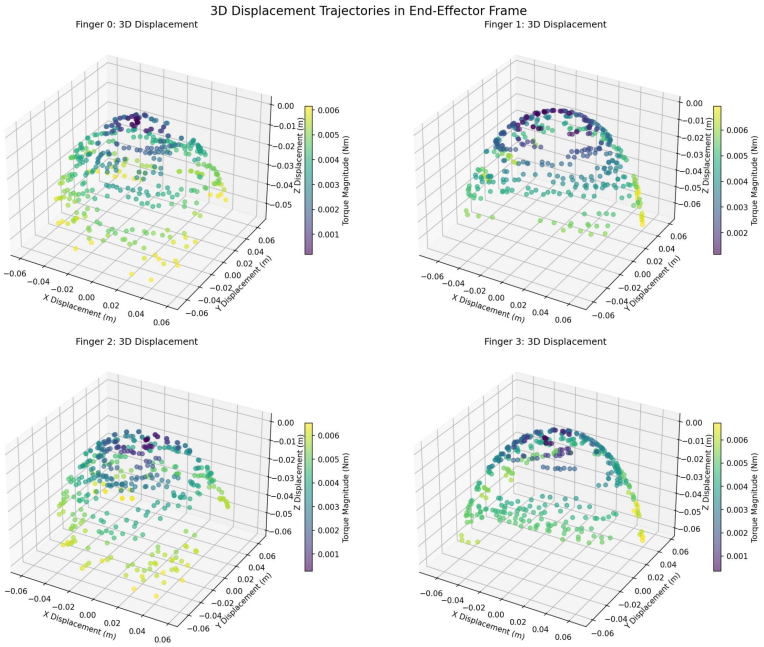

Fig. 8: Down-sampled fingertip displacement data