

Table 2: Definition of Symbols

Symbol	Description
g^*	Reference value corresponding to quantity g
\bar{g}	Nominal value corresponding to quantity g
${}^A\mathbf{r}_{AB} \in \mathbb{R}^3$	Position of frame B from frame A in frame A
$\Phi_{AB} \in SO(3)$	Orientation of frame B in frame A
${}^A\mathbf{v}_{AB} \in \mathbb{R}^3$	Linear velocity of frame B from frame A in frame A
${}^A\boldsymbol{\omega}_{AB} \in \mathbb{R}^3$	Angular velocity of frame B from frame A in frame A
$\Delta : SO(3) \times SO(3) \rightarrow \mathbb{R}^3$	Difference between two orientations as a rotation error vector [30]
B	Frame attached to the base of the robot
E	Frame attached to the arm’s end-effector of the robot
H	Frame attached to the handle’s center on the object
I	Inertial frame (fixed to the initial frame H on execution)
\mathbf{h}_{com}	Centroidal momentum of the robot
$\mathbf{q}_b = ({}^I\mathbf{r}_{IB}, \Phi_{IB})$	Base pose of the robot
\mathbf{q}_j	Joint positions of the robot
\mathbf{q}_o	Joint positions of the object
$\dot{\mathbf{q}}_j$	Joint velocities of the robot
$\dot{\mathbf{q}}_o$	Joint velocities of the object
$\ddot{\mathbf{q}}_j$	Joint acceleration of the robot
$\boldsymbol{\tau}_j$	Joint torques applied to the robot
\mathbf{m}	Manipulation contact mode between robot and the object
$\mathbf{x}_r = (\mathbf{h}_{com}, {}^I\mathbf{r}_{IB}, \Phi_{IB}, \mathbf{q}_j)$	Kinematic state of the robot
$\mathbf{x}_o = (\mathbf{q}_o, \dot{\mathbf{q}}_o)$	Kinematic state of the object
$\mathbf{x} = (\mathbf{x}_r, \mathbf{x}_o)$	Kinematic state of the robot and the object
$\mathbf{M} = \{\mathbf{m}_t\}_{t=1}^{T_{task}}$	Manipulation schedule (Sequence of manipulation contact modes)
$\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T_{task}}$	Kinematic state trajectory (Sequence of kinematic states)
T_{task}	Duration (in s) of the trajectory for the task
$\mathbb{1}_{object} := \mathbb{1}_{object}(\mathbf{m})$	True iff at least one end-effector is either already in contact with or is establishing contact with the object
$\mathbb{1}_{prehensile} := \mathbb{1}_{prehensile}(\mathbf{m})$	True iff a prehensile interaction is active
$\phi \in [0, 1]$	Phase signal that helps index the reference trajectory $(\mathbf{X}^*, \mathbf{M}^*)$
v_t^ϕ	Task phase rate (first order-dynamics model for ϕ)
\hat{v}_t^ϕ	Task phase rate based on reward-dependent functions
$\bar{\phi}$	Nominal phase signal computed using $v^\phi = \frac{1}{T_{task}}$
δ_v	Residual phase rate (output from the policy)
dt	Step-size (in s) of the environment
\mathbf{o}_t	Observation from the environment at time-step t
\mathbf{a}_t	Action applied to the environment at time-step t
r_t	Reward from the environment at time-step t

B Demonstrations using Loco-Manipulation Planner

The loco-manipulation planner from Sleiman et al. [10] efficiently generates physically consistent demonstrations for our proposed framework. The planner relies on a high-fidelity model that integrates the robot’s full centroidal dynamics and first-order kinematics with the object’s full dynamics [35]. This helps ensure that the discovered behaviors are dynamically feasible. The planner outputs the following the continuous states and system inputs: $\mathbf{x} := (\mathbf{x}_r, \mathbf{x}_o) = (\mathbf{h}_{com}, \mathbf{q}_b, \mathbf{q}_j, \mathbf{q}_o, \dot{\mathbf{q}}_o)$ and $\mathbf{u} = (\mathbf{w}_e, \dot{\mathbf{q}}_j)$, where the robot state \mathbf{x}_r includes the centroidal momentum \mathbf{h}_{com} , base pose \mathbf{q}_b , and joint positions \mathbf{q}_j , whereas the object state \mathbf{x}_o consists of its generalized coordinates \mathbf{q}_o and velocities $\dot{\mathbf{q}}_o$. The control input \mathbf{u} is composed of the robot’s joint velocities $\dot{\mathbf{q}}_j$ and the contact wrenches \mathbf{w}_e acting at the robot’s end-effectors.

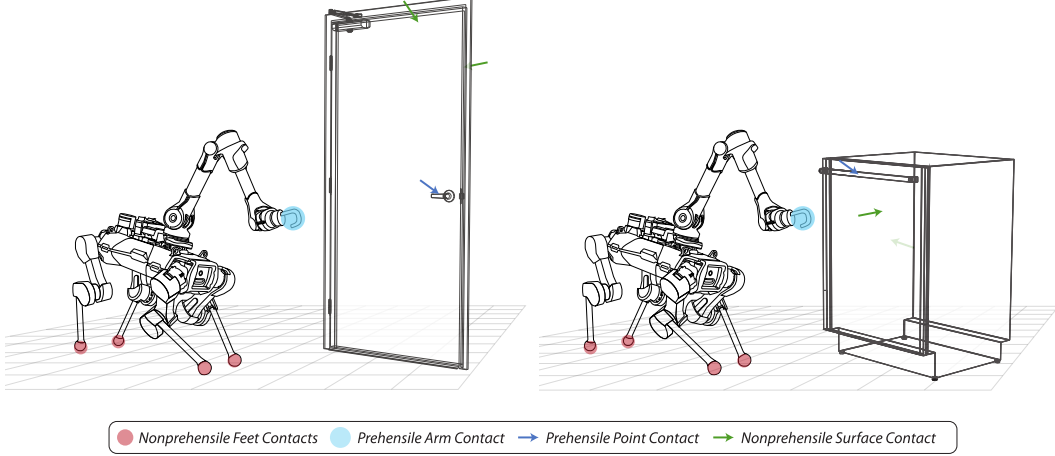


Figure 7: Illustration of two loco-manipulation tasks: i) traversal of a large articulated object, and ii) dishwasher manipulation. The user-defined robot end-effector contacts and object affordances are highlighted.

Moreover, from a set of user-defined object affordances and the robot’s end-effectors for interaction, a discrete variable \mathbf{m}_k represents the manipulation contact mode. A contact mode is a state-action pair, where the contact state encodes possible robot-object interaction combinations, and a contact-switching action indicates whether a contact is established, broken, or maintained. By exploiting loco-manipulation-specific pruning rules, the planning algorithm in [10] efficiently solves for a multi-modal sequence via a sampling-based bi-level search over manipulation modes \mathbf{m}_k and continuous state-input trajectories $\langle \mathbf{x}_k(t), \mathbf{u}_k(t) \rangle$, aiming to connect the start and goal states. It then refines the resulting plan through a single long-horizon TO while fixing the discovered contact sequence. We refer the reader to [10] for further details on the multi-contact planner.

While the references generated from the planner contain the control inputs $\mathbf{u}_k(t)$, these signals are usually tracked on hardware through a whole-body quadratic programming (QP) controller. This controller computes the necessary joint torques to achieve the desired motions. However, the QP controller’s robustness is limited because of its several assumptions, such as no slippages and precise command tracking. We use only the reference states \mathbf{X}^* and contact modes \mathbf{M}^* to address these limitations to guide the RL training process. This approach allows the NN policy to learn the underlying actuator dynamics during training and adapt better to the inherent uncertainties and variations encountered during real-world operations.

Table 3 summarizes the single demonstrations generated for each task. In less than a minute, all demonstrations are discovered on an Intel Core i7-10750H CPU@2.6GHz hexacore processor. Navigating through a spring-loaded pull door stands out as the most complex task. This can be attributed to several factors, including the long time horizon, the requirement for a stable prehensile interaction, and the multiple contact transitions involved. Discovering these modes using standard RL would necessitate careful design and tuning of handcrafted rewards, which we want to alleviate through our formulation.

Table 3: Computation time and trajectory duration for demonstrations generated using the planner [10].

Task	Computation Time (s)	Trajectory Duration (s)	Trajectory Length
Door Push	6.8	16.8	1195
Dishwasher Open	25.0	11.2	814
Dishwasher Close	23.1	12.6	902
Door Pull	44.2	23.8	1725

C MDP Formulation and PPO Training

This section summarizes the terms that formulate the proposed MDP and the learning algorithm.

C.1 Observation Terms

Table 4 specifies the observation terms and the noise added to them during training. The critic obtains the same observations as the actor but without any noise applied. Importantly, for the adaptive-phase dynamics formulation, the previous action \mathbf{a}_{t-1} comprises both the robot commands $\bar{\mathbf{a}}_{t-1}$ and the residual phase δ_v .

Table 4: Observation Terms Summary. We do not perform any scaling or clipping on individual observation terms. All noise models are additive in nature.

Term Name	Definition	Noise
Robot Base Position Difference	$I\mathbf{r}_{IB} - I\mathbf{r}_{IB}^*$	$\mathcal{U}(-0.05, 0.05)$
Robot Base Orientation Difference	$\Phi_{IB} \boxminus \Phi_{IB}^*$	$\mathcal{U}(-0.1, 0.1)$
Robot Base Linear Velocity	${}_B\mathbf{v}_{IB}$	$\mathcal{U}(-0.1, 0.1)$
Robot Base Angular Velocity	${}_B\boldsymbol{\omega}_{IB}$	$\mathcal{U}(-0.2, 0.2)$
Robot Joint Position Difference	$\mathbf{q}_j - \mathbf{q}_j^*$	$\mathcal{U}(-0.01, 0.01)$
Robot Joint Velocity	$\dot{\mathbf{q}}_j$	$\mathcal{U}(-1.5, 1.5)$
Robot Arm End-effector Position	$I\mathbf{r}_{IE}$	$\mathcal{U}(-0.05, 0.05)$
Object Joint Position Difference	$\mathbf{q}_o - \mathbf{q}_o^*$	$\mathcal{U}(-0.05, 0.05)$
Nominal Task Phase	$\bar{\phi}$	-
Adaptive Task Phase	ϕ	$\mathcal{U}(-0.005, 0.005)$
Adaptive Task Phase Speed	$\dot{\phi}$	$\mathcal{U}(-0.005, 0.005)$
Previous Action	\mathbf{a}_{t-1}	-

C.2 Reward Terms

To design a task-agnostic reward function, we split the reward function into generic reference-tracking terms that stabilize the open-loop trajectories and standard penalty terms that ensure smooth motions: $\bar{r}_t^{total} = r_t^{track} + r_t^{regularize}$. For the adaptive phase formulation, the reward also includes the task progress: $r_t^{total} = \bar{r}_t^{total} + r_t^\phi$. The individual terms are:

$$\begin{aligned}
r_t^{track} &= \alpha_1 \cdot \|I\mathbf{r}_{IB} - I\mathbf{r}_{IB}^*\|^2 + \alpha_2 \cdot \|\Phi_{IB} \boxminus \Phi_{IB}^*\|^2 + \alpha_3 \cdot \|\mathbf{q}_j - \mathbf{q}_j^*\|^2 \\
&\quad + \alpha_4 \cdot \mathbb{1}_{object}^* \cdot \|\mathbf{q}_o - \mathbf{q}_o^*\|^2 + \alpha_5 \cdot \mathbb{1}_{prehensile}^* \cdot \|I\mathbf{r}_{IE} - I\mathbf{r}_{IH}\|^2, \\
r_t^{regularize} &= \beta_1 \cdot \|\boldsymbol{\tau}\|^2 + \beta_2 \cdot \|\dot{\mathbf{v}}_j\|^2 + \beta_3 \cdot \|\mathbf{v}_j\|^2 + \beta_4 \cdot \|{}_B\mathbf{v}_{IB}^z\|^2 + \beta_5 \cdot \|{}_B\boldsymbol{\omega}_{IB}^{xy}\|^2 \\
&\quad + \beta_6 \cdot \|\bar{\mathbf{a}}_t - \bar{\mathbf{a}}_{t-1}\|^2 \\
r_t^\phi &= \kappa_1 \cdot [\dot{\phi} \cdot \exp(-\kappa_2 \cdot \|\phi - \bar{\phi}\|^2)]
\end{aligned}$$

where symbols have their meanings from Table 2. The individual weights are tabulated in Table 5.

C.3 Domain Randomization

Domain randomization helps mitigate overfitting to specific models and addresses inherent unmodelled effects by introducing variability during training. In our setup, it takes the following form:

- **Object’s kinematics:** These include object dimensions (e.g. door width and height), positioning of object affordances (e.g. handle location on the panel), and handle types (cylinder or box). For every object category, we load 128 different kinematic variations.
- **Object’s dynamics:** These include friction and restitution, spring-damping coefficients for the hinge joint, and constant force/torque offset on the hinge joint.
- **Robot’s dynamics:** Similar to object dynamics, we vary the friction and restitution within $[0.4, 1.0]$. Additionally, the mass of the robot’s base is randomized within $\pm 10\%$ of its nominal values.
- **External disturbances:** At randomly sampled episode intervals, external pushes are applied to both the robot and the object. For the robot, this implies adding random velocity

Table 5: Reward Terms Summary. The environment scales the reward weights with the time-step dt [36]. For brevity, we drop the time-step t from individual quantities unless necessary. We use the same reward weights for all the loco-manipulation tasks.

Term Name	Definition	Weight
Robot Base Position Tracking	$\ I\mathbf{r}_{IB} - I\mathbf{r}_{IB}^*\ ^2$	-0.2
Robot Base Orientation Tracking	$\ \Phi_{IB} \ominus \Phi_{IB}^*\ ^2$	-0.2
Robot Joint Position Tracking	$\ \mathbf{q}_j - \mathbf{q}_j^*\ ^2$	-0.2
Object Joint Position Tracking	$\mathbb{I}_{object}^* \cdot \ \mathbf{q}_o - \mathbf{q}_o^*\ ^2$	-0.2
Robot Arm End-effector Position Tracking	$\mathbb{I}_{prehensile}^* \cdot \ I\mathbf{r}_{IE} - I\mathbf{r}_{IH}\ ^2$	-10.0
Action Rate	$\ \mathbf{a}_t - \mathbf{a}_{t-1}\ ^2$	-0.05
Robot Base Linear Velocity (along z)	$\ {}_B\mathbf{v}_{IB}^z\ ^2$	-0.5
Robot Base Angular Velocity (along xy)	$\ {}_B\boldsymbol{\omega}_{IB}^{xy}\ ^2$	-0.05
Robot Joint Velocity	$\ \dot{\mathbf{q}}_j\ ^2$	-1.0×10^{-5}
Robot Joint Acceleration	$\ \ddot{\mathbf{q}}_j\ ^2$	-1.0×10^{-5}
Robot Applied Joint Torque	$\ \boldsymbol{\tau}_j\ ^2$	-2.5×10^{-5}
Task Progress (only with adaptive phase)	$\hat{v}^\phi \cdot \exp(-10.0 \cdot \ \phi - \bar{\phi}\ ^2)$	25.0

(pushes) to the robot’s base. For the door, this is done by applying a randomly sampled external force on the door panel.

- **Reset state at the beginning of an episode:** We apply additive offsets to the robot’s reference configuration at $\phi_{init} = 0$ so that the policy is robust to varying initial locations of the robot in front of the door. Ideally, we would like to apply this at any randomly sampled ϕ ; doing so is non-trivial due to the difficulty in filtering invalid collision configurations.

C.4 Termination Term

We trigger episode termination when the robotic system loses balance or episode length times out. Typically, we infer a fall from a significant force acting on the robot’s base, indicating ground contact. However, distinguishing the source of this force becomes challenging during loco-manipulation tasks, as contact between the robot’s base and an object is both expected and sometimes permissible. Thus, we rely on the base to not drop below 0.3 m to correctly detect falls.

C.5 Adaptive-Phase Hyperparameters

We now list down the hyperparameters for the remainder of the MDP formulation. These include parameters for scaling the input actions and those for the adaptive-phase formulation in Eq. 4.

Table 6: MDP Hyperparameters.

Hyperparameter	Value
Episode length	15 s
Simulation time-step	0.005 s
Control decimation	4
Robot action scale: σ_1	0.5
Residual phase action scale (3): σ_2	0.01
Adaptive phase (4): α_1	0.2
Adaptive phase (4): α_2	0.2
Adaptive phase (4): α_4	0.2
Adaptive phase (4): λ	50.0
Adaptive phase (4): γ_i	0.002

485 C.6 Learning Algorithm

486 For each task, we train the policy using the on-policy RL algorithm, Proximal Policy Optimization
 487 (PPO) [34]. The actor and critic networks are designed as a Multi-Layer Perceptron (MLP) with
 488 a $[256 \times 128 \times 64]$ hidden-layer structure and an *ELU* activation function. A complete list of
 489 hyperparameters and their values is specified in Table 7.

Table 7: PPO Hyperparameters

Hyperparameter	Value
Empirical Normalization	True
Learning Rate (start of training)	1e-3
Learning Rate Schedule	“adaptive” (based on KL-divergence [36])
Discount Factor	0.99
GAE Discount Factor	0.95
Desired KL-divergence	0.01
Clip Range	0.2
Entropy Coefficient	0.0
Value Function Loss Coefficient	1.0
Batch Size	245,760 (4096×60)
Mini-Batch Size	61,440 (4096×15)
Number of Epochs	5
Number of iterations	10,000

490 D Supplementary Discussions

491 In this section, we highlight some of the overlooked challenges our trained policy could face upon
 492 real-world deployment and briefly mention solutions for them.

493 D.1 Object Locking Mechanism

494 Our policy is trained on doors where handles are treated as fixed object-attached links. However, a
 495 typical door can only be opened after being unlocked using its handle. By adapting our environment
 496 to incorporate a door-locking mechanism, our current setup results in behaviors involving the robot
 497 pushing the handle downwards, but with significantly lower success rates. One way to resolve this
 498 would be to use an asymmetric actor-critic structure, where the critic obtains the handle angle.

499 D.2 Unknown Object Type

500 In real-world scenarios, we expect the robot to autonomously traverse diverse doors without speci-
 501 fying the type of door (*i.e.*, push or pull door). One way to achieve this objective would be to first
 502 train a multi-task policy that can execute both door-traversal behaviors and then separately train a
 503 door-type estimator that outputs the appropriate task command to the policy.

504 D.3 Unknown Object State

505 In the current hardware experiments, we rely on explicit sensors such as door encoders to obtain
 506 the joint position of the panel. However, in more realistic scenarios, this information needs to be
 507 extracted from the robot’s onboard sensors. One mechanism to achieve this goal is doing teacher-
 508 student learning by treating the current policy as the teacher policy and training a student policy that
 509 receives depth images directly.

510 D.4 Applicability to Other Scenarios

511 In this work, we demonstrated the approach on multi-contact tasks that primarily involved articulated
 512 object manipulation. However, there is an open question on the generalization of the approach to
 513 other tasks and robots. These include rigid object manipulation, such as box pushing, and other
 514 morphologies, such as bimanual manipulation and dexterous hands. While we believe the method
 515 should work for these other scenarios, we leave this as part of future work.