HumanoidMimicGen: Data Generation for Loco-Manipulation via Whole-Body Planning and Adaptation

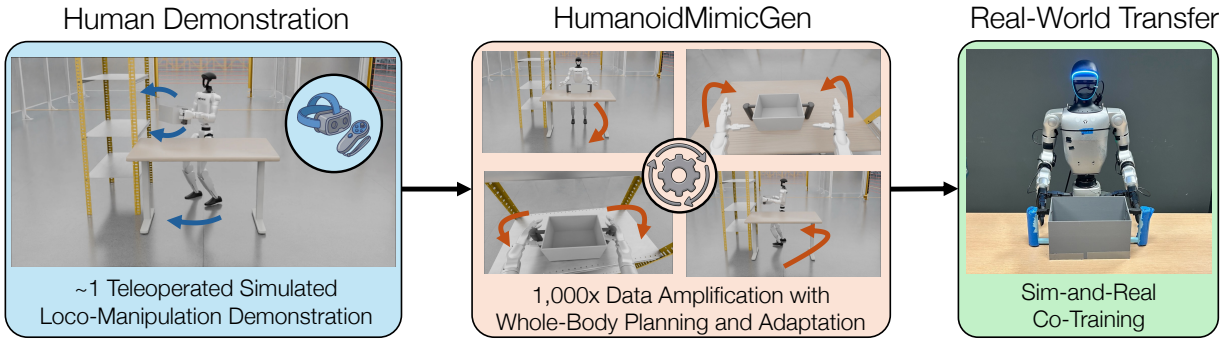

Fig. 1: **HumanoidMimicGen Overview.** We present HumanoidMimicGen, an algorithm for humanoid legged loco-manipulation demonstration generation. *Left*: first, a human teleoperator collects a loco-manipulation task demonstration. *Center*: HumanoidMimicGen synthetically generates thousands of new demonstrations across different scene and object layouts by adapting local segments of the human demonstration using a novel integration of whole-body planning and control. *Right*: These demonstrations can be used to train performant loco-manipulation policies in both simulated and real-world settings.

I INTRODUCTION

Vision-Language-Action (VLA) models [1, 2], trained on large robotic manipulation datasets, have recently demonstrated a remarkable capability to enable robots to autonomously complete a wide range of manipulation tasks. However, this paradigm has largely been demonstrated in stationary robot manipulation settings, where the robot does not need to walk around in its workspace. This contrasts with the promise of humanoid robots: their human-like form factor should enable them to move and interact fluidly with environments designed for humans. However, applying the VLA paradigm to humanoid loco-manipulation remains challenging due to two key difficulties. First, VLAs require large-scale manipulation datasets to develop their capabilities — sourcing these datasets for even stationary manipulation settings remains a challenging endeavor due to the persistent human time and effort required to collect them. The data collection process typically involves robot teleoperation, where teams of human operators control fleets of robots to collect datasets over months [3, 4, 5, 6]. Second, intelligently controlling a humanoid is a difficult problem due to its high degrees of freedom and the constant need to dynamically balance itself during simultaneous manipulation and locomotion.

Consequently, large-scale loco-manipulation datasets are not readily available, making VLA training prohibitive.

Simulation and synthetic data generation offer a compelling means to address the challenge of sourcing large-scale datasets for training VLAs. Automated data generation tools [7, 8] allow for generating large volumes of data, and furthermore, recent results have shown that these synthetic datasets can train real-world manipulation policies [9, 10, 11, 12]. Furthermore, such tools can enable studies into how data quality and policy learning decisions can impact model performance [13, 14]. This kind of systematic investigation is needed for the loco-manipulation setting due to the lack of available datasets and the difficulty of evaluating VLA models in the real world [15].

To address this, we present HumanoidMimicGen, **an algorithm for humanoid legged loco-manipulation demonstration generation.** HumanoidMimicGen interleaves locomotion planning, arm planning, and skill demonstration adaptation to generate bimanual loco-manipulation demonstrations in new environments and states (see Fig. 1).

We propose a hybrid control space for humanoids, where legs are controlled by a reinforcement-learned policy and, in turn, adopt a decoupled approach that factors planning into lower- and upper-body motion, corresponding to segments that require dynamic and static stability.

To study data quality and policy training questions in the setting of loco-manipulation, we introduce a new simulation benchmark of humanoid factory tasks. We show that HumanoidMimicGen can generate high-quality data and train performant policies in this setting. Finally, we demonstrate that generating large simulation datasets with HumanoidMimicGen enables sim-and-real co-training for sample-efficient deployment on real-world humanoids.

II PREREQUISITES AND RELATED WORK

We seek to teach a bimanual legged humanoid robot to manipulate movable objects. We use the "Table-to-Shelf" task in Figure 1 as a running example. Here, a large box is randomly initialized on a table. The robot must walk to the box, pick it up with both hands, and move it to the shelf.

The robot has controllable joints $\mathcal{J} = J_l \cup J_t \cup J_{a_l} \cup J_{h_l} \cup J_{a_r} \cup J_{h_r}$ arising from its legs (J_l), torso (J_t), left & right arms (J_{a_l}, J_{a_r}), and left & right hands (J_{h_l}, J_{h_r}). The state of the robot is represented by configuration $q \in \mathbf{R}^{|\mathcal{J}|}$. Let $\mathcal{E} = \{e_l, e_r\}$ be the set of robot end-effector coordinate frames for the left and right arms, respectively. The state of each object in the world, for example, the box and shelf, is represented by the $\mathrm{SE}(3)$ pose of its coordinate frame $f \in \mathcal{F}$.

II-A Problem Statement

We model each humanoid loco-manipulation task as a Partially Observable Markov Decision Process with state space $\mathcal{S}$, observation space $\mathcal{O}$, action space $\mathcal{A}$, and initial state distribution $\mathcal{S}_0$, where $\operatorname{supp} \mathcal{S}_0 \subseteq \mathcal{S}$. States $s \in \mathcal{S}$ contain the joint positions of the robot ($s[\mathcal{J}] = q \in \mathbf{R}^{|\mathcal{J}|}$) as well as the world poses of the end-effector ($s[e] \in \mathrm{SE}(3)$) and object ($s[f] \in \mathrm{SE}(3)$) frames. Observations $o \in \mathcal{O}$ contain the robot's proprioception as well as images from one or more cameras on the robot. Actions $a \in \mathcal{A}$ are joint-space position controller targets $a[\mathcal{J}] \in \mathbf{R}^{|\mathcal{J}|}$ for joints $\mathcal{J}$; however, we modify this assumption in Section III-A. For notational simplicity and because the robot's end-effector poses are a function of its joint positions, let $a[e] \in \mathrm{SE}(3)$ give the world pose of end-effector $e \in \mathcal{E}$ derived from its target joint positions.

For each task, we are interested in generating a dataset of N demonstrations $D_N = \{d^1, ..., d^N\}$ where each demonstration $d^i = (\langle s^i_0, o^i_0, a^i_0 \rangle, ..., \langle s^i_{H_i}, o^i_{H_i}, a^i_{H_i} \rangle)$ is a sequence of state, observation, and action triplets. Using that dataset, we train Behavior Cloning [16, 17] policies $\pi : \mathcal{O} \to \mathcal{A}$ that map observations to actions. During data generation, we assume privileged state access, which is readily available in simulation, but we seek to train policies that only use observations.

II-B Data Generation via Demonstration Adaption

We are interested in generating demonstrations D_N that scale automatically with N. Like in MimicGen [7], we assume access to a small set of annotated source demonstrations $\mathcal{D}_0$ and seek to automatically amplify them into a much larger set D_N, where $N >> |\mathcal{D}_0|$, by adapting them to new initial states $s_0 \sim \mathcal{S}_0$. Rather than adapt demonstrations in their entirety, which were collected under a specific context involving the full world, we adapt short object-centric *skill* segments [18] that involve contact with one or more objects. Specifically, we segment each demonstration d into annotated skill demonstrations $\psi = \langle e, f, d^\psi \rangle$, where $e \in \mathcal{E}$ is an end-effector frame, $f \in \mathcal{F}$ is a reference object frame, and $d^\psi \subseteq d$ is a contiguous subsequence of d. Let $s^\psi_t = d^\psi[t, 0]$ and $a^\psi_t = d^\psi[t, 2]$ be the tth state and action on the skill demonstration. Let $\Psi = \{\psi_1, ..., \psi_*\}$ be the set of skills segmented from the same demonstration. In the Table-to-Shelf task, we define a skill per end-effector both for picking the box (f is the box) and for placing the box (f is the shelf).

Annotation enables skill demonstrations to be adapted to new states s' through rigid transformation. Specifically, we adapt all target end-effector poses $a^\psi[e]$ on actions $a^\psi \in d^\psi[:, 2]$, to a new state s' by computing: $a'[e] = s'[f] s^\psi_0[f]^{-1} a^\psi[e]$. Intuitively, this computes the target poses of end-effector e relative to the initial pose $s^\psi_0[f]$ of object frame f in the skill demonstration and then applies them to the current object frame pose $s'[f]$ in a spatially invariant manner, where, in the absence of external factors, they may produce the same effect.

Our strategy is to adapt and execute each skill $\psi \in \Psi$ to attempt to generate a successful demonstration. For single manipulators, these skills can be executed in sequence [7, 18]. However, for bimanual robots, we must simultaneously control multiple arms, where, at times, the arms must execute skills in parallel. Additionally, some skills need to be completed *prior* to others or started *concurrently* with others. Like in DexMimicGen [19], we represent this with a set of precedence order pairs $\mathcal{P} = \{\langle \psi_{i_1}, \psi_{i_2} \rangle, ...\}$ and concurrent order pairs $\mathcal{C} = \{\langle \psi_{j_1}, \psi_{j_2} \rangle, ...\}$, where $\langle \psi, \psi' \rangle \in \mathcal{P}$ indicates that skill ψ must be completed before skill ψ' is started and $\langle \psi, \psi' \rangle \in \mathcal{C}$ specifies that skills ψ, ψ' must start at the same time. Figure 4 (*left*) visualizes the precedence and coordination constraints applied to the four skills in the Table-to-Shelf task. Here, each arm must perform its pick before its place, and because dual-arm grasping is required due to the box's size, the start of both the pick and place must be coordinated.

III HumanoidMimicGen

Existing demonstration generation algorithms require that the robot's action space be in task space, where end-effector target poses are mapped to commands using, for example, Operational-Space Control (OSC) [20]. However, we seek to control legged humanoids and induce bipedal locomotion, a key challenge being maintaining stability during both navigation and manipulation. Doing so requires careful coordination of the whole body to statically balance and dynamically walk. As a result, we cannot effectively apply an independent task-space control strategy to all robot limbs, notably the legs.

In HumanoidMimicGen, we address this by first adopting a *hybrid* action space, where the upper body is governed by a joint-space controller, but the lower body is commanded by a Reinforcement Learning (RL) policy trained to control the velocity of the robot's pelvis. The RL policy can dynamically maintain stability, but not in a way that perfectly tracks velocity actions. To compensate for this, we take a *decoupled* approach to data generation, where we separate planning into static manipulation and dynamic locomotion phases. Additionally, we introduce a *whole-body* adaptation and planning scheme in order to replay demonstrated skills in a manner robust to changes in the environment.

III-A Humanoid Hybrid Action Space

We use a *hierarchical hybrid action space* for humanoid control [21]. Low-level joint commands are issued through a joint-space interface, while base motion and leg coordination are handled by a learned locomotion controller. At the lowest level, the torso, arm, and hand joints $J_t \cup J_{a_l} \cup J_{h_l} \cup J_{a_r} \cup J_{h_r}$ are directly commanded via joint position control. Leg joints J_l are not directly actuated by the policy. Above this interface, we expose a high-level action API with two components: (i)

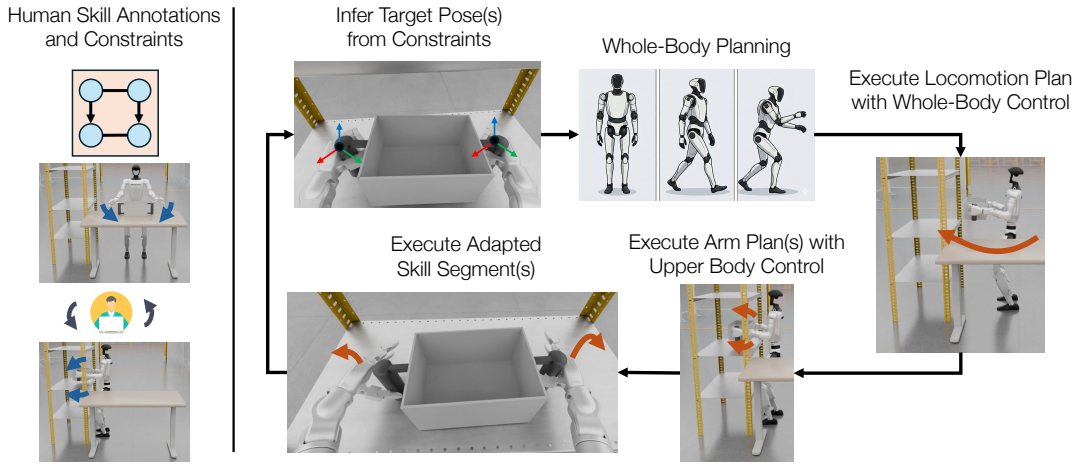

Fig. 2: **Method Overview.** *Left:* HumanoidMimicGen takes a source demonstration with per-arm skill annotations (blue arrows) and constraints (orange box, described in Sec. B) that govern the order of base and arm execution. *Right:* The skill annotations and constraints are used to determine an order for skill execution, and a target end-effector pose for the beginning of each skill. The Whole-Body Planning process constructs a decoupled locomotion plan and per-arm plan to achieve the target(s) (shown in orange arrows), which are then executed by the whole-body controller and upper-body controller, respectively. Finally, the adapted skill segment(s) are executed. This process repeats until there are no more skill segments to execute.

joint position commands for the upper body (arms, hands, torso) and (ii) base motion commands for locomotion.

We adopt the Homie [21] RL locomotion controller. It takes as input the current arm and torso configuration, together with a base command $a[l] = [\dot{x}, \dot{y}, \dot{\theta}, z]$, where $[\dot{x}, \dot{y}]$ specify planar base velocity, $\dot{\theta}$ specifies yaw rate, and z specifies the desired torso height. Given this input, the RL controller produces dynamically feasible leg joint position commands, which are forwarded to the low-level joint-space API. This hierarchy enables intuitive teleoperation and data generation while delegating balance, contact handling, and leg coordination to the learned locomotion controller.

III-B Whole-Body Data Generation

HumanoidMimicGen greedily processes applicable sets of skills Ψ_i in sequence. See Appendix B for details on how the full set of skills Ψ is serialized into a sequence of skill subsets $[\Psi_1, \Psi_2, ...]$, where $\Psi_i \subseteq \Psi$, that satisfies the precedence and coordination constraints in Section II-B. For each skill $\psi \in \Psi_i$, it computes an end-effector target pose $T[e]$ to start the skill and solves whole-body inverse kinematics to find a reachable configuration q'' that jointly achieves each active end-effector's target pose. Then, it decomposes planning into a locomotion subproblem to an intermediate configuration q', along with a stationary motion subproblem, ultimately to q''. Finally, it performs whole-body skill adaptation to replay the skill demonstrations in the current state.

Algorithm 1 in Appendix C displays the full pseudocode for HUMANOIDMIMICGEN; however, we walk through its operation here. It takes as input an initial state s_0, a set of skills Ψ, and a set of precedence partial orders on the skills $\mathcal{P}$. To generate a dataset D_N of N demonstrations, HUMANOIDMIMICGEN is called repeatedly until it returns successful N times, using sampled initial states $s_0 \sim \mathcal{S}_0$ and skills and partial order pairs $\langle \Psi, \mathcal{P} \rangle$ sampled uniformly at random from the source demonstrations.

HUMANOIDMIMICGEN iteratively plans and executes until all skills $\psi \in \Psi$ have been completed. On its ith iteration, it greedily selects skills $\Psi_i \in \Psi$ that do not have any partial order constraints involving another skill ψ' that has not been completed. For each skill $\psi \in \Psi_i$, using the current state s and the first state s_0^ψ on the active skill demonstrations, it adapts the demonstration pose $s_0^\psi[e]$ for end-effector e using the current pose $s[f]$ of object frame f, resulting in target end-effector pose $T[e]$. Next, it solves whole-body inverse kinematics to produce a batch of full configurations Q that reach $T[e]$ for each active end-effector. See Appendix D for full detail on our humanoid collision representation, whole-body inverse kinematics, and motion planning.

For each candidate target configuration $q'' \in Q$, it first computes a *switch* configuration q', formed from the upper joint positions of q and the lower joint positions of q'', where the robot switches from locomotion to manipulation. Then, it plans a locomotion trajectory τ_l between the current configuration q and the switch configuration q''. If the planning succeeds, HUMANOIDMIMICGEN executes the τ_l using the RL controller. If the planning fails, the next configuration in the batch is attempted. If the batch is exhausted, then the episode is declared a failure.

Because of the imperfect execution of the RL controller, we replace q' with the achieved switch configuration, plan a manipulation trajectory τ_m between the switch configuration q' and target configuration q'', and execute τ_m using the upper-body joint-space controller. The skill demonstration actions a^ψ for each skill $\psi \in \Psi_i$ are adapted by $s[f]$ into a skill trajectory τ_{Ψ_i} using subroutine ADAPT-SKILL-DEMOS (Algorithm 2 in Appendix C), which solves upper-body IK to track the adapted end-effector target poses $a^\psi[e]$. Figure 5 (see the Appendix) visualizes this process applied to a single-

TABLE I: **G1 Simulation Benchmark Results.** We compare policies trained on source human demonstrations, DexMimicGen+ [19]-generated datasets, and HumanoidMimicGen-generated datasets, from left to right.

Task	Src Demo	DexMimicGen+	Ours
Box Lift Floor	0.14	0.87	1.00
Push Button	0.18	0.26	1.00
Box Lift	0.95	0.68	0.98
Push Shelf Forward	0.70	0.35	0.93
Drill Lift	0.20	0.13	0.45
Drill PnP	0.08	0.13	0.17
Box Table To Shelf	0.04	0.17	0.52
Pick Drill From Holder	0.00	0.00	0.35
Drill Lift Obstacle	0.04	0.00	0.27
Average	0.26	0.16	**0.63**

arm pick skill on the "Drill Lift" task. Hand joint positions $a^{\psi}[J_h]$ are replayed without modification. After completing each skill in Ψ, the current state s is checked for task success using CHECK-SUCCESS.

IV Experiments

We evaluate HumanoidMimicGen on a new G1 loco-manipulation benchmark that we developed (Appendix E). We describe the experimental setup in Appendix F.

IV-A HumanoidMimicGen Capabilities

HumanoidMimicGen boosts policy success rates over baseline data generation methods and the source demonstrations only. We compare HumanoidMimicGen against policies trained purely on the single source human demonstration, as well as a DexMimicGen+ baseline. The DexMimicGen+ baseline involves extending DexMimicGen [19] so that it outputs velocity actions for the RL-based locomotion controller whenever it performs task-space control. Like in Jiang *et al.* [19], DexMimicGen+ does not perform any skill reasoning, planning, or collision checking. Instead, it directly adapts full demonstrations greedily. As seen in Table I, HumanoidMimicGen consistently improves policy performance over both the single-source demonstration baseline and DexMimicGen+ across all tasks. Averaged over nine tasks, HumanoidMimicGen increases policy performance from 16.3% (DexMimicGen+) to 58.7%. These results show that HumanoidMimicGen can reliably transform a single demonstration into diverse, high-quality training data, whereas direct imitation and prior data-generation baselines fail to generalize over varying states.

HumanoidMimicGen can be applied to long-horizon loco-manipulation problems. HumanoidMimicGen successfully generates demonstrations for tasks that require coordinated navigation and whole-body manipulation over longer periods of time. Policies trained on HumanoidMimicGen datasets maintain high success rates on these long-horizon tasks. For example, HumanoidMimicGen achieves 0.93 PSR on PushShelfForward (1230 steps) and 1.00 on BoxLiftFloor (900 steps), compared to 0.35 and 0.87 with DexMimicGen+, and 0.70 and 0.14 with source demonstrations alone. We provide further analysis in the Appendix H.

IV-B Real-World Evaluation

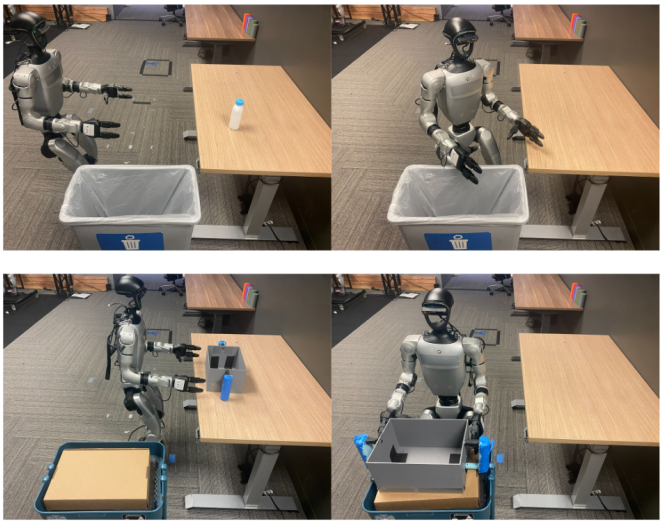

Fig. 3: **Real-World Deployment.** Policies trained on HumanoidMimicGen-generated simulation data and co-trained with real-world demonstrations outperform real-only training on two tasks. **Top**: ThrowBottle policy performance improves from 0.60 to 0.75 when co-training with simulation-generated data. **Bottom**: BoxToCart policy performance increases from 0.35 to 0.60. On average, sim-and-real co-training improves policy performance from 0.48 to 0.68.

We evaluate HumanoidMimicGen on a physical G1 humanoid robot using sim-and-real co-trained policies [9] and compare with policies trained only on real robot data. Each policy is evaluated on 10 episodes. We use a Luxonis OAK-D camera for image observations. For the upper body joints, the control frequency is 25 Hz; for the lower body policy, inference is at 50 Hz. We use interpolation for the lowest-level position control API, which runs at 200 Hz.

Sim-and-real co-training. For each task, we collect one simulation demonstration and generate 500 more using HumanoidMimicGen. Policies are co-trained on HumanoidMimicGen generated and 30 real robot demonstrations.

Tasks. We evaluate flow-matching policies trained from scratch on a **BoxToCart** and **ThrowBottle** task as seen in Figure 3. **BoxToCart** involves picking up a box with two hands, walking to a cart, then placing the box on the cart. **ThrowBottle** involves walking to a table, picking up a bottle, turning the waist, and throwing it into the bin. Scoring criteria for **BoxToCart**: 0.5 for lifting box, 0.25 for walking to cart, 0.25 for placing on cart. Scoring criteria for **ThrowBottle**: 0.5 for lifting bottle, 0.5 for throwing bottle into bin.

Results. As seen in Figure 3, on **ThrowBottle**, the policy trained on both simulation and real-world data outperforms that trained purely on real-world data by 0.15. On **BoxToCart**, the sim-and-real co-trained policy outperforms the real data policy by 0.3. These results show that HumanoidMimicGen data boosts real-world loco-manipulation policy performance.

References

[1] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge Data: Boosting Generalization of Robotic Skills with Cross-Domain Datasets. In *Robotics: Science and Systems*, 2022.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE Int'l Conf on Robotics and Automation (ICRA)*, 2024.

[6] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[7] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=dk-2R1f_LR.

[8] Murtaza Dalal, Ajay Mandlekar, Caelan Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox. Imitating task and motion planning with visuomotor transformers. *arXiv preprint arXiv:2305.16309*, 2023.

[9] Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, et al. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. *arXiv preprint arXiv:2503.24361*, 2025.

[10] Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, and Russ Tedrake. Empirical analysis of sim-and-real cotraining of diffusion policies for planar pushing from pixels. *arXiv preprint arXiv:2503.22634*, 2025.

[11] Shuo Cheng, Liqian Ma, Zhenyang Chen, Ajay Mandlekar, Caelan Garrett, and Danfei Xu. Generalizable domain adaptation for sim-and-real policy co-training. *arXiv preprint arXiv:2509.18631*, 2025.

[12] Siddhant Haldar, Lars Johannsmeier, Lerrel Pinto, Abhishek Gupta, Dieter Fox, Yashraj Narang, and Ajay Mandlekar. Point bridge: 3d representations for cross domain policy learning. *arXiv preprint arXiv:2601.16212*, 2026.

[13] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

[14] Vaibhav Saxena, Matthew Bronars, Nadun Ranawaka Arachchige, Kuancheng Wang, Woo Chul Shin, Soroush Nasiriany, Ajay Mandlekar, and Danfei Xu. What matters in learning from large-scale datasets for robot manipulation. *arXiv preprint arXiv:2506.13536*, 2025.

[15] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.

[16] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[17] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine intelligence 15*, pages 103–129, 1995.

[18] Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillgen: Automated demonstration generation for efficient skill learning and deployment. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=YOFrRTDC6d.

[19] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. *arXiv preprint arXiv:2410.24185*, 2024.

[20] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[21] Qingwei Ben, Feiyu Jia, Jia Zeng, Junting Dong, Dahua Lin, and Jiangmiao Pang. Homie: Humanoid loco-manipulation with isomorphic exoskeleton cockpit. *arXiv preprint arXiv:2502.13013*, 2025.

[22] Lujie Yang, HJ Suh, Tong Zhao, Bernhard Paus Graesdal, Tarik Kelestemur, Jiuguang Wang, Tao Pang, and Russ Tedrake. Physics-driven data generation for contact-rich manipulation via trajectory optimization. *arXiv preprint arXiv:2502.20382*, 2025.

[23] Kevin Lin, Varun Ragunath, Andrew McAlinden, Aaditya Prasad, Jimmy Wu, Yuke Zhu, and Jeannette Bohg. Constraint-preserving data generation for visuomotor policy generalization. In *9th Annual Conference on Robot Learning*, 2025. URL https://openreview.net/forum?id=KSKzA1mwKs.

[24] Zihan Zhou, Animesh Garg, Ajay Mandlekar, and Caelan Garrett. Reinforcegen: Hybrid skill policies with automated data generation and reinforcement learning. *arXiv preprint arXiv:2512.16861*, 2025.

[25] Chengshu Li, Mengdi Xu, Arpit Bahety, Hang Yin, Yunfan Jiang, Huang Huang, Josiah Wong, Sujay Garlanka, Cem Gokmen, Ruohan Zhang, et al. Momagen: Generating demonstrations under soft and hard constraints for multi-step bimanual mobile manipulation. *arXiv preprint arXiv:2510.18316*, 2025.

[26] Minghuan Liu, Zhengbang Zhu, Xiaoshen Han, Peng Hu, Haotong Lin, Xinyao Li, Jingxiao Chen, Jiafeng Xu, Yichu Yang, Yunfeng Lin, et al. Manipulation as in simulation: Enabling accurate geometry perception in robots. *arXiv preprint arXiv:2509.02530*, 2025.

[27] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[28] Michael James McDonald and Dylan Hadfield-Menell. Guided imitation of task and motion planning. In *Conference on Robot Learning*, pages 630–640. PMLR, 2022.

[29] Ajay Mandlekar, Caelan Garrett, Danfei Xu, and Dieter Fox. Human-in-the-loop task and motion planning for imitation learning. In *7th Annual Conference on Robot Learning*, 2023.

[30] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3, 1999.

[31] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. *Proceedings 2002 IEEE Int'l Conf on Robotics and Automation*, 2, 2002.

[32] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The Int'l Journal of Robotics Research*, 2023.

[33] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot programming by demonstration. In *Springer Handbook of Robotics*, 2008.

[34] Sylvain Calinon, Florent D'halluin, Eric L. Sauser, Darwin G. Caldwell, and Aude Billard. Learning and reproduction of gestures by imitation. *IEEE Robotics and Automation Magazine*, 17, 2010.

[35] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[36] Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. *arXiv preprint arXiv:2511.16651*, 2025.

[37] Chenghao Yin, Da Huang, Di Yang, Jichao Wang, Nanshu Zhao, Chen Xu, Wenjun Sun, Linjie Hou, Zhijun Li, Junhui Wu, et al. Genie sim 3.0: A high-fidelity comprehensive simulation platform for humanoid robot. *arXiv preprint arXiv:2601.02078*, 2026.

[38] Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, et al. Hover: Versatile neural whole-body controller for humanoid robots. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9989–9996. IEEE, 2025.

[39] Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Sirui Chen, Fernando Castañeda, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, et al. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *arXiv preprint arXiv:2511.07820*, 2025.

[40] Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C Karen Liu, Rocky Duan, and Guanya Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction. *arXiv preprint arXiv:2509.26633*, 2025.

[41] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119, 2023. doi: 10.1109/ICRA48891.2023.10160765.

[42] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, Nathan Ratliff, and Dieter Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[43] Masatomo Inui, Nobuyuki Umezu, and Ryohei Shimane. Shrinking sphere: A parallel algorithm for computing the thickness of 3d objects. *Computer-Aided Design and Applications*, 13(2):199–207, 2016.

[44] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

[45] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[46] Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow:

Imitation learning with variance-adaptive flow-based policies. *arXiv preprint arXiv:2402.04292*, 2024.

APPENDIX

Overview

The appendix contains the following content.

- **Extended Related Work** (Appendix A): extended discussion of prior work on automated data generation, imitation learning, and humanoid loco-manipulation.
- **Skill Planning** (Appendix B): details on precedence and coordination constraints used for skill planning.
- **Pseudocode** (Appendix C): system-level pseudocode for HumanoidMimicGen and whole-body skill adaptation.
- **Motion Planning Details** (Appendix D): collision representation and whole-body inverse kinematics details.
- **Loco-Manipulation Benchmark** (Appendix E): description of the humanoid loco-manipulation simulation benchmark and tasks.
- **Experimental Setup** (Appendix F): source demonstrations, data generation, and policy training setup.
- **DexMimicGen+ baseline** (Appendix G): implementation details for the DexMimicGen+ baseline.
- **HumanoidMimicGen Analysis** (Appendix H): analysis of motion noise, initialization, and policy architecture.
- **Comparison G1 and G1 Floating** (Table IV): comparison between legged and floating-base embodiments.
- **Policy training details** (Appendix J): VLA training and evaluation details.
- **Raw tables for bar plots** (Appendix K): raw tables corresponding to ablation plots.

A Extended Related Work

Automated Data Generation. MimicGen [7] introduced a data generation method for generating large-scale datasets by adapting a small set of expert demonstrations to new object poses. This method has been extended to support stationary bimanual dexterous manipulation [19] and incorporate planning and optimization techniques [18, 22, 23] and Reinforcement Learning (RL) [24] to improve data quality. MoMaGen [25] applied planning and skill adaptation [18] to mobile manipulation and incorporated soft visibility constraints to aid visuomotor policy learning. WBCMimicGen [26] also addressed mobile manipulation, but focused on a quadratic programming approach to improving its tracking controller. Task and Motion Planning (TAMP) [27] algorithms have been used for data generation [28, 8, 29]; but, they require a planning model, which is prohibitive for contact-rich tasks. These approaches all rely on the assumption of stable and independent end-effector control per limb, which does not transfer to legged humanoids, where careful whole-body coordination is required during both locomotion and manipulation.

Learning Manipulation from Demonstrations. Behavior Cloning (BC) [16, 30, 31, 13, 32] is a prominent method for training policies from demonstrations via supervised learning. This method has proved to be very effective for robot manipulation [33, 34, 2, 6, 35, 1], its success relies on having large-scale high-quality datasets to learn from. Recent evidence has shown that synthetic simulation datasets can supplement or even replace real-world human demonstrations to produce high-performing real-world manipulation policies [9, 10, 1, 11, 36, 37, 12]. In our work, we leverage sim-and-real co-training to train real-world policies.

Loco-Manipulation. We build on recent work enabling humanoid loco-manipulation control using teleoperation systems combined with reinforcement-learning-based control. Homie [21] and HOVER [38] develop whole-body control frameworks that integrate locomotion and manipulation, enabling stable execution of loco-manipulation behaviors under learned or hybrid controllers. SONIC [39] scales up control to work for a diverse range of loco-manipulation behaviors and poses. OmniRetarget [40] trains locomanipulation policies using reinforcement learning and deploys them in the real world. We focus on imitation learning to avoid the need for reward tuning.

B Skill Planning

From Section II-B, we assume that a set of skills Ψ for demonstration d is annotated with precedence $\mathcal{P}$ and coordination constraints $\mathcal{C}$. Because each end-effector e can only perform a single skill at a time, we assume that the skills per end-effector are totally ordered in $\mathcal{P}$, which intuitively means at most $|E| = 2$ skills can be active at a time.

Coordination constraints tie the start of two skills together. Because of this, we can reduce the coordination constraint $\langle \psi, \psi' \rangle$ to additional precedence constraints by transitively applying the precedence constraints for ψ to ψ' and vice versa:

$$\mathcal{P} \leftarrow \mathcal{P} \cup \{\langle \psi^*, \psi' \rangle \in \Psi^2 \mid \exists \psi \in \Psi . \langle \psi, \psi' \rangle \in \mathcal{C} \wedge \langle \psi^*, \psi \rangle \in \mathcal{P}\}.$$

Afterward, the space of skill plans can be represented as a Directed Acyclic Graph (DAG) $\langle \Psi, \mathcal{P} \rangle$, where vertices are skills $\psi \in \Psi$ and directed edges $\langle \psi, \psi' \rangle \in \mathcal{P}$ are precedent constraints. Figure 4 (*right*) visualizes the reduction from coordination constraints to additional precedence constraints.

In Section C, we use a greedy algorithm to iteratively produce a set of skills $\Psi_i \subseteq \Psi$ on-demand to execute next. After planning skills Ψ_{i-1}, we remove them from Ψ and plan skills Ψ_i by selecting the remaining skills $\psi \in \Psi$ free of precedence constraints involving other skills $\psi' \in \Psi$:

$$\Psi_i \leftarrow \{\psi \in \Psi \mid \neg \exists \psi' \in \Psi . \langle \psi', \psi \rangle \in \mathcal{P}\}.$$

This procedure resembles computing a topological sort of DAG $\langle \Psi, \mathcal{P} \rangle$ online and grouping incomparable skills to be executed together. In the Table-to-Shelf task, this manifests as two iterations, where the two pick skills are executed as a part of Ψ_1, and the two place skills are executed as part of Ψ_2.

C Pseudocode

Algorithms 1 and 2 give the pseudocode for HUMANOIDMIMICGEN and ADAPT-SKILL-DEMOS respectively, the core HumanoidMimicGen algorithms in Section III-B.

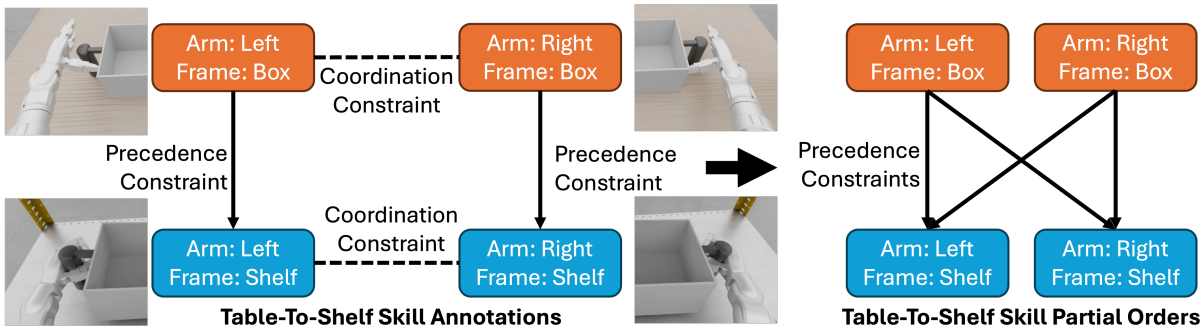

Fig. 4: **Precedence and Coordination Constraints.** A human annotates precedence and coordination constraints among the skills. These constraints are automatically compiled into partial orders on the skills, defining legal execution orders.

Algorithm 1 HumanoidMimicGen Pseudocode

procedure HUMANOIDMIMICGEN($s_0, \langle \Psi, \mathcal{P} \rangle$)
  $s \leftarrow$ **copy**(s_0) ▷ Current state
  while $|\Psi| \neq 0$:
    $\Psi_i \leftarrow \{\psi \in \Psi \mid \neg\exists \psi' \in \Psi . \langle \psi', \psi \rangle \in \mathcal{P}\}$
    $\Psi \leftarrow \Psi \setminus \Psi_i$ ▷ Subtract current skills Ψ_i from Ψ
    $T \leftarrow \{\ \}$ ▷ End-effector pose targets
    for $\langle e, f, d^\psi \rangle \in \Psi_i$: ▷ Skill ψ for end-effector e
      $\langle s_0^\psi, \_, \_ \rangle \leftarrow d^\psi[0]$ ▷ Reference state s_0^ψ
      $T[e] \leftarrow s[f] s_0^\psi[f]^{-1} s_0^\psi[e]$ ▷ Target pose for e
    $q \leftarrow s[\mathcal{J}]$ ▷ Current configuration
    $Q \leftarrow$ WHOLE-INV-KINEMATICS$(\mathcal{J}, T, s)$
    for $q'' \in Q$: ▷ IK configurations batch
      $q' \leftarrow$ **copy**(q) ▷ Target switch configuration
      $q'[J_l] \leftarrow q''[J_l]$ ▷ Set the locomotion joints
      $\tau_l \leftarrow$ PLAN-MOTION(J_l, q, q', s) ▷ Legs
      if $\tau_l \neq$ **None**:
        $s \leftarrow$ CONTROL-LOCOMOTION(τ_l)
        break ▷ Planning success
    else
      return False ▷ Planning failure
    $q' \leftarrow s[\mathcal{J}]$ ▷ Achieved switch configuration
    $\tau_m \leftarrow$ PLAN-MOTION$(J_t \cup J_{a_l} \cup J_{a_r}, q', q'', s)$ ▷ Arms
    $s \leftarrow$ CONTROL-MANIPULATION(τ_m)
    $\tau_{\Psi_i} \leftarrow$ ADAPT-SKILL-DEMOS(s, Ψ_i)
    $s \leftarrow$ CONTROL-MANIPULATION(τ_{Ψ_i})
  return CHECK-SUCCESS(s) ▷ Execution success

Algorithm 2 Whole-Body Skill Adaptation Pseudocode

procedure ADAPT-SKILL-DEMOS(s, Ψ_i)
  $\tau \leftarrow [\ ]$ ▷ Adapted skill trajectory
  while True:
    $T \leftarrow \{\ \}$ ▷ End-effector pose targets
    for $\langle e, f, d^\psi \rangle \in \Psi_i$: ▷ Skill ψ for end-effector e
      $\langle s_0^\psi, \_, \_ \rangle \leftarrow d^\psi[0]$ ▷ Reference state s_0^ψ
      if $|\tau| \leq |d^\psi|$:
        $\langle \_, \_, a^\psi \rangle \leftarrow d^\psi[|\tau|]$ ▷ Skill demo action a^ψ
        $T[e] \leftarrow s[f] s_0^\psi[f]^{-1} a^\psi[e]$ ▷ Adapt a to s
    if $|T| = 0$: ▷ All skill actions processed
      return τ_{Ψ_i}
    $q' \leftarrow$ WHOLE-INV-KINEMATICS$(J_t \cup J_{a_l} \cup J_{a_r}, T, s)[0]$
    $\tau \leftarrow \tau + [q']$ ▷ Add the next waypoint q

D *Motion Planning Details*

We adapt skills to new environments and states by solving for new robot configurations that satisfy the same constraints as the demonstrations, while responding to changes in pose and avoiding collisions. We develop our algorithms on top of cuRobo [41, 42] to take advantage of GPU-accelerated collision checking and batch inverse kinematics.

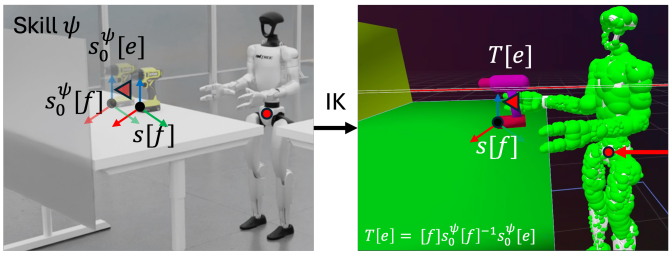

Fig. 5: **Whole-Body Skill Adaptation.** *Left:* skill ψ for end-effector e relative to object frame f is adapted to a new state s. *Right*: spherical collision representation used for planning and IK configuration q'' for adapted skill target pose $T[e]$.

Humanoid Collision Representation. To leverage GPU-accelerated collision checking, we represent the robot and all moving objects as a set of spheres with varying radii. Instead of manually placing these spheres per robot [41, 42], which is challenging for humanoids due to, for example, the intricacy of their dexterous hands, we take an automated approach. For each mesh on the robot or object, we sample a large set of points on its surface. For each sampled point, we create a candidate sphere by solving for the minimum volume sphere that is tangent to the sampled point and another face on the mesh [43]. We then inflate the sphere radii by a hyperparameter $\epsilon \approx 0.01m$ to cover more of the mesh's surface. Finally, we use a greedy combinatorial optimization algorithm to select a set of N spheres that maximizes coverage of the mesh's surface subject to a computation budget. Figure 5 (*right*) displays a set of collision spheres automatically computed for the G1 robot. To model collisions with held objects, we detect which movable objects are in contact with a robot end-effector then lazily compute a sphere decomposition for them.

Representing the robot as a set of spheres over-approximates its geometry. Because we consider inverse kinematics and motion planning subproblems induced through manipulation, the initial configuration and goal end-effector poses are often in contact with manipulable objects, which are collision states causing the subproblem to be infeasible, especially due to sphere over-approximation. However, intuitively, the robot can safely make contact between certain pairs of robot and object bodies. To account for this, we compute the spheres currently in collision before each IK or planning call and shrink them until they are no longer in collision. We do the same for target end-effector poses, but only for the robot links in the same rigid (given the planning joints) connected component of the kinematic tree as the end-effector.

Whole-Body Inverse Kinematics. We leverage cuRobo to implement WHOLE-INV-KINEMATICS, which performs multi-link batch inverse kinematics. Sometimes the robot can reach the start of the next skill without moving its legs. In such cases, it is preferable to skip locomotion planning, saving computation time, avoiding the use of both control modalities, and reducing the risk of control errors during dynamic walking. Additionally, because torso motion affects both end-effectors, it is desirable to minimize unneeded torso movement away from the default torso posture. To do this, we minimize the L_0 distance from the current configuration $||q'' - q||_0$, weighted by joint group. We achieve this by performing an iterative optimization, where we lock a subset of the joints $J \subseteq \mathcal{J}$ to have their current positions $q[J]$. In our experiments, we considered the free joint order $[J_a, J_a \cup J_t, J_a \cup J_l, \mathcal{J}]$, which corresponds to allowing 1) arm motion, 2) arm and torso motion, 3) arm and leg motion, and 4) free motion.

E Loco-Manipulation Benchmark

We developed a humanoid **loco-manipulation benchmark** built on robosuite [44] and MuJoCo [45]. This benchmark specifically tests loco-manipulation; success critically depends on accurate base motion and whole-body coordination. This benchmark enables controlled comparisons of data generation strategies, policy architectures, and embodiment choices in loco-manipulation settings, as analyzed in Section IV. We will **open source** this benchmark along with downloadable HumanoidMimicGen datasets after double-blind review.

The benchmark contains **9 tasks** involving a simulated G1 humanoid robot. Tasks vary along three axes: (i) the extent of required base motion (from minimal repositioning along one axis to multi-stage navigation), (ii) object interaction complexity (single-arm, bimanual, and whole-body), and (iii) execution horizon. Each task specifies a goal represented as a binary success condition. Initial states are generated by randomly sampling object poses and the robot's root pose. Together, these tasks emphasize the locomotion–manipulation interface: small errors in foot placement directly impact reachability.

1) **Box Lift Floor**: Grasp the box from the floor and lift to a target height.
2) **Push Button**: Approach an industrial panel and press its button.
3) **Box Lift**: Approach the table, grasp the box, and lift to a target height.
4) **Push Shelf Forward**: Push the shelving cart into a marked target zone.
5) **Drill Lift**: Approach table, grasp drill, and lift to target height.
6) **Drill PnP**: Pick a drill from one table and place it on a second table.
7) **Box Table To Shelf**: Transfer a box from a table into a shelf.
8) **Pick Drill From Holder**: Extract a drill from a holder and lift it to a target height.
9) **Drill Lift Obstacle**: Navigate around a blocking shelf to reach the table, then grasp and lift the drill to a target height.

In Table II, we further describe the tasks in the G1 Loco-manipulation Benchmark according the attributes they contain.

TABLE II: Capabilities tested by each benchmark task.

Task	Loco	Nav	1-arm	2-arm	Vert	Contact	Long
PushButton	**X**		**X**				
DrillLift	**X**		**X**				
BoxLift	**X**			**X**			
BoxLiftFloor	**X**			**X**	**X**		
PickDrillFromHolder	**X**		**X**		**X**	**X**	
PushShelfForward	**X**			**X**		**X**	
DrillLiftObstacle	**X**	**X**	**X**				**X**
DrillPnP	**X**		**X**				**X**
BoxTableToShelf	**X**			**X**	**X**		**X**

Capabilities: **Loco** = locomotion; **Nav** = obstacle-free navigation; **1-arm** = single-arm manipulation; **2-arm** = bimanual coordination; **Vert** = vertical reach; **Contact** = contact-rich interaction; **Long** = long-horizon execution.

F Experimental Setup

Source demonstrations. For each task, we collect a *single* human demonstration using a Pico VR controller for whole-body teleoperation. We obtain end-effector commands from controller poses, gripper commands using controller triggers, and navigation commands with the controller thumbsticks. Note that the VR head-up display is not used, though the headset is used to track the controller poses. The same teleoperation interface is used in simulation and the real world.

Data generation. Starting from the single human demonstration, HumanoidMimicGen generates 1,000 successful loco-manipulation trajectories per task in simulation. Data generation includes randomized initial robot base poses and object poses, subject to task feasibility constraints.

Policy learning. We train visuomotor policies via imitation learning using a flow-matching transformer policy [1, 46] trained from scratch. Policies take RGB observations from an ego-view camera with a resolution of (224, 224, 3) as well as proprioceptive state information. and output actions which are passed to the Homie [21] RL loco-manipulation controller as specified in Section III-A Unless stated otherwise, we use the flow-matching policy for our simulation and real-world experiments by default. For a given dataset, we train the policy for 25K steps, evaluate checkpoints on 100 episodes, and take the best-performing checkpoint.

G DexMimicGen+ Baseline

As described in Section IV-A, we compare HumanoidMimicGen against a DexMimicGen+ baseline in order to highlight the importance of intelligent whole-body planning and adaptation. The DexMimicGen+ baseline extends DexMimicGen [19] with the ability to handle locomotion when needed, as the original work solely focused on stationary bimanual manipulation. DexMimicGen+ inherits the same action space that we propose for HumanoidMimicGen (Section III-A),

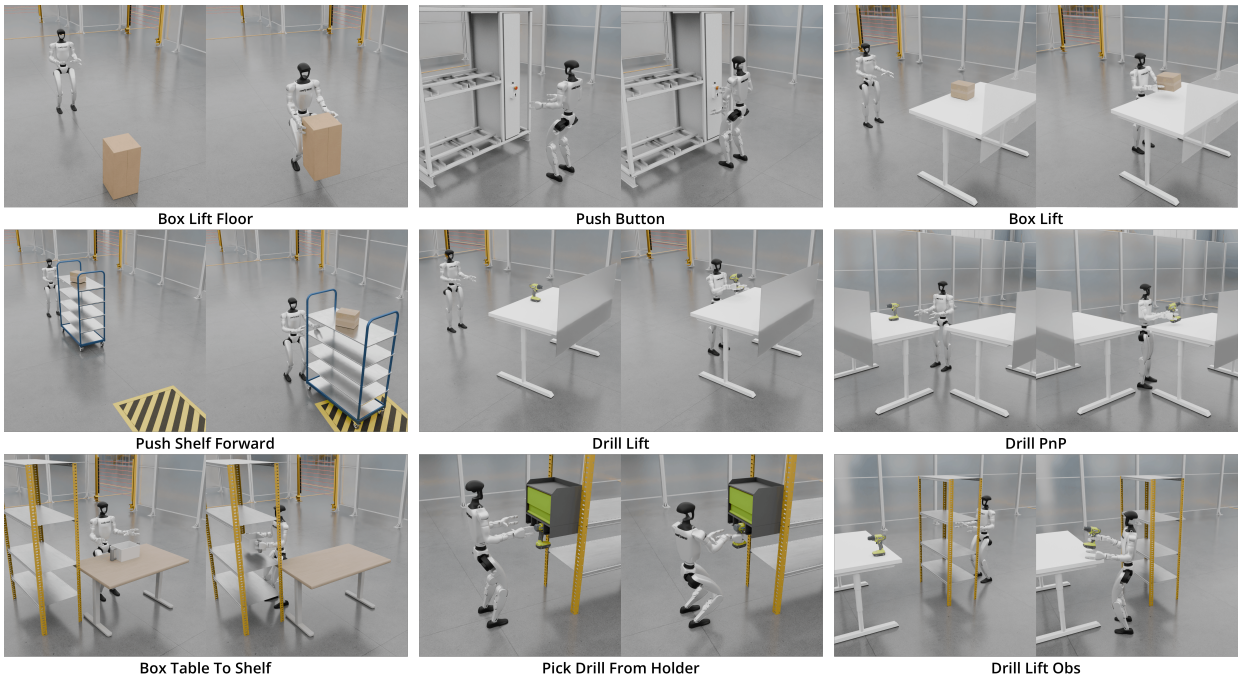

Fig. 6: **G1 Loco-Manipulation Benchmark.** We introduce a new simulation benchmark with nine tasks requiring robot loco-manipulation, along with HumanoidMimicGen-generated datasets. For each task, we provide two images to illustrate a randomly sampled initial scene configuration (*left*) and a task-completion configuration (*right*).

where locomotion actions are represented as x, y, θ velocity commands along with an additional z height command. No motion planning is used for this baseline, to stay consistent with the original work. We make the following modifications to the DexMimicGen algorithm:

1) We manually annotate source demonstration subtasks that will require locomotion.
2) For each subtask that requires locomotion, we invoke WHOLE-INV-KINEMATICS to infer a target base pose for one (or both) arm poses at the start of each subtask. The WHOLE-INV-KINEMATICS procedure does not consider collisions, and all joints apart from the legs are unlocked and free to move from the current configuration.
3) To move from a current base configuration to a new base configuration, a straight-line interpolated path is used (similar to interpolation segments for the arms in DexMimicGen [19]).

This baseline lacks several crucial features introduced by HumanoidMimicGen, including the use of skill reasoning, motion planning for locomotion and arm movement, and collision checking.

H HumanoidMimicGen Analysis

Effect of motion noise and robot pose initialization randomization. Figure 7 compares datasets generated with synthetic motion noise and randomized robot base initialization against counterparts without these perturbations. Removing motion noise reduces mean success from 0.58 to 0.49, while fixing the initial robot pose reduces success from 0.58 to 0.51. These degradations are consistent across most tasks, indicating that both trajectory-level and state-level randomization improve robustness. These results show that expanding robot-centric state coverage through noise and initialization randomization is a key factor for boosting policy success rates.

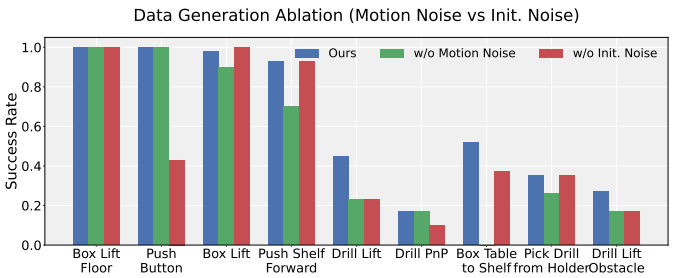

Fig. 7: **Effect of data generation design on policy success rates.** We compare policies trained on datasets generated by HumanoidMimicGen (*Ours*), without motion noise, and without robot pose initialization noise. Removing motion noise reduces the average success from 0.63 to 0.49, while removing initialization noise reduces it to 0.51. All variants use the same policy architecture and training budget; only the data generation procedure differs.

Effect of different policy architectures trained on HumanoidMimicGen datasets. We compare a flow-matching transformer policy to a diffusion policy [32] on the G1 loco-manipulation benchmark. All policies are trained on 1000 successful demonstrations generated by HumanoidMimicGen and evaluated at their best-performing checkpoint over 100 episodes. As shown in Figure 8, flow-matching outperforms

diffusion policy (0.63 vs. 0.40 average PSR).

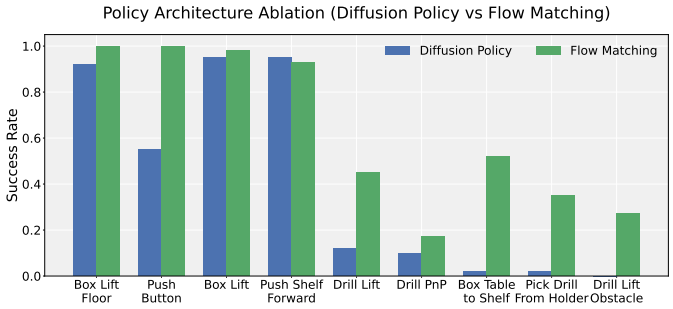

Fig. 8: **Policy Architecture Ablation.** Comparison of Diffusion Policy and Flow Matching policies trained on 1,000 demonstrations generated by HumanoidMimicGen and evaluated using the best-performing checkpoint. Flow Matching achieves a higher average success (0.63 vs. 0.40).

I Comparison G1 and G1 Floating

Scaling across embodiments. While our main results focus on the G1 humanoid, HumanoidMimicGen also applies to alternative embodiments with different locomotion and manipulation constraints. We demonstrate data generation on a floating-base humanoid variant using the same pipeline without algorithmic changes and present policy success rates on generated data in Table IV. Policies trained with and evaluated on the floating body humanoid consistently outperforms those in the legged setting, highlighting the challenge of policy learning in the legged loco-manipulation setting.

TABLE III: **Policy architecture ablation.** All policies are trained on 1,000 demonstrations generated by HumanoidMimicGen and evaluated using the best-performing checkpoint.

Task	VLA	DP	Flow Matching
Box Lift Floor	1.00	0.92	1.00
Push Button	0.80	0.55	1.00
Box Lift	1.00	0.95	0.98
Push Shelf Forward	0.98	0.95	0.93
Drill Lift	0.73	0.12	0.45
Drill PnP	0.70	0.10	0.17
Box Table to Shelf	0.63	0.02	0.52
Pick Drill From Holder	0.02	0.02	0.35
Drill Lift Obstacle	0.87	0.00	0.27
Average	**0.75**	0.40	0.63

J Policy training details

We additionally include results from post-training a vision-language action model.

Training Configurations The VLA model in our policy architecture ablation table (Table III) is initialized from the pre-trained GR00T N1.6 base model [1]. For post-training of the VLA, we use a learning rate of 1e-5 with a global batch size of 128 and train for 25,000 optimization steps. The policy uses a single egocentric camera as visual input with an image resolution of 640x480. Training is performed on a single node equipped with eight NVIDIA H100 GPUs.

TABLE IV: Embodiment comparison under the Humanoid-MimicGen pipeline. Policy success rate (PSR). Floating-body variants consistently outperform legged G1.

Task	G1	G1 w/o Legs
Box Lift Floor	1.00	1.00
Push Button	1.00	1.00
Box Lift	0.87	1.00
Push Shelf Forward	0.77	0.53
Drill Lift	0.45	1.00
Drill PnP	0.17	0.57
Box Table to Shelf	0.37	1.00
Pick Drill from Holder	0.35	1.00
Drill Lift Obstacle	0.27	1.00
Average	0.63	**0.90**

Evaluation Details During evaluation, each task is executed for 100 rollout trials to obtain statistically reliable performance metrics. Model checkpoints are saved every 5,000 training steps, and evaluation is performed across all saved checkpoints. The final reported results correspond to the checkpoint that achieves the best evaluation performance. Both training and evaluation are conducted on the same hardware platform to ensure consistency across experimental settings.

K Raw tables for bar plots

We additionally provide raw tables for the bar plots data generation ablations in Table V and policy architecture ablations in Table III.

TABLE V: Effect of data generation design on policy success rates. All variants use the same policy architecture and training budget; only the data generation procedure differs.

Task	Ours	w/o Motion Noise	w/o Init. Noise
Box Lift Floor	1.00	1.00	1.00
Push Button	1.00	1.00	0.43
Box Lift	0.98	0.90	1.00
Push Shelf Forward	0.93	0.70	0.93
Drill Lift	0.45	0.23	0.23
Drill PnP	0.17	0.17	0.10
Box Table to Shelf	0.52	0.00	0.37
Pick Drill from Holder	0.35	0.26	0.35
Drill Lift Obstacle	0.27	0.17	0.17
Average	**0.63**	0.49	0.51