2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Switch: Learning Agile Skills Switching for Humanoid Robots

Yuen-Fui Lau*, Qihan Zhao*, Yinhuai Wang*, Runyi Yu, Hok Wai Tsui
Qifeng Chen†, Ping Tan†
The Hong Kong University of Science and Technology

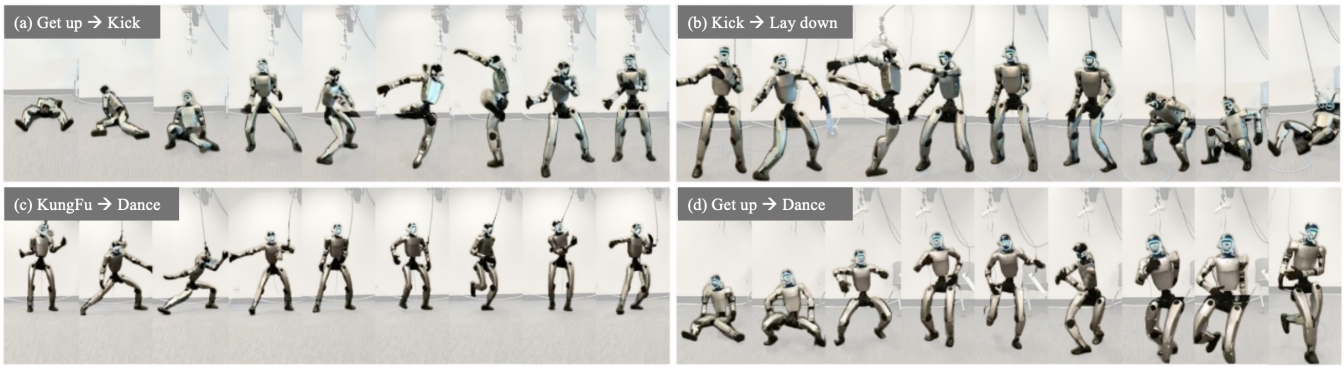

Fig. 1: We introduce Switch, a hierarchical whole-body control system that enables humanoid robots to perform agile and seamless skill switching between highly dynamic skills: (a) Get up to Kick; (b) Kick to Lay down; (c) Kungfu to Dance; (d) Get up to Dance.

***Abstract*— Recent advancements in whole-body control through deep reinforcement learning have enabled humanoid robots to achieve remarkable progress in real-world challenging locomotion skills. However, existing approaches often struggle with flexible transitions between distinct skills, creating safety concerns and practical limitations. To address this challenge, we introduce a hierarchical multi-skill system, Switch, enabling seamless skill transitions at any moment. Our approach comprises three key components: (1) a Skill Graph (SG) that establishes potential cross-skill transitions based on kinematic similarity within multi-skill motion data, (2) a whole-body tracking policy trained on this skill graph through deep reinforcement learning, and (3) an online skill scheduler to drive the tracking policy for robust skill execution and smooth transitions. For skill switching or significant tracking deviations, the scheduler performs online graph search to find the optimal feasible path, which ensures efficient, stable, and real-time execution of diverse locomotion skills. Comprehensive experiments demonstrate that Switch empowers humanoid to execute agile skill transitions with high success rates while maintaining strong motion imitation performance.**

I. Introduction

Recent advances in whole-body motion control, powered by simulation and reinforcement learning, have enabled humanoid robots to perform dynamic real-world skills such as highly agile acrobatics (e.g., flips and kicks) and expressive human-like dances[1–5]. For practical deployment, it is essential that robots not only perform individual skills robustly but also switch between them seamlessly. This paper aims to develop a unified framework for flexible skill switching in humanoid robots.

Multi-skill execution and switching typically require the existence of common states between skills, which imposes significant data requirements. Although using large-scale motion data to train a general whole-body tracking controller [6–9] allows skill switching via goal-conditioned tracking, this approach still relies on pre-defined trajectories with feasible transition states. Otherwise, it may lead to low switching success rate with unnatural movements such as tripping or stumbling [8, 10–13]. Furthermore, due to the open-loop nature of tracking predefined motions, the controller is vulnerable to disturbances: once the tracking error exceeds a certain threshold, stability is compromised, potentially resulting in dangerous and uncontrolled behavior.

We observe that the main challenge in skill switching comes from poorly modeled transitions between different skills. Since the controller lacks training on these transitions, the robot often fails when encountering such unseen states. One straightforward solution is to collect sufficient motion data that cover all possible inter-skill state-level transitions. However, this approach becomes prohibitively expensive as the number of skills grows, since the required transitions scale combinatorially. To overcome this issue, we propose to establish the connections between motion states by exploring state similarity level across different skills, and use reinforcement learning to learn feasible transitions from such augmented data. Furthermore, instead of tracking a fixed reference trajectory in open-loop, the system should have online replanning capabilities. For example, when the tracking error exceeds a safety threshold, the system should replan a feasible trajectory to recover stable execution rather than allowing error to accumulate until failure.

Based on these insights, we propose Switch, a hierarchical whole-body control system that enables seamless skill switching at any moment (Fig. 2). Our framework consists of three key components: a skill graph as augmented dataset,

* Equal contribution †Corresponding author

a unified whole-body tracking policy, and an online skill scheduler. Specifically, we treat motion data frames as graph nodes and transitions between frames as directed edges. Given multiple skill sequences, we insert edges between previously disconnected nodes based on similarity, thus automatically creating transition opportunities. For newly added transitions with significant gaps, we insert buffer nodes to enhance switching smoothness and stability. The number of buffer nodes is calculated by the distance between the endpoints of each edge, with larger gaps leading to more buffer nodes. This process significantly enriches and augments the reference motion dataset. We then train a unified tracking policy in simulation to control the robot for executing both the skills and the learned transitions. A buffer-aware imitation mechanism is introduced to train the policy by reinforcement learning with transitions involving these buffer nodes. Finally, an online skill scheduler provides real-time tracking guidance to the policy. When a skill switch is intended or when tracking failure is detected, the scheduler performs real-time motion planning via shortest-path search over the skill graph. This generates new tracking targets to guide the tracking policy toward smooth skill switching or recovery.

The proposed **Switch** is designed to be versatile across a wide range of locomotion skills, significantly improving both skill transition success rate and execution stability compared to existing methods. Experimental results on the Unitree G1 demonstrate that our system can be successfully deployed in the real world to achieve continuous transitions between dynamic and complex motor skills such as Kung Fu squats, back kicks, and rapid recoveries.

II. Related Work

A. Learning-Based Motion Tracking

Learning-based tracking has achieved significant progress in character animation, enabling lifelike whole-body motions by imitating individual references [14–17] or learning universal tracking [18–22]. Recent efforts extend these techniques to real-world robots, focusing on improving accuracy [2, 3, 5, 9, 23], generalization [6, 7, 24–27], and robustness [1, 5, 28]. For accuracy, some works [5, 23] achieve low short-term tracking errors, while CLONE [9] mitigates long-horizon drift by incorporating an LiDAR odometry. UniTracker [7] further improves tracking precision by introducing a fast adaptation module. For generalization, GMT [6] employs MoE to handle diverse motion tracking, while UniTracker leverages a teacher-student framework to enable scalable tracking across thousands of motions. For robustness, SoFTA [28] proposes to use a slow-fast agent design for end-effector control. BeyondMimic [1] enhances robustness by distilling a diffusion model.

Despite these advancements, inter-sequence transitions remain a challenge, often resulting in inefficiencies or failures during skill switching. To bridge this gap, we introduce short buffer states between pre-linked sequences and propose a method to explicitly learn these transitions.

B. Skill Switching and Sequencing

Skill switching and sequencing have been investigated across animation and robotics through kinematic composition, attractor-based primitives, hierarchical decision layers, and reachability-aware planning. Early work in character animation [29–34] addresses switching by organizing motion data into discrete graphs and local blends. Motion Graphs model a dataset as a directed graph whose nodes are poses or short clips and whose edges are kinematically compatible transitions; long behaviors arise by walking paths on the graph [29]. Parametric Motion Graphs and runtime blending further improve connectivity by interpolating families of motions [35]. While these techniques achieve visually smooth transitions (e.g., via LERP/SLERP interpolation), they operate primarily at the kinematic level and do not guarantee dynamic feasibility or stability on physical robots, revealing a persistent kinematics–dynamics gap.

In robotics, reusable skills are often encoded as dynamical systems with attractors, as in DMP/SEDS, enabling time/space retargeting and composition with stability guarantees [36, 37]. Geometric formulations such as RMPflow provide principled rules to combine task policies while preserving stability properties [38]. Orthogonally, hierarchical RL abstracts time by letting a high-level policy select among low-level options or subgoals [39], and a large body of work discovers/composes skills (e.g., skill chaining and skill graphs) to handle long-horizon tasks [40, 41]. Bridging selection and planning, PRM-RL admits an edge only if a local controller can reliably traverse it, reducing long-range behavior to graph search over feasible edges [42]. In control, Reference/Command Governors and tube-based MPC adjust or filter targets so that tracking remains within constraints, while viability/capture-basin concepts formalize the set of states from which safe progress is possible [43–45].

Building upon these principles, our approach, **Switch**, frames switching as *state-conditioned reference selection*. Specifically, it uses a short-edge skill graph to provide the tracker with feasible subgoals that lead to the target skill. By combining low-level learned tracking with high-level graph search, **Switch** enables smooth transitions and robust recoveries, even with limited transition data or poorly timed user commands.

III. Method

A. Building a Skill Graph for Data Augmentation

Training a universal whole-body motion tracker to enable state transitions between different skills is challenging, as it requires massive, often unavailable high-quality motion data. Alternatively, stitching motion trajectories between all pairs of states from different skills leads to convergence issues, especially since transitions between distant state neighborhoods may be physically infeasible. To address these issues, we need to establish a feasible state mapping for each neighborhood as shown in fig. 2(a) and identify physically achievable state for skill switching.

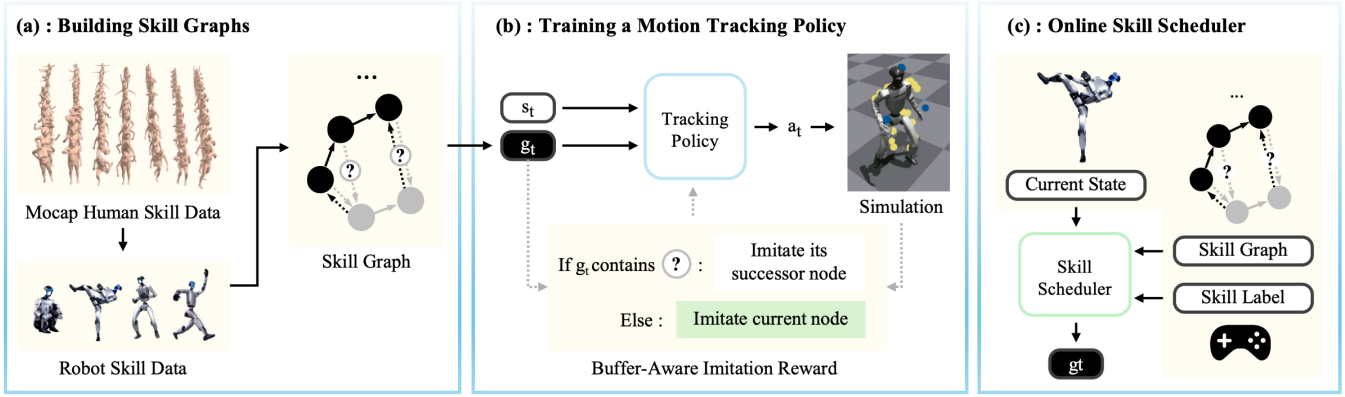

Fig. 2: The Switch system: **(a)** We retarget human motion capture skills onto the robot. We then construct a skill graph where frames serve as nodes (solid circles) and temporal transitions as edges (solid arrows). **black** and **gray** represents two different skills. Based on frame similarity, we add additional transitions between different skills (dashed arrows). When the added transition has significant gap, we insert buffer nodes (question marks) to enhance switching feasibility. **(b)** The assembled Skill Graph is used for training a motion tracking policy. During normal operation, imitation rewards measure the distance between simulated results and reference frame. When referencing a buffer node, we measure with its first non-buffer successor node. **(c)** During deployment, users can change skill labels at any time, and our online scheduler plans optimal paths through the skill graph based on the robot's current state, providing real-time targets to guide the tracking policy.

1) Definition of Skill Graph: We represent candidate state-to-state transitions with a directed, weighted graph $\mathcal{G} = (V, E, w)$, with edge weight function $w : E \to \mathbb{R}_{\geq 0}$. Let $\Phi = \{\phi_1, \ldots, \phi_K\}$ be the set of skills. Each skill $\phi \in \Phi$ provides a reference sequence $\{\hat{s}_t^{\phi}\}_{t=1}^{T_\phi}$, where T_ϕ is its length. We collect all reference states as

$$V = \bigcup_{\phi \in \Phi} \{\hat{s}_t^{\phi}\}_{t=1}^{T_\phi}. \tag{1}$$

Edges E consist of (a) all consecutive pairs within each reference sequence and (b) cross-skill connections between sufficiently similar states. The meaning of the edge weight $w(u, v)$ is phase-dependent and will be instantiated in Sec. III-A.2 (training-time) and Sec. III-C.1 (deployment-time).

2) Construction of Skill Graph:

a) Training-time edge weight (distance): During graph construction we instantiate the edge weight primarily via a state distance. In the local frame (with global x–y translation and yaw/twist removed), define

$$d(\hat{s}^m, \hat{s}^n) = \left\|q^m - q^n\right\|_1 + \left\|\dot{q}^m - \dot{q}^n\right\|_1 + \left\|\hat{p}^m - \hat{p}^n\right\|_1. \tag{2}$$

We then assign the training-time edge weight as

$$w_{\text{train}}(u, v) = \begin{cases} 1, & \text{if } (u, v) = (\hat{s}_t^{\phi}, \hat{s}_{t+1}^{\phi}) \\ d(u, v), & \text{otherwise.} \end{cases} \tag{3}$$

b) Cross-skill connections: Sample a source skill $\phi_p \in \Phi$ and index $i \in \{1, \ldots, T_{\phi_p}\}$, then connect $\hat{s}_i^{\phi_p}$ to its nearest neighbor in a different target skill $\phi_q \in \Phi$ ($\phi_q \neq \phi_p$) under d:

$$j = \arg\min_{t \in [T_{\phi_q}]} d(\hat{s}_i^{\phi_p}, \hat{s}_t^{\phi_q}), \qquad E \leftarrow E \cup \{(\hat{s}_i^{\phi_p}, \hat{s}_j^{\phi_q})\}. \tag{4}$$

B. Training a Whole-Body Tracking Controller in Simulation

1) Problem Formulation: We formalize the multi-skill learning problem as a Markov Decision Process (MDP) $M = \langle S, A, P, r, \gamma \rangle$, where $s \in S$ is the robot state, $a \in A$ is the action, $P(\cdot \mid s, a)$ is the state-transition kernel over S, r is the reward function (covering motion imitation and regularization terms), and $\gamma \in [0, 1)$ is the discount factor. At time t, the policy input is $o_t = \langle s_t^o, g_t \rangle$, where $s_t^o \triangleq \langle a_{t:t-n}, f_{t:t-n}, q_{t:t-n}, \dot{q}_{t:t-n}, \omega_{t:t-n}^{\text{root}} \rangle$ with $a_t \in \mathbb{R}^{23}$ the previous actions (target joint positions), $f_t \in \mathbb{R}^3$ the body-frame gravity vector, $q_t, \dot{q}_t \in \mathbb{R}^{23}$ the joint positions/velocities, and $\omega^{\text{root}} \in \mathbb{R}^3$ the root angular velocity. The guidance input is $g_t = \langle \hat{s}_t^g, \kappa_t \rangle$, where $\hat{s}_t^g = \hat{p}_t^g$ encodes reference local rigid-body positions with $\hat{p}_t^g \in \mathbb{R}^{3\times 23}$, and $\kappa_t \in \mathbb{N}0$ is an integer indicating the remaining number of steps to the end of a buffer segment (set $\kappa_t = 0$ if the target is not a buffer node). The instantaneous reward is $r_t = r(s_t^o, \hat{s}_t^g)$, and the action $a_t \in \mathbb{R}^{23}$ specifies desired joint positions tracked by a PD controller. Our objective is to learn a robust policy π that enables smooth skill transitions while maintaining high-fidelity motion imitation; given reference sequences $\phi_p = \{\hat{s}_0^p, \ldots, \hat{s}_{T_{\phi_p}}^p\}$ and $\phi_q = \{\hat{s}_0^q, \ldots, \hat{s}_{T_{\phi_q}}^q\}$ from any two skills, we maximize $\mathbb{E}[\sum_{t=0}^{T-1} \gamma^{t-1} r_t]$. We adopt Proximal Policy Optimization (PPO) for training.

2) Reference State Inititaion: Proper task initialization is critical for reinforcement learning (RL) training, and previous works have employed Reference State Initialization (RSI) [2, 14, 46–48] to enhance the learning process. However, directly applying RSI to our augmented motion trajectories hinders the learning of skill transitions when states are sampled uniformly across the entire motion sequence, some sampled states occur after skill transitions, leading to scenarios where the agent fails to experience skill transitions when initialized from these states. To resolve this issue, we modify RSI by only sampling initial states that are n steps before skill transitions. This adjustment ensures that every

sampled initial state allows the agent to encounter a skill transition during training.

3) Buffer-aware Imitation Learning: Basic imitation learning methods are constrained by their reliance on explicitly defined reference states for supervision, yet cross-skill transition states are often unavailable. Furthermore, their quantity grows polynomially with the number of skills which make data collection prohibitively expensive and their sparsity may render single-step transitions infeasible. To address this, we insert N buffer nodes between states, with the number N determined by the similarity level between the states; these nodes act as temporal buffers to bridge distant transitions, enabling the agent to explore viable paths instead of relying on unavailable predefined reference trajectories. Formally, the motion trajectory is constructed as:

$$\left\{ \hat{s}_0^{\phi_p}, \ldots, \hat{s}_i^{\phi_p}, \underbrace{s_\emptyset, \ldots, s_\emptyset}_{N}, \hat{s}_j^{\phi_q}, \ldots, \hat{s}_L^{\phi_q} \right\}, \tag{5}$$

where N is computed similarly to [49]. Unlike [49], which omits reward computation during the buffer stage, we use the target state $\hat{s}_j^{\phi_q}$ to calculate rewards in this phase as demonstrated in fig. 2(b). This guides the model toward the target state, facilitating convergence and preventing drastic deviations that could cause performance collapse.

4) Unified Imitation Rewards: We observe that standard imitation rewards underweight high-frequency foot–ground events (e.g., dancing, martial arts), causing the policy to act conservatively and degrade agility and motion fidelity. To explicitly supervise contact events, we add a Foot–Ground Contact Reward (FGR).

Let G denote the index set of foot end-effectors that can contact the ground (e.g., $G = \{\mathrm{LF}, \mathrm{RF}\}$). For each $i \in G$ and time t, let $c_{i,t}^{\mathrm{ref}} \in \{0, 1\}$ be the reference contact label (obtained via self-labeling adapted from [3]) and $c_{i,t} \in \{0, 1\}$ be the measured contact status. We define

$$r_t^{\mathrm{FGR}} = \exp\Big(- \lambda_c \sum_{i \in G} |c_{i,t} - c_{i,t}^{\mathrm{ref}}| \Big), \tag{6}$$

The overall reward function r_t is the sum of three components: task rewards for achieving precise whole-body tracking, penalties for preventing undesirable motions, and a regularizations for refining motion.

5) Skills Training Curriculum: Training a policy to achieve smooth skill transitions in simulation remains challenging, particularly during early stages when the agent lacks prior knowledge of locomotor skills. To address this, we introduce three progressive training curricula for step-by-step learning, ensuring that the agent builds foundational locomotive skills first before tackling complex skill transitions.

- *Skill Augmentation Curriculum* initializes with a 10% augmented motion trajectories and 90% of trajectories are single-skill to let the agent focus on learning basic locomotion through consistent single-skill practice. As training progress, the probability is incrementally raised to 50%, systematically increasing demands for exploring and mastering skill transitions.

TABLE I: Reward Specification

Task Reward			
Body position	1.125	VR 3-point	1.8
Body position (feet)	2.3625	Body rotation	0.5
Body angular velocity	0.5	Body velocity	0.5
DoF position	0.75	DoF velocity	0.5
FGR	1.8		
Penalty			
DoF position limits	-10.0	DoF velocity limits	-5.0
Torque limits	-5.0	Termination	-200.0
Regularization			
Torques	-1×10^{-5}	Action rate	-0.5

- *Reward Penalty Curriculum* [2, 7] applies to regularization and penalty terms: it allows the policy to prioritize core tasks (e.g., basic locomotion or single-skill tracking) initially without being overwhelmed by extra constraints. By introducing these components incrementally, the policy can develop more reasonable, well-adjusted behaviors over time, rather than struggling to balance multiple objectives from the start
- *Termination Curriculum* [2] terminates episodes when the robot deviates from the reference motion beyond a threshold: starting with a generous threshold, it is gradually tightened to incrementally raise tracking demands, facilitating the learning of agile motions and improving tracking performance.

C. Deploy: Online Scheduling with Search-Based Planners

After training, the tracking policy demonstrates seamless tracking of skills and skill switching, while maintaining robustness against certain level of disturbances. Nevertheless, a naive approach that merely follows a predefined motion reference suffers from active runtime controllability [2, 6]. This deficiency becomes particularly problematic when addressing accumulated tracking errors or severe perturbations, as illustrated in Fig. 3b. We therefore employ an *online skill scheduler* in deployment for switching and safety. When a trigger occurs, it invokes a *planner* to produce an entry and a path to a target set T, from which we synthesize the reference for the controller.

1) Shared Graph Model and Planner Interface: In deployment, we reuse the identical skill graph $\mathcal{G}$ constructed during training. Reference frames serve as nodes, and directed edges correspond to the transitions established in training. Given that collecting or learning transitions for all $O(n^2)$ skill pairs is impractical, this graph acts as a data-driven prior—restricting the planner to search only over edges sampled or synthesized during training.

Each edge is assigned a nonnegative deployment-time weight $w_{\text{deploy}}(u \to v) = c(u \to v)$. We set $c = 1$ for consecutive edges $(\hat{s}^{\phi}t \to \hat{s}^{\phi}t + 1)$ within the same skill. For

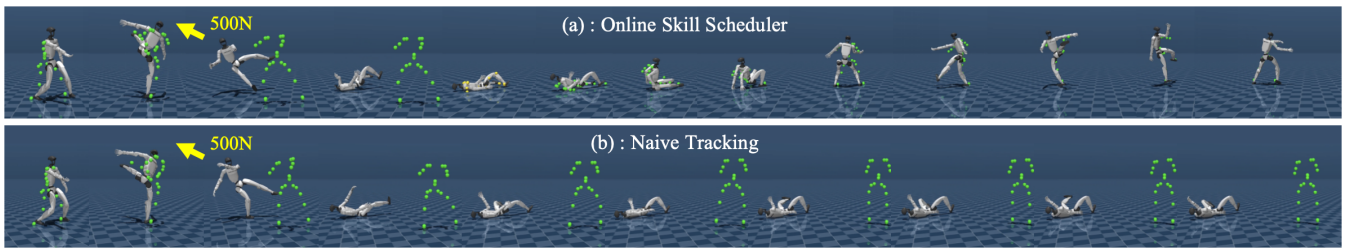

Fig. 3: Effect of the online skill scheduler. (a) When the robot performs a kicking skill, we apply a 500N disturbance force causing it to fall. This triggers the scheduler to automatically replan based on tracking errors, selecting an appropriate path (utilizing get-up skill segments) to resume the kicking execution. (b) Without the scheduler, tracking errors will gradually amplify and unable to recover.

cross-skill edges ($\hat{s}_i^{\phi_p} \to \hat{s}_j^{\phi_q},\ \phi_p \neq \phi_q$) we use a distance-plus-penalty form:

$$c(u \to v) \;=\; d(u,v) \;+\; \lambda_{\text{sw}}\,[\,\text{skill}(u) \neq \text{skill}(v)\,], \qquad (7)$$

where $d(\cdot,\cdot)$ is the state distance defined in Sec. III-A.2 . The total path weight is the sum of edge weights, and planning is posed as a shortest-path-to-set problem on $\mathcal{G}$:

$$\min_{\pi: v \to T} \sum_{e \in \pi} c(e), \qquad (8)$$

2) Planner Choices:

a) Graph-Search planner: We run a reverse multi-source shortest path from T to obtain a value function V and a next-hop map `NextHop`. Given an entry, the path is reconstructed by iterating `NextHop`. This global view can leverage overlaps between RoAs e.g., briefly proceeding in the current skill to a nearby feasible state before merging into the target (see Fig. 3).

b) Nearest-Neighbor (NN) planner: We select the entry by nearest-neighbor similarity (optionally among a small candidate set) and form a single-hop or short-hop transition toward T without global graph search. This planner has minimal latency and a simple implementation; in our hardware deployment we use the NN variant.

3) Problem A: Intent-Driven Switching: Given a commanded skill, we define the target set as its prefix T_{cmd} (e.g., the first τ fraction of frames) and seek a plan that reaches any $t \in T_{\text{cmd}}$ with minimum cost.

Entry check and selection. We compute similarity in the local pose–velocity subspace. If $\text{sim} \leq A$, the state lies in a reference RoA and we attach directly. If $\text{sim} \geq B$, we defer switching and enter emergency stop (e-stop). For $A < \text{sim} < B$, we evaluate the top-k candidates by a composite score

$$J(v) = \lambda_{\text{cost}}\, c(x,v) + \underbrace{V(v)}_{\text{if available}}, \qquad (9)$$

where V is used when the Graph-Search planner is selected; with NN this term is omitted. The scheduler then calls $\texttt{Plan}(\mathcal{G}, T_{\text{cmd}}, x, \text{candidates})$ and installs the synthesized reference.

4) Problem B: Safety Recovery (emergency stop): Safety recovery is triggered when safety checks detect excessive divergence either when the similarity $sim \geq B$ during tracking, or when the best candidate still violates B during entry selection. By default, recovery retains the original commanded target set T_{cmd}. The planner is allowed to route through recovery skills as intermediate segments on the path to T_{cmd} whenever this yields a lower-cost or safer plan. For the Graph-Search planner, we set $T = T_{cmd}$ and compute the shortest path, with recovery skills naturally integrated into the path. For the Nearest-Neighbor (NN) planner, if a direct jump toward T_{cmd} is unsafe, we optionally introduce a recovery target T_{rec} and adopt a two-stage plan: first reach T_{rec}, then re-plan to T_{cmd}. During e-stop, we override the policy with a damping controller and wait until the system is stationary (e.g. angular velocity of the root below thresholds) before executing the recovery plan.

5) Online Skill Scheduling: The planner is invoked under four conditions: initialization, a user-commanded target change, approaching the end of the current reference, and safety-threshold events when sim crosses A or B. On a trigger, the scheduler performs the RoA-based entry check, forms `candidates`, and invokes the selected planner with $T = T_{\text{cmd}}$ (*for the NN planner only*, if a direct switch toward T_{cmd} is unsafe, we optionally use a two-stage call with a recovery target T_{rec} before returning to T_{cmd}). With Graph Search, caching V and `NextHop` makes replanning a nearest-neighbor lookup plus pointer chasing; With NN, replanning is just the nearest-neighbor lookup. Both yield low online latency.

IV. Experiment

A. Experiment Setup

1) Hardware Setup: We conduct experiments on the Unitree G1 humanoid robot with 29 DoF. The learned policy runs onboard the robot using a Jetson Orin NX.

2) Baseline: We compare **Switch** with its ablated versions and GMT [6].

- Base: The basic RL policy trained on single skill data only using motion tracking method [2].
- Base + SG: Trained using motion tracking on Skill Graph-augmented trajectories.
- Base + SG + B: Trained using motion tracking on Skill Graph-augmented trajectories with buffer states.
- Base + SG + B + C (**Switch**): Trained using motion tracking on Skill Graph-augmented trajectories with buffer states and incorporating Foot-Ground Contact Reward (FGR).
- GMT [6]: A state-of-the-art general tracking model.

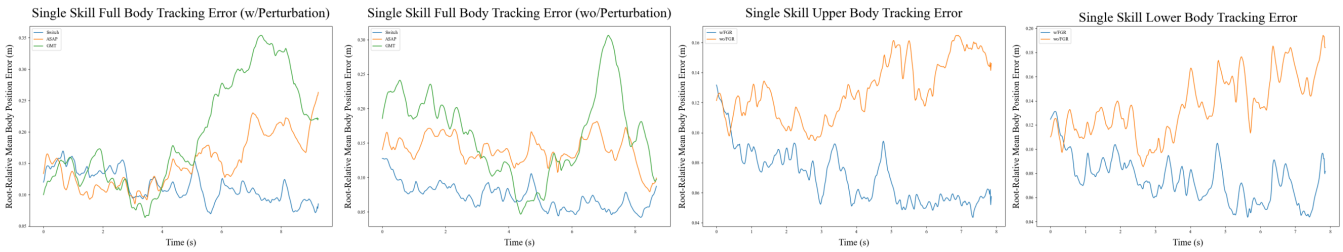

Fig. 4: Quantitative Evaluation of Skill Execution Performance. We analyzes the full-body tracking performance of the proposed **Switch**, alongside baselines ASAP and GMT, under both perturbed and unperturbed conditions. The results show that Switch maintains lower overall full-body tracking errors in single-skill execution experiments regardless of the presence of perturbation demonstraing its robotness to disturbance. Additionally, we also investigate the effectiveness of the Foot-Ground Contact Reward (FGR) on tracking performance. The results shows that it enhance full body motion with a more distinct improvement observed in the lower body motion tracking.

TABLE II: Sim2Sim Comparison in MuJoCo. We compare **Switch** with ablated versions and GMT across three task difficulty levels. **Switch** consistently achieves the highest Skill Switching Success Rate (SSR) and lowest motion errors, highlighting the effectiveness of each component. Here we use SG to represent the skill graph, B for the buffer state inpainting, and C for the foot-ground contact reward.

Method	SSR (%) ↑	NR ↑	$E_{\text{g-mpbpe}}$ ↓	E_{mpbpe} ↓	E_{mpjpe} ↓	E_{mpjve} ↓	E_{mpbve} ↓	E_{mpbae} ↓
Easy								
GMT [6]	30.00	2.636	0.396	0.392	0.295	1.420	**0.391**	**5.651**
Base	2.00	4.520	0.120	0.118	0.307	**1.310**	0.398	6.669
Base + SG	**100.0**	5.802	0.115	0.121	0.241	1.489	0.451	7.883
Base + SG + B	**100.0**	6.629	0.087	0.092	**0.213**	1.472	0.446	8.089
Base + SG + B + C (**Switch**)	**100.0**	**7.029**	**0.075**	**0.078**	0.220	1.483	0.454	8.026
Medium								
GMT [6]	10.00	2.531	0.491	0.486	0.304	1.476	0.397	**5.681**
Base	2.00	4.505	0.120	0.120	0.307	**1.275**	**0.393**	6.560
Base + SG	**100.0**	5.175	0.147	0.151	0.255	1.464	0.460	7.734
Base + SG + B	**100.0**	6.128	0.103	0.106	**0.214**	1.492	0.459	8.304
Base + SG + B + C (**Switch**)	**100.0**	**6.616**	**0.090**	**0.098**	0.223	1.484	0.475	7.778
Hard								
GMT [6]	2.00	2.877	0.588	0.591	0.236	**1.400**	**0.405**	**6.062**
Base	2.00	3.570	0.251	0.253	0.306	1.630	0.468	6.416
Base + SG	**100.0**	5.091	0.146	0.150	0.246	1.531	0.469	8.129
Base + SG + B	**100.0**	6.015	0.105	0.105	**0.213**	1.446	0.460	8.078
Base + SG + B + C (**Switch**)	**100.0**	**6.284**	**0.098**	**0.093**	0.225	1.458	0.463	7.909

3) Evaluation Tasks: We evaluate the policy's tracking performance using a motion dataset that encompasses 4 distinct skills. Skill transitions are categorized into three difficulty levels, easy, medium, and hard, based on their number of transition to perform. Specifically, an "easy" transition refers to switching between skills once, a "medium" transition involves switching skills twice, and a "hard" transition refers switching skills three times. For each of these difficulty settings, the policies are trained in IsaacGym and subsequently evaluated in MuJoCo [50] over 50 trials each.

4) Evaluation Metric: We employ a set of metrics to assess both the skill transition effectiveness and motion imitation accuracy of the proposed method. First, the Skill Switching Success Rate (SSR) quantifies the percentage of successful target skill transitions when the policy is initialized from an arbitrary state different from the target skill; a transition is deemed unsuccessful if the average body position error relative to the root frame exceeds 0.5 meters at any point during imitation. Second, the Normalized Reward (NR) calculates the average reward per frame, formulated as $NR = \frac{1}{T}\sum_{t=0}^{T-1} \bar{r}_t$ where $\bar{r}_t$ denotes the normalized reward at time t and T is the total number of frames processed. Additionally, we evaluate motion imitation fidelity via multiple tracking error metrics: Global Mean Per Body Position Error ($E_{\text{g-mpbpe}}$, in meters), Root-Relative Mean Per Body Position Error (E_{mpbpe}, in meters), Mean Per Joint

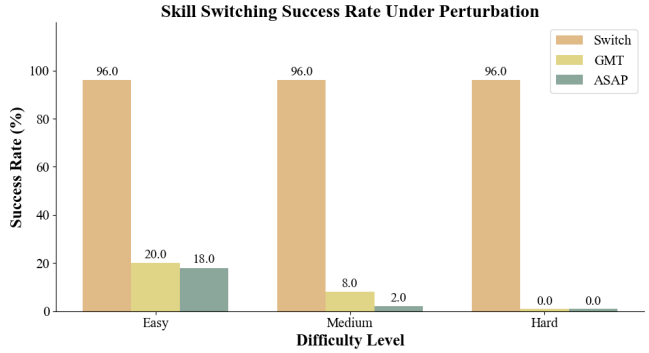

Fig. 5: Skill switching success rate under perturbation. The perturbation is random 25 N pushes in simulation.

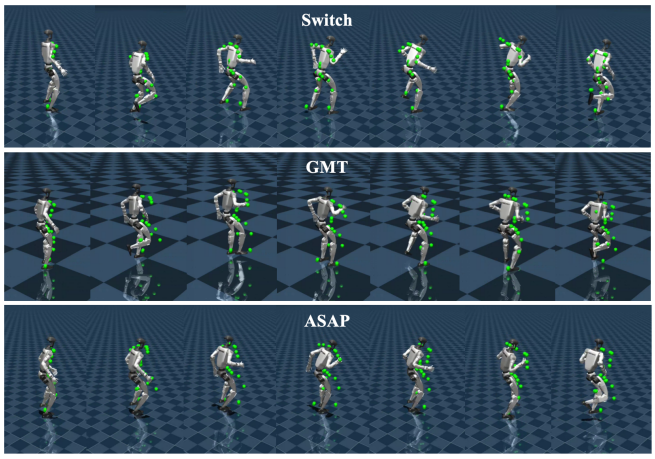

Fig. 6: Visual comparisons of skill execution (wo/Perturbation) across different methods in dancing which requires high-frequency foot-ground contact. **Switch** demonstrates more coordinated lower-body movement and attains better foot-ground interaction while ASAP [2] and GMT [6] exhibit conservative and jerky lower body motions, particularly when agile movements are required.

Position Error (E_{mpjpe}, in radians), Mean Per Joint Velocity Error (E_{mpjve}, in radians per frame), Mean Per Body Velocity Error (E_{mpbve}, in meters per frame), and Mean Per Body Acceleration Error (E_{mpbae}, in meters per frame^2).

B. Experimental Results

1) Performance in Skills Switching: As summarized in Table II, **Switch** (denoted as Base + SG + B + C) achieves consistent and superior performance across all difficulty levels, outperforming all compared baselines. Regarding SSR, the Base model, trained solely on single-skill data stagnates at a mere 2.00% across all levels, failing to execute even simple skill switching. The state-of-the-art general tracking model GMT [6] exhibits a drastic decline in SSR with increasing difficulty, dropping from 30.00% (Easy) to 2.00% (Hard), showing the limitation of general tracking model's reliance on pre-defined trajectories with feasible transition states in skill switching tasks. In contrast, models integrating the Skill Graph (SG) (Base + SG, Base + SG + B, **Switch**) attain a perfect 100% SSR across all levels, proving the effectiveness of SG in enabling cross-skill transitions learning.

2) Performance in Skills Execution: In terms of performance on skill execution, **Switch** maintains the lowest errors in key metrics across most difficulty levels. From Table II, its Global Mean Per Body Position Error is 0.075m (Easy), 0.090m (Medium), and 0.098m (Hard), substantially lower than GMT's 0.396m, 0.491m, and 0.588m, and also outperforming Base + SG + B (0.087m, 0.103m, 0.105m). Similarly, Switch achieves the smallest Root-Relative Mean Per Body Position Error (0.078m, 0.098m, 0.093m), reflecting precise whole-body motion alignment with reference trajectories. The improvement is particularly pronounced in lower-body motion as shown in Figure 6, which demonstrates Switch's more coordinated lower-body movements compared to GMT [6] and ASAP [2]. Critically, even in the Hard difficulty level (involving three consecutive skill switches) as shown in Figure 4, Switch retains low tracking errors, demonstrating its robustness to perturbation and escalating task complexity.

3) Effectiveness of Foot-Ground Contact Reward: To investigate the impact of the Foot-Ground Contact Reward (FGR) on motion tracking, we compared the upper and lower body motion tracking performance for individual skills by assessing our policy trained both with and without FGR. From Figure 4, we found that when FGR is incorporated, the tracking performance improves, with the lower body exhibiting a more pronounced enhancement . This is attributed to the fact that the foot-ground contact reward directly strengthens the interaction between the robot's lower body and the ground, refining the precision of lower-body motion. Figure 6 presents the qualitative comparsion between **Switch** and other motion tracking baseline. It can be observed that **Switch** achieves better foot-ground contact compared to GMT and ASAP on agile skill execution such as dancing, and demonstrates stable, high-fidelity motion imitation.

V. Conclusion

In this paper, we propose **Switch**, which addresses the critical challenge of multi-skill switching in humanoid robots by integrating a Skill Graph for data augmentation, a reinforced whole-body tracking controller with buffer states and foot-ground contact optimization, and an online scheduler for real-time planning. Unlike baselines that struggle with increasing task difficulty or perturbations, Switch achieves 100% SSR across all skill switching levels, maintains low tracking errors, and autonomously recovers from disturbances, demonstrating its ability to enable agile, seamless skill execution and laying a foundation for practical deployment of humanoid robots in dynamic real-world scenarios.

References

[1] T. E. Truong, Q. Liao, X. Huang, G. Tevet, C. K. Liu, and K. Sreenath, "Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion," *arXiv preprint arXiv:2508.08241*, 2025.

[2] T. He *et al.*, "Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills," *arXiv preprint arXiv:2502.01143*, 2025.

[3] W. Xie *et al.*, "Kungfubot: Physics-based humanoid whole-body control for learning highly-dynamic skills," *arXiv preprint arXiv:2506.12851*, 2025.

[4] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang, "Expressive whole-body control for humanoid robots," *arXiv preprint arXiv:2402.16796*, 2024.

[5] M. Ji *et al.*, "Exbody2: Advanced expressive humanoid whole-body control," *arXiv preprint arXiv:2412.13196*, 2024.

[6] Z. Chen, M. Ji, X. Cheng, X. Peng, X. B. Peng, and X. Wang, "Gmt: General motion tracking for humanoid whole-body control," *arXiv preprint arXiv:2506.14770*, 2025.

[7] K. Yin *et al.*, "Unitracker: Learning universal whole-body motion tracker for humanoid robots," *arXiv preprint arXiv:2507.07356*, 2025.

[8] T. He *et al.*, "Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning," *arXiv preprint arXiv:2406.08858*, 2024.

[9] Y. Li *et al.*, "Clone: Closed-loop whole-body humanoid teleoperation for long-horizon tasks," *arXiv preprint arXiv:2506.08931*, 2025.

[10] T. He *et al.*, "Learning human-to-humanoid real-time whole-body teleoperation. in 2024 ieee," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8944–8951.

[11] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," *arXiv preprint arXiv:2407.01512*, 2024.

[12] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," *arXiv preprint arXiv:2406.10454*, 2024.

[13] C. Lu *et al.*, "Mobile-television: Predictive motion priors for humanoid whole-body control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2025, pp. 5364–5371.

[14] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[15] Y. Wang, J. Lin, A. Zeng, Z. Luo, J. Zhang, and L. Zhang, "Physhoi: Physics-based imitation of dynamic human-object interaction," *arXiv preprint arXiv:2312.04393*, 2023.

[16] M. Xu, Y. Shi, K. Yin, and X. B. Peng, "Parc: Physics-based augmentation with reinforcement learning for character controllers," in *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, 2025, pp. 1–11.

[17] Y. Wang *et al.*, "Skillmimic: Learning basketball interaction skills from demonstrations," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 17 540–17 549.

[18] Z. Luo, J. Cao, K. Kitani, W. Xu, *et al.*, "Perpetual humanoid control for real-time simulated avatars," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 895–10 904.

[19] Z. Luo *et al.*, "Universal humanoid motion representations for physics-based control," in *The Twelfth International Conference on Learning Representations*, 2024.

[20] S. Xu, H. Y. Ling, Y.-X. Wang, and L.-Y. Gui, "Intermimic: Towards universal whole-body control for physics-based human-object interactions," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 12 266–12 277.

[21] C. Tessler, Y. Guo, O. Nabati, G. Chechik, and X. B. Peng, "Maskedmimic: Unified physics-based character control through masked motion inpainting," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–21, 2024.

[22] C. Tessler, Y. Jiang, E. Coumans, Z. Luo, G. Chechik, and X. B. Peng, "Maskedmanipulator: Versatile whole-body control for loco-manipulation," *arXiv preprint arXiv:2505.19086*, 2025.

[23] Y. Ze *et al.*, "Twist: Teleoperated whole-body imitation system," *arXiv preprint arXiv:2505.02833*, 2025.

[24] Y. Shao *et al.*, "Langwbc: Language-directed humanoid whole-body control via end-to-end learning," *arXiv preprint arXiv:2504.21738*, 2025.

[25] Y. Wang *et al.*, "From experts to a generalist: Toward general whole-body control for humanoid robots," *arXiv preprint arXiv:2506.12779*, 2025.

[26] H. Xue *et al.*, "Leverb: Humanoid whole-body control with latent vision-language instruction," *arXiv preprint arXiv:2506.13751*, 2025.

[27] Y. Wang *et al.*, "Humanx: Toward agile and generalizable humanoid interaction skills from human videos," *arXiv preprint arXiv:2602.02473*, 2026.

[28] Y. Li *et al.*, "Hold my beer: Learning gentle humanoid locomotion and end-effector stabilization control," in *RSS 2025 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*.

[29] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 473–482, 2002.

[30] R. Heck and M. Gleicher, "Parametric motion graphs," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D)*, ACM, 2007, pp. 129–136.

[31] L. Zhao and A. Safonova, "Achieving good connectivity in motion graphs," in *Eurographics/SIGGRAPH Symposium on Computer Animation (SCA)*, Eurographics Association, 2008, pp. 127–136.

[32] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin, "Motion-motif graphs," in *Eurographics/SIGGRAPH Symposium on Computer Animation (SCA)*, Eurographics Association, 2008, pp. 117–126.

[33] P. S. A. Reitsma and N. S. Pollard, "Evaluating motion graphs for character animation," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 4, 18:1–18:es, 2007.

[34] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large data sets," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 559–568, 2004.

[35] R. Heck and M. Gleicher, "Parametric motion graphs," in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, 2007, pp. 129–136.

[36] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[37] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[38] C.-A. Cheng *et al.*, "Rmp flow: A computational graph for automatic motion policy generation," in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2018, pp. 441–457.

[39] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[40] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, 2009.

[41] A. Bagaria, J. K. Senthil, and G. Konidaris, "Skill discovery for exploration and planning using deep skill graphs," in *International conference on machine learning*, PMLR, 2021, pp. 521–531.

[42] A. Faust *et al.*, "Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 5113–5120.

[43] A. Bemporad, "Reference governor for constrained nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 415–419, 2002.

[44] S. Raković and D. Mayne, "A simple tube controller for efficient robust model predictive control of constrained linear discrete time systems subject to bounded disturbances," *IFAC proceedings volumes*, vol. 38, no. 1, pp. 241–246, 2005.

[45] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability theory: new directions*. Springer Science & Business Media, 2011.

[46] T. Haarnoja *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Science Robotics*, vol. 9, no. 89, eadi8022, 2024.

[47] T. He *et al.*, "Hover: Versatile neural whole-body controller for humanoid robots," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2025, pp. 9989–9996.

[48] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions On Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[49] R. Yu *et al.*, "Skillmimic-v2: Learning robust and generalizable interaction skills from sparse and noisy demonstrations," in *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, 2025, pp. 1–11.

[50] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2012, pp. 5026–5033.