ARTICLE

Check for updates

Journal of
Sensor Science and Technology

The Korean Sensors Society

Sensor Data-Driven Modular Network Model for Jump Motion in One-Legged Robot

Hyeonjin Kim[1] and Jinhyun Kim[2,+]

[1]*Department of Mechanical Engineering, Seoul National University of Science and Technology, 232 Gongneung-ro, Nowon-gu, Seoul, 01811, Republic of Korea*
[2]*Department of Mechanical and Automotive Engineering, Seoul National University of Science and Technology, 232 Gongneung-ro, Nowon-gu, Seoul, 01811, Republic of Korea*

Cite This: *J. Sens. Sci. Technol. Vol. 34, No. 6 (2025) 701-709*

https://doi.org/10.46670/JSST.2025.34.6.701

ABSTRACT: In this study, we propose a three-phase learning network-based reinforcement-learning model to implement the jumping motion of a one-legged robot. The jumping motion of a one-legged robot is divided into three phases, each of which can be trained independently. This method ensures high performance and faster convergence through partial optimization and efficient learning. Furthermore, to mitigate the discontinuities during phase transitions, we propose a switch-transition algorithm. The results indicate that the switch-transition model complements the movement during the transition section, thus ensuring continuity throughout the entire jumping motion.

KEYWORDS: *One-legged robot, Reinforcement learning, Modular network model*

1. INTRODUCTION

In the field of robot control, reinforcement learning (RL) is garnering significant attention as a data-driven approach that can achieve high control performance even in environments where modeling is challenging or the dynamic characteristics are complex and highly uncertain. RL enables robots to self-learn optimal policies through interactions with the environment without explicit mathematical formulas, thus rendering it particularly suitable for high-dimensional, nonlinear, and multi-degree-of-freedom systems. Based on these characteristics, RL is actively applied to motion-control problems that challenging to solve using conventional control techniques, such as walking, jumping, and task manipulation [1-3], and its performance continues to improve through integration with various physics-based simulation environments.

Recently, RL-based robot control has been actively investigated to enable robots to perform high-difficulty motions with advanced flexibility and agility, i.e., extending beyond repetitive tasks such as object manipulation or simple walking. Jumping motions, as shown in Fig. 1, are extremely difficult to implement because they require instantaneous propulsion generation, precise balance control, and stable posture maintenance during the aerial phase. Accordingly, several researchers have proposed various approaches for implementing jumping motions in legged robots. Early studies on robot jumping primarily used methods that defined control trajectories over time through precise dynamic analysis and mathematical modeling [4-6]. However, these approaches demonstrate limitations in achieving stable performance in actual environments because of issues such as robot modeling errors, parameter uncertainty, and sensitivity to disturbances. Hence, recent studies used optimization techniques to numerically calculate robot motion and derive control strategies that account for energy efficiency or landing stability [7,8].

Furthermore, owing to advancements in artificial intelligence technology, studies pertaining to data-driven learning techniques using motion-capture data [9,10] or allowing robots to learn jumping motions directly using RL are actively underway [11]. In such RL-based studies, a curriculum-learning method is primarily used that analyzes human jumping motions, separates them into phases, and designs reward functions for each phase to support progressive learning [11]. However, because the curriculum-

+Corresponding author: jinhyun@seoultech.ac.kr
Received : Oct. 28, 2025, Revised : Nov. 7, 2025, Accepted : Nov. 10, 2025

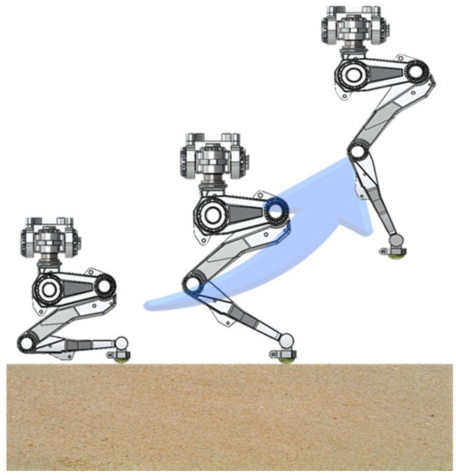

Fig. 1. Schematic diagram of overall jumping motion.

learning method trains the entire jump sequence in a single network, clearly distinguishing the dynamic differences between phases remains challenging, and instability inevitably occurs at the boundaries [12,13]. Additionally, the learning process is complex, as the reward functions for each phase must be finely tuned over time, and minute errors in the early stages of learning can accumulate in later motions, thus resulting in overall performance degradation. Hence, a phase-wise modular learning method has been proposed that classifies complex motions into phases, learns each phase individually, and switches between them appropriately based on the situation [14-16]. The modular learning method improves the learning efficiency and performance by independently acquiring various robot motion skills and allowing the flexible execution of motions based on situations by combining or switching learned skills. Moreover, because the policies for each phase are individually optimized, complex dynamic characteristics can be learned more precisely. Therefore, even when the environmental or target conditions change, rapid adaptation is achievable through partial fine-tuning.

This study proposes a new control framework that can learn stable and flexible jumping motions, even in structurally unstable one-legged robot systems, by overcoming the limitations of existing individual-network-based RL control through modular learning. Several studies have implemented stable motion generation and recovery motion learning based on RL by introducing CNN-based policy networks and domain randomization techniques for bipedal and quadrupedal robots. Whereas these systems feature a relatively large support base, thus rendering posture control easier, one-legged robots are structurally unstable, and control studies applying RL are relatively scarce. One-legged robots have a narrow support

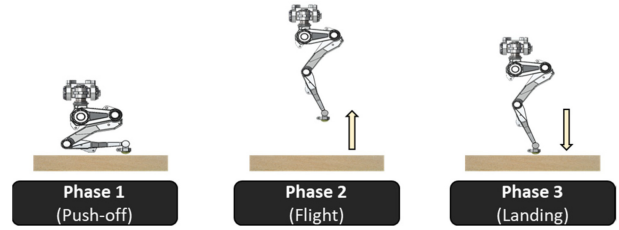

Fig. 2. Definition of jumping motion at each phase.

base and the entire body is airborne during jumping, thus rendering posture control and balance maintenance more difficult. In particular, stable performance during high-speed, high-power movements such as jumping poses a significant challenge.

Therefore, this study aims to design an RL-based jump-control framework that enables a one-legged robot to stably perform vertical jumping. First, we analyze human jumping motions and the jumping phases proposed in existing studies to select the optimal jumping phases. Subsequently, we design phase-specific reward functions and learning strategies to implement them. The jumping motion is divided into three distinct phases based on system state changes: push-off preparing for the jump, flight jumping through propulsion, and landing maintaining balance upon landing. For each motion phase, an optimal reward function is defined to train the individual policy networks. However, this phase-wise modular learning method may cause control discontinuity or motion instability during the robot's action transitions, as well as can cause performance degradation if the transition timing and policy harmony are not sufficiently considered.

Hence, we propose a transition management system that defines phase-transition sections and enables stable phase switching. Furthermore, we enable stable phase transitions by utilizing a skill-transition model that corrects the state differences and discontinuities occurring in the transition sections, thus allowing policies to switch naturally. This approach is expected to be effectively applied not only to jumping motions but also to the phase-wise learning and switching processes of various motions. A structural overview of the model proposed in this study is shown in Fig. 2, and the functions and interactions of each phase are described in detail in Section 2.

2. METHODOLOGY

Section 2 proposes a learning method for implementing the vertical jumping motion of a one-legged robot, based on existing studies pertaining to robot jumping. In Section 2.1, we define the motion phases required for a one-legged robot to

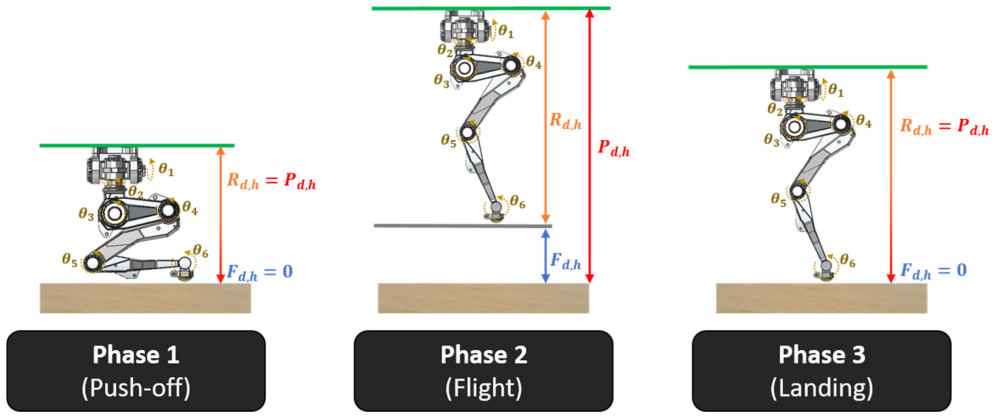

Fig. 3. Design of reward function for implementing robot's jumping motion at each phase.

achieve optimal jumping based on an analysis of human and robot jumping motions reviewed in prior studies. Section 2.2 discusses the design of reward functions for learning the robot motion in each phase, while Section 2.3 introduces a skill-transition model to address the learning instability that may occur during phase transitions. This model is designed to reduce state discrepancies in transition intervals and ensure natural policy switching. Finally, Section 2.4 introduces the training strategy for the jumping motion of a one-legged robot using the skill-transition model proposed in this study.

2.1 Analysis of Human Jumping Motion and Phase Definition

The classification of "preparation, acceleration, flight, buffering, and recovery" [17] serves as a foundational reference for understanding the characteristics of human jumping and for learning humanoid robot motions. Based on this, various studies pertaining to robot jumping have classified motion into three phases, i.e., preparation, flight, and landing [18,19], or defined it as a two-phase process comprising flight and landing by merging the preparation and flight phases [20,21].

In this study, we categorized the robot's jumping motion into three phases, i.e., push-off, flight, and landing, based on the points of system state transitions, as shown in Fig. 2. The push-off phase focuses on generating propulsion by maximizing ground reaction forces; the flight phase prioritizes center-of-mass (CoM) stabilization and posture maintenance, and the landing phase centers on shock absorption and balance restoration.

One-legged robots have an extremely narrow support base, which renders posture and balance maintenance challenging. Under these conditions, the robot can learn precise jumping motions while maintaining a stable balance by distinguishing the phases of the robot based on the system state transitions and by clearly defining the operational requirements and postures for each phase.

2.2 Design of Reward Functions

The one-legged robot used for RL in this study comprises six joints, and its action is defined as the torque of each joint at each time step.

$$a_t = [\tau_1, \tau_2, \ldots, \tau_6] \in \mathbb{R}^6 \tag{1}$$

In this study, we utilized robot state data obtained from a simulation environment based on information measured using sensors in an actual robot system. The state vector is defined as follows:

$$s_t = [h, \dot{x}, \dot{y}, \dot{z}, \alpha, \beta, \gamma, \dot{\alpha}, \dot{\beta}, \dot{\gamma}, \theta_{1:6}, \dot{\theta}_{1:6}] \in \mathbb{R}^{22} \tag{2}$$

Here, h denotes the height of the robot top, which can be measured using motion-capture equipment with attached markers in an actual robot system. Meanwhile, α, β, and γ represent roll, pitch, and yaw, respectively, which allow the robot's posture and velocity to be measured using an IMU sensor. Additionally, the angle and angular velocity of each joint can be measured using absolute encoders attached to the joints.

As illustrated in Figs. 2 and 3, the motion of the one-legged robot is divided into three phases, i.e., push-off, flight, and landing, based on the system state changes, with each phase having distinct target height and posture requirements. The reward function is decomposed into multiple components to systematically reflect the varying target motions that the robot must achieve throughout this process. Additionally, the reward function designed to achieve the target height and postural stability required in each phase of the jumping sequence is expressed as follows:

Table 1. Definition of reward function for jumping motion

Reward Name	Reward symbol	Reward function
Velocity Reward	r_v	$e(-\omega_v \sum_{i=x,y} \lVert v_i \rVert^2)$
Height Reward	r_h	$e(-\omega_h(\lVert P_{d,h}-P_h \rVert^2+\lVert R_{d,h}-R_h \rVert^2+\lVert F_{d,h}-F_h \rVert^2))$
Posture Reward	r_p	$\mathrm{e}(-\omega_p \sum_{i=1}^{6} \lVert \theta_{d,i}-\theta \rVert^2)$
CoM Reward	r_c	$e-(\omega_c \sum_{i=x,y} \lVert p_{\mathrm{CoM},i}-p_{\mathrm{foot},i} \rVert^2)$
Action Reward	r_a	$e(-\omega_a \sum_{i=1}^{6} \lVert a_{t,i}-a_{t-1,i} \rVert^2)$

$$R = w_v r_v + w_h w_h + w_p r_p + w_c r_c + w_a r_a \tag{3}$$

Here, r_v represents the velocity reward, r_h the height reward, r_p the posture reward, r_c the center of mass (CoM) reward, and r_a the action constraint reward. The velocity reward contributes to maintaining propulsion, the posture reward ensures rotational stability, and the CoM reward ensures landing stability. The mathematical formulations for each reward are listed in Table 1. The velocity reward is defined as the squared error relative to the target velocity, whereas the posture reward is designed to minimize deviations in the robot's upper-body inclination (roll and pitch). The CoM reward is designed to minimize the distance deviation from the vertical axis.

By adjusting the target values for r_h and r_p (height and posture) within a unified reward-function structure, the specific characteristics of each phase are effectively reflected. This intuitive, systematic design enables efficient learning of the propulsion, postural stability, and balance recovery required in each phase. In Table 1, $P_{d,h}$, $R_{d,h}$, and $F_{d,h}$ correspond to the robot's target height, body length, and foot height for each phase, respectively, as illustrated in Fig. 3; P_h, R_h, and F_h denote the robot's real-time height, body length, and foot height, respectively; and p_{CoM} and p_{foot} represent the positions of the robot's CoM and foot, respectively.

2.3 Transition Model for Phase-wise Motion Switching

In the preceding section, the jumping motion of the robot was classified into three phases, and distinct reward functions were employed in each phase to enable independent learning. However, during motion generation, if the posture of the robot at the moment of phase transition deviates significantly from the initial posture used during training, then the transition may not be smooth, thus resulting in postural instability. Such discrepancies can result in propulsion loss, balance collapse, or landing failure, thereby degrading the overall stability of the jumping motion.

Hence, we propose a skill-transition model that mitigates state discrepancies during transition intervals and ensures smooth policy switching. The architecture of the model is illustrated in Fig. 4. Each phase-specific policy network is trained independently and, upon completion, is sequentially connected by a transition manager. The transition model

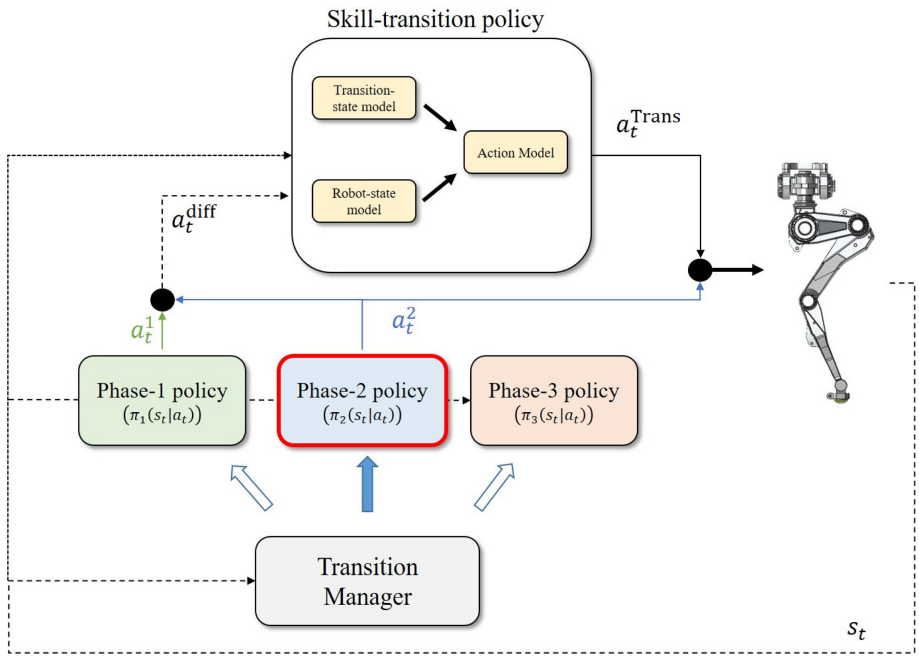

Fig. 4. Schematic diagram of overall system utilizing skill-transition model

encodes the action vector from the policy of the previous phase and the state vector of the current phase to generate an action correction term, a_t^{Trans}, for the transition interval. The skill-transition model receives two primary inputs, which are expressed as follows:

$$o_t^{\text{Trans}} = [a_t^{\text{diff}}, s_t^{\text{switch}}] \in \mathbb{R}^{28} \tag{4}$$

$$o_t^{\text{Robot}} = [a_t^{\text{select}}, s_t] \in \mathbb{R}^{28} \tag{5}$$

Here, a_t^{diff} denotes the difference in actions ($a_t^1 - a_t^2$) output by the policies of the two transitioning phases, s_t^{switch} represents the robot's state information at the moment of transition, and a_t^{select} indicates the action output generated by the policy of the selected phase.

Meanwhile, o_t^{Trans} and o_t^{Robot} are processed through separately designed dedicated encoder models, i.e., the transition-s mand robot-state models, to extract the features. Subsequently, these features are concatenated and input into the action model. Based on the combined features, the action model derives the optimal corrective action, a_t^{Trans}, required during the transition interval. Finally, the corrective action a_t^{Trans} is combined with the action a_t^{select} from the selected phase model to determine the final control action of the robot.

2.4 Training Strategy for Skill-Transition Model

This section introduces a training strategy that utilizes the skill-transition model. The skill-transition model is trained to correct actions for a specified duration precisely when the phase-specific learned policies (π_1, π_2, π_3) transition. As illustrated in Fig. 4, the transition manager, which selects the subsequent model based on the robot's current state, s_t, is designed as follows:

$$n = \begin{cases} 1, & h < P_{h,1} \text{ and } n = 3 \\ 2, & h < P_{h,2} \text{ and } n = 1 \\ 3, & h < P_{h,3} \text{ and } n = 2 \end{cases} \tag{6}$$

$$a_t^{\text{select}} = \pi_n(s_t | a_t) \tag{7}$$

Here, P_h represents the target height of the robot used during the training of each phase. The next phase is determined based on the current phase and the robot's real-time height information. Meanwhile, the robot's action is determined by the policy for the corresponding phase, as shown in Eq. (7).

At the moment of phase transition, the skill-transition model is activated for a specified duration to correct the action and simultaneously obtain the training data for that interval. Data acquisition is repeated across Phases 1–3, and the data from each phase are used to train the skill-transition model. In this study, the activation times were experimentally set to 0.2, 0.5, and 1.0 s for the transitions from Phase 3 to Phase 1, Phase 1 to Phase 2, and Phase 2 to Phase 3, respectively. These durations represent the minimum time required to correct actions stably and allow the policy to adapt to each phase. This phase-transition training was performed in parallel across 100 environments, and by sampling at 0.01 s intervals, we obtained 2,000, 5,000, and 10,000 training data points. Utilizing the obtained data significantly improved the generalization performance and stability at action transitions and across various robot postures.

$$R_{Trans} = R_{phase} + R_{\text{Imitation}} \tag{8}$$

$$R_{\text{Imitation}} = \text{e}(-\sum_{i=1}^{6} \| a_{t,i}^{\text{select}} - a_t^{\text{Trans}} \|^2) \tag{9}$$

Here, R_{Phase} denotes the phase-specific reward function and $R_{\text{Imitation}}$ represents the error between the output of the phase model and that of the skill-transition model. By introducing a reward ($R_{\text{Imitation}}$) that mimics the action of the transitioned phase, the skill-transition model is trained to stably assist the output of the phase model without generating excessive control signals.

3. RESULTS AND DISCUSSIONS

3.1 Simulation Environment

Simulations were conducted to evaluate the performance of the phase-wise modular learning method and the skill-transition model proposed in this study. The proximal policy optimization (PPO) algorithm [22] was employed for the robot's policy training; it is a representative policy optimization method known for providing stable, efficient learning in reinforcement learning. RaiSim [23] was used as the simulation environment, as it enables precise modeling of robot dynamics, supports parallel configuration of environments, and allows control of diverse physical conditions. To verify the performance of the phase-wise learning and the Skill Transition Model, 100 parallel simulations were executed. This approach enabled the effective collection of phase-wise training data and transition-interval data, facilitating a systematic evaluation of the model's generalization performance and stability.

3.2 Results of Jumping Motion via Modular Learning

The jumping motion of the one-legged robot implemented using the proposed phase-wise modular learning method is illustrated in Fig. 5. Fig. 6 presents the training results for each phase-specific model, which show that the reward values converged rapidly. Additionally, the skill-transition model trained

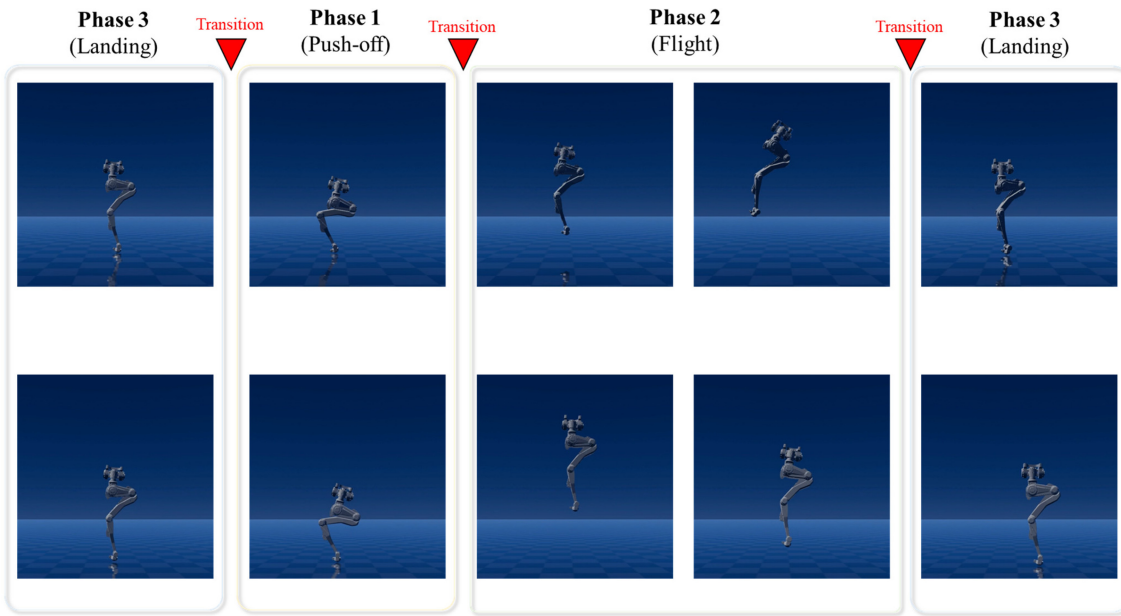

Fig. 5. Simulation results of robot's jumping motion (top: without skill-transition model; bottom: with skill-transition model)

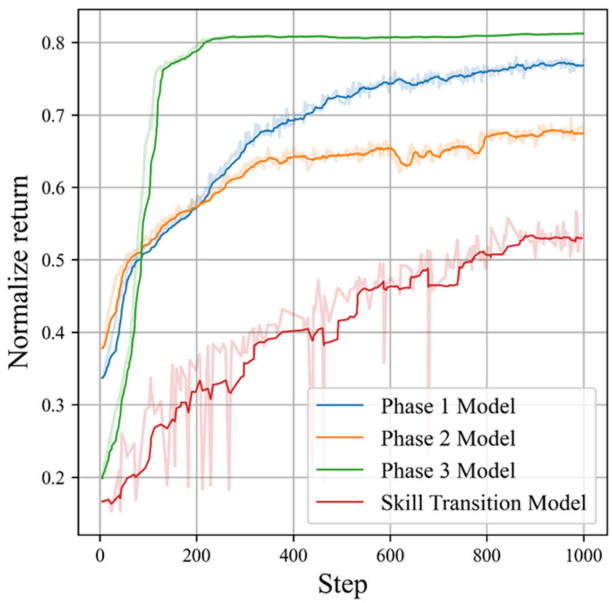

Fig. 6. Training results for each model

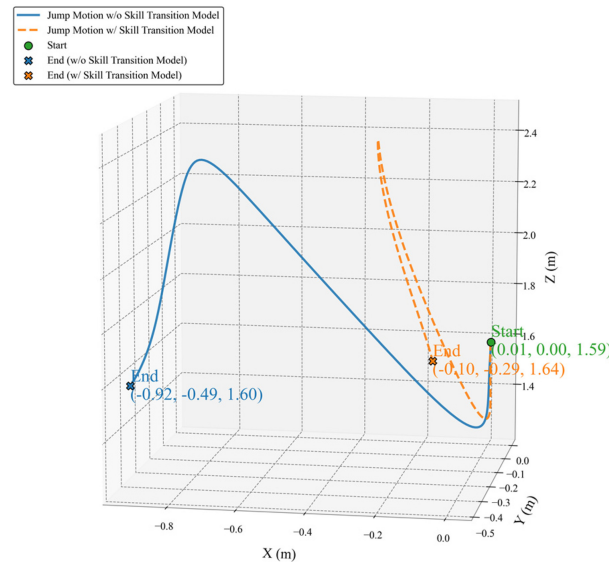

Fig. 7. Three-dimension results of robot's jumping motion

based on these phase models exhibited stable learning. The maximum reward for all models was normalized to 1. However, the reward for the skill-transition model was lower than that for the phase models because of the relatively short transition interval, which resulted in a smaller accumulated reward. The motions learned by each phase model were combined sequentially to constitute a single complete jumping motion.

The upper section of Fig. 5 shows the results obtained only with transitions between phase models, i.e., without the skill-transition model, whereas the lower section shows the results with the skill-transition model applied. In the absence of the skill-transition model, while the basic jumping motion was achieved, the robot's posture exhibited discontinuous changes during transition intervals, resulting in instability. Conversely, when the skill-transition model was applied, this unstable transition was mitigated by reducing the postural instability observed during the flight phase. These results indicate that the skill-transition model effectively maintains a stable posture throughout the jumping motion by smoothly correcting action discrepancies at the moment of phase transition.

To verify these results more intuitively, Fig. 7 presents the

robot's top-coordinate trajectory in three dimensions. Without the skill-transition model, the robot was able to perform the jumping motion; however, the landing point deviated significantly from the target position. An analysis of the actual landing coordinates revealed an average error of x = -0.92 m and y = -0.49 m relative to the target point. By contrast, when the skill-transition model was applied, instability during the transition process was reduced significantly, thus resulting in a landing position that closely aligned with the target point (average error x = -0.1 m, y = -0.29 m). These results indicate that the skill-transition model effectively corrects the discontinuous motion changes that occur during phase transitions, thereby aiding the robot in performing more stable and accurate jumping motions.

3.3 Analysis of Jumping Motion Under Disturbances

To verify the robustness of the skill-transition model, experiments were conducted by artificially introducing disturbances to the robot's initial posture and velocity errors, as listed in Table 2. These experiments aimed to evaluate the ability of the model to maintain a stable motion when unexpected environmental changes or uncertainties occur in the initial conditions.

Fig. 8 shows the results obtained without applying disturbances. In the absence of disturbances, both cases, i.e., with and without the skill-transition model, successfully demonstrated jumping motions close to the target height of 2.5 m and maintained a generally stable posture. This indicates that basic phase-wise learning alone is sufficient to acquire the primary patterns of the jumping motion.

Table 2. Initial joint, posture, and velocity disturbance values

	Joint Angle (°)	Robot Angle (°)	Velocity (m/s)
Initial Error	3.82	1.37	0.46
Disturbance	±1.758	±2.291	±0.407

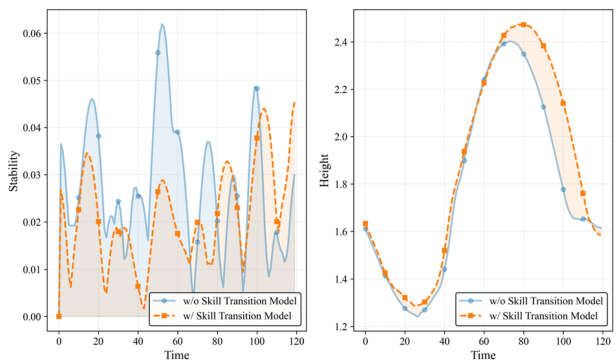

Fig. 8. Analysis results of robot's jumping motion without disturbance

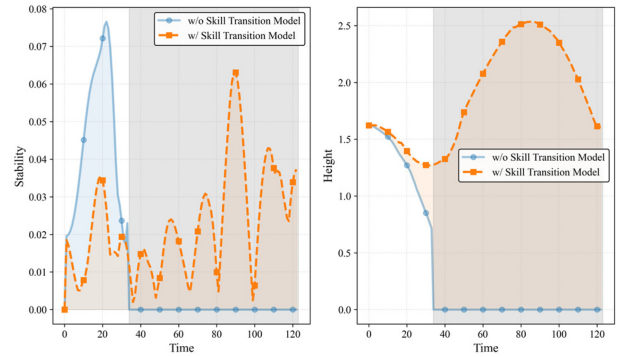

Fig. 9. Analysis results of robot's jumping motion under disturbance

However, when disturbances were introduced to the initial posture and velocity under the same conditions, the robot without the skill-transition model exhibited severe postural instability during the phase-transition intervals, as shown in Fig. 9. Specifically, the discontinuity in the transition interval caused the robot to lose its balance and fall, thus preventing the simulation from proceeding normally. These results imply that, in the presence of disturbances, the lack of smooth phase transitions causes the stability of the overall motion to degrade rapidly.

By contrast, when the skill-transition model was applied, the postural instability during the transition intervals reduced significantly, even under disturbances. This shows that the skill-transition model effectively corrects motion discontinuities across phases and enhances the resilience against disturbances, thus ultimately enabling a more stable and consistent performance.

3.4 Analysis of Skill-Transition Model Outputs

We analyzed the manner by which the skill-transition model corrected the motion of the robot during each phase-transition interval. Fig. 10 shows the correction amounts applied by the skill-transition model to the phase-specific policy outputs for each joint. These results confirmed that the skill-transition model selectively affected different joints during each transition interval, thereby ensuring motion continuity and stability.

First, during the transition from Phase 3 to Phase 1, the correction occurred primarily on the third joint. This suggests that the model fine-tuning the movement of this joint and its surrounding joints to maintain balance while lowering the body as the robot transitions from landing back to the jump-preparation posture.

Next, during the transition from Phase 1 to Phase 2, as the robot pushes off the ground to jump, the major joints

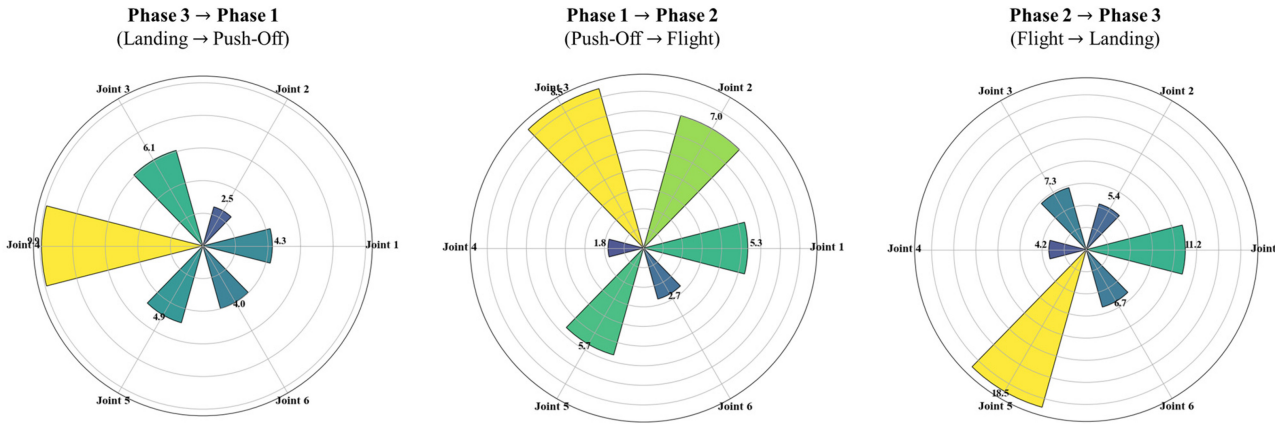

Fig. 10. Joint-wise correction results of skill-transition model

responsible for upper-body stabilization and propulsion were primarily corrected. This correction aids efficient propulsion transmission by maintaining the directionality and balance of the jump.

Conversely, during the transition from Phase 2 to Phase 3, as the robot entered the landing phase, the most significant correction was applied to the fifth joint (knee). This represents an adjustment motion designed to absorb the impact upon landing and rapidly stabilize posture, thus demonstrating that the skill-transition model effectively performs posture control against external impacts.

These findings suggest that the skill-transition model extends beyond merely connecting outputs across phases. In fact, it enhances the continuity and stability of the entire jumping motion by performing adaptive corrections at each joint based on the motion characteristics of each transition interval.

4. CONCLUSIONS

In this study, we implemented the jumping motion of a robot using an RL-based phase-wise modular learning method. The proposed method allows independent learning for each phase, thereby enabling phase-specific optimization and efficient training. Notably, it achieved stable performance more rapidly than conventional curriculum-learning approaches. This approach ensured that each phase module individually achieved high performance while maintaining accuracy and stability when integrated into the complete motion sequence.

Furthermore, a skill-transition model was introduced to address the discontinuities and postural instabilities that may arise during phase transitions. This model corrects the phase discrepancies during transitions, thereby facilitating the robot in performing more stable and continuous jumping motions. The experimental results confirmed that the skill-transition model provided adaptive corrections to key joints during transition intervals, thus significantly enhancing the continuity and stability of the overall jumping motion.

Consequently, the combination of the proposed phase-wise modular learning and skill-transition model was proven to be an effective approach for the efficient and stable learning of complex robot motions, i.e., extending beyond simple jumps. Additionally, this study demonstrated the potential of these methods to be extended to control various complex motions based on phase transitions, including walking, landing, and obstacle avoidance, instead of being limited to jumping.

CRediT Authorship Contribution Statement

Hyeonjin Kim: Methodology, Software, Validation, Writing – original draft**. Jinhyun Kim:** Conceptualization, Supervision, Writing – review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (No. RS-2022-NR070333).

REFERENCES

[1] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, et al., Planning walking patterns for a biped robot, IEEE Trans. Robot. Autom. 17 (2001) 280–289.

[2] J. Kulk, J. Welsh, A low power walk for the NAO robot, Proceedings of the Australasian Conference on Robotics & Automation (ACRA), Canberra, Australia, 2008, pp. 1–7.

[3] D.C. Kar, Design of statically stable walking robot: a review, J. Robot. Syst. 20 (2003) 671–686.
[4] J. Reher, A.D. Ames, Dynamic walking: Toward agile and efficient bipedal robots, Annu. Rev. Control Robot. Auton. Syst. 4 (2021) 535–572.
[5] P.A. Bhounsule, J. Cortell, A. Ruina, Design and control of ranger: an energy-efficient, dynamic walking robot, Adaptive Mobile Robotics. (2012) 441–448.
[6] J. Reher, W.-L. Ma, A.D. Ames, Dynamic walking with compliance on a cassie bipedal robot, Proceedings of 2019 18th European Control Conference (ECC), Naples, Italy, 2019, pp. 2589–2595.
[7] T. Hemker, M. Stelzer, O. von Stryk, H. Sakamoto, Efficient walking speed optimization of a humanoid robot, Int. J. Robot. Res. 28 (2009) 303–314.
[8] K.H. Koch, K. Mombaur, P. Soueres, Optimization-based walking generation for humanoid robot, IFAC Proc. Vol. 45 (2012) 498–504.
[9] X.B. Peng, P. Abbeel, S. Levine, M. Van de Panne, Deepmimic: Example-guided deep reinforcement learning of physics-based character skills, ACM Trans. Graph. 37 (2018) 1–14.
[10] X.B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, S. Levine, Learning agile robotic locomotion skills by imitating animals, ArXiv., https://arxiv.org/abs/2004.00784 (2020).
[11] R. Bussola, M. Focchi, G. Turrisi, C. Semini, L. Palopoli, Guided reinforcement learning for omnidirectional 3d jumping in quadruped robots, ArXiv., https://arxiv.org/abs/2507.16481 (2025).
[12] V. Atanassov, J. Ding, J. Kober, I. Havoutis, C. Della Santina, Curriculum-Based Reinforcement Learning for Quadrupedal Jumping: A Reference-free Design, ArXiv., https://arxiv.org/pdf/2401.16337 (2024).
[13] R. Soni, D. Harnack, H. Isermann, S. Fushimi, S. Kumar, F. Kirchner, End-to-end reinforcement learning for torque based variable height hopping, Proceedings of 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, USA, 2023, pp. 7531–7538.
[14] K. Frans, J. Ho, X. Chen, P. Abbeel, J. Schulman, Meta learning shared hierarchies, ArXiv., https://arxiv.org/abs/1710.09767 (2017).
[15] W. Yu, F. Acero, V. Atanassov, C. Yang, I. Havoutis, D. Kanoulas, et al., Discovery of skill switching criteria for learning agile quadruped locomotion, ArXiv., https://arxiv.org/abs/2502.06676 (2025).
[16] C. Zhang, W. Zou, N. Cheng, S. Zhang, Towards Jumping Skill Learning by Target-guided Policy Optimization for Quadruped Robots, Mach. Intell. Res. 21 (2024) 1162–1177.
[17] J. Zhang, M. Li, J. Cao, Y. Dou, X. Xiong, Research on bionic jumping and soft landing of single leg system in quadruped robot, J. Bionic Eng. 20 (2023) 2088–2107.
[18] Q. Liu, D. Xu, B. Yuan, Z. Mou, M. Wang, Distance-controllable long jump of quadruped robot based on parameter optimization using deep reinforcement learning, IEEE Access 11 (2023) 98566–98577.
[19] J. Qi, H. Gao, H. Su, L. Han, B. Su, M. Huo, et al., Reinforcement learning-based stable jump control method for asteroid-exploration quadruped robots, Aerospace Sci. Technol. 142 (2023) 108689.
[20] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, K. Sreenath, Robust and versatile bipedal jumping control through reinforcement learning, ArXiv., https://arxiv.org/abs/2302.09450 (2023).
[21] R. Bussola, M. Focchi, A. Del Prete, D. Fontanelli, L. Palopoli, Efficient reinforcement learning for 3d jumping monopods, Sensors 24 (2024) 4981.
[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, ArXiv., https://arxiv.org/abs/1707.06347 (2017).
[23] J. Hwangbo, J. Lee, M. Hutter, Per-contact iteration method for solving contact dynamics, IEEE Robot. Autom. Lett. 3 (2018) 895–902.