FRASA: An End-to-End Reinforcement Learning Agent for Fall Recovery and Stand Up of Humanoid Robots

Clément Gaspard[1*], Marc Duclusaud[1*], Grégoire Passault[1], Mélodie Daniel[1], Olivier Ly[1]

***Abstract*— Humanoid robotics faces significant challenges in achieving stable locomotion and recovering from falls in dynamic environments. Traditional methods, such as Model Predictive Control (MPC) and Key Frame Based (KFB) routines, either require extensive fine-tuning or lack real-time adaptability. This paper introduces FRASA, a Deep Reinforcement Learning (DRL) agent that integrates fall recovery and stand up strategies into a unified framework. Leveraging the Cross-Q algorithm, FRASA significantly reduces training time and offers a versatile recovery strategy that adapts to unpredictable disturbances. Comparative tests on Sigmaban humanoid robots demonstrate FRASA superior performance against the KFB method deployed in the RoboCup 2023 by the Rhoban Team, world champion of the KidSize League.**

I. Introduction

Humanoid robotics has made significant strides in recent years, driven by advancements in both hardware and machine learning. One of the key challenges in this field is developing robots that can autonomously perform complex tasks, including locomotion, in dynamic and unpredictable environments [1]. This is the case in several humanoid robotics competitions or challenges, such as RoboCup [2], the DARPA Robotics Challenge [3], and the recent Avatar XPRIZE [4]. Reinforcement Learning (RL) has emerged as a powerful tool for addressing these challenges, enabling robots to learn from interactions with their environment and have more adaptive responses [5].

Achieving stable locomotion is one of the primary obstacles, if not the greatest, in humanoid robotics. The difficulty of this task arises primarily from the limited number of ground contact points for a humanoid, which results in an inherently unstable standing pose [6]. Moreover, a high Center of Mass (CoM) increases the risk of damage in the event of a fall for humanoid robots [7]. This leads researchers to prioritize fall prevention over recovery.

A widely adopted approach in humanoid robotics to achieve stable walking and prevents falls is the use of Model Predictive Control (MPC) [8]. This method relies on planning the trajectory of the CoM while ensuring stability criteria over a finite time horizon. The trajectory is, then, frequently updated and adjusted based on sensor feedback [9].

When significant deviations from the planned CoM trajectory occur, such as due to an external disturbance, three solutions are generally presented in push recovery problems: modifying the center of pressure using ankle joints, generating angular momentum with hip joints, or taking a step [10] [11]. One usually uses the capture point to determine the good strategy [12]. The CP is a specific measure derived from the linear inverted pendulum model [13], corresponding to the location where a robot should step to halt its linear momentum and eventually come to a stop. While effective, this method is constrained by arbitrary adjustment choices, strong linearity assumptions, and significant computational complexity.

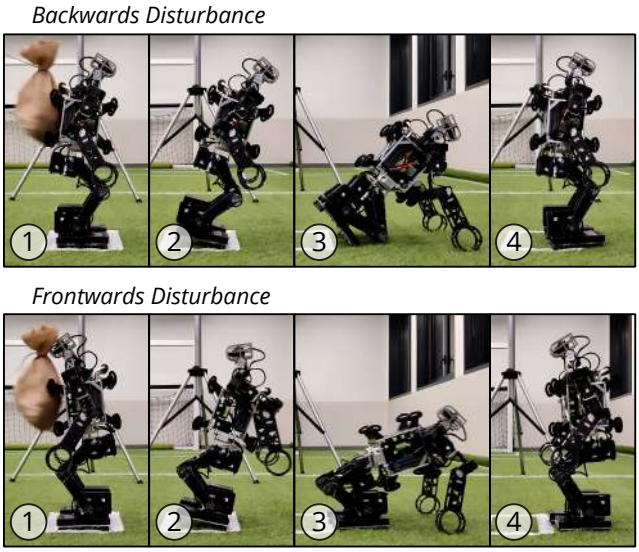

Fig. 1. FRASA adaptative response to a backwards and a frontwards disturbances on the Sigmaban platform. The recovery behavior using the arms accelerates the return to a stable position while minimizing the risk of damage.

More recently, other approaches involving training Deep Reinforcement Learning (DRL) agents have emerged to address push recovery problems [14] [15] [16]. These methods address the drawbacks of MPC-based approaches but typically require extensive training times. Moreover, these approaches do not offer recovery solutions after a fall.

This paper does not focus on the problem of push recovery but rather on **fall recovery**, which involves creating new points of contact with the ground as a strategy to regain stability in minimal time. This approach is crucial when external forces are too strong on the robot to prevent a fall. In robotics competitions [2] [3] [4], **fall recovery** is usually handled by allowing the fall to complete before initiating a **stand-up** routine, which is often based on key frame animations [17]. This method is simple to implement but lacks versatility, requires extensive fine-tuning, and must be

[1]Univ. Bordeaux, CNRS, LaBRI, UMR 5800, 33400 Talence, France. Corresponding author: Clément Gaspard, e-mail: clement.gaspard@u-bordeaux.fr.

This study has received financial support from the French government in the framework of the France 2030 program, Initiative of Excellence (IdEx) University of Bordeaux / RRI ROBSYS and from the ENS Rennes.

* These authors contributed equally.

frequently adjusted due to motor wear and loss of precision.

Another approach proposes using genetic algorithms to optimize the parameters of stand up splines [18]. This method offers the advantage of generalizing a stand up routine across different humanoid architectures. However, it remains an open-loop animation, limiting real-time adaptability to unexpected disturbances.

Recent RL-based studies tackle the stand up challenge. A Q-learning approach in [19] incorporates environment feedback but relies on complex problem discretization and lacks fall recovery. Another method trains DRL agents to learn stand up motions [20], but it still depends on expert key frames and requires approximately one day of training.

In this paper, we propose a Fall Recovery And Stand up Agent (**FRASA**) that integrates both fall recovery and stand up strategies into a single agent, aiming to resume walking as fast as possible after a disturbance. The adaptative response allowed by FRASA in the case of backwards and frontwards disturbances is presented in Fig. 1.

The key contributions of FRASA are:

- **Unified task handling**: The proposed reward function enables the DRL agent to efficiently address both fall recovery and stand up tasks within a unified framework.
- **Reduced training time**: By leveraging the CrossQ algorithm, FRASA achieves effective training in significantly less time compared to existing RL approaches [20].
- **Versatile and adaptative recovery strategy**: The use of a DRL agent allows adaptation to unpredictable real-world feedback and recovery from various postures without the need for expert tuning.

Tests are conducted on Sigmaban humanoid robots [21] to compare FRASA with a Key Frame Based (KFB) approach [17]. The KFB method used for comparison was deployed in the RoboCup 2023 by the Rhoban Team, world champion of the KidSize League. Real robots experiments are presented in the following video[1] and the open-source code for this project is available on our GitHub repository[2] for further use and contributions.

II. Problem Statement

The problem we aim to solve is twofold: first, the ability to stand up from a prone or supine position, and second, to recover when a fall is about to occur. The fall recovery problem necessitates creating new contact points with the ground in cases of strong or repeated disturbances. In both cases, the objective is to resume movement, specifically walking, as quickly and smoothly as possible.

Let us note that, the humanoid platform's design allows it to naturally roll into prone or supine positions after a fall. In addition, reaching these positions after a fall is generally a feasible task for humanoids. Consequently, the stand up problem can be restricted to these two fall configurations.

[1]https://youtu.be/NL65XW0O0mk

[2]https://github.com/Rhoban/frasa

III. Background

A. RL and DRL

RL is a machine learning field where an agent learns to make decisions by interacting with its environment to maximize cumulative rewards, typically modeled as a Markov Decision Process (MDP). It is represented by the tuple $(\mathcal{S}, \mathcal{A}, P, R)$ [22], where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition function, and R is the reward function. The agent's goal is to find an optimal policy that maximizes the expected episode reward, represented as a discounted sum of future rewards.

DRL uses deep neural networks to approximate the action-value function and the policy, particularly through actor-critic architectures [23]. The critic estimates the value of state-action pairs using temporal difference learning, while the actor updates the policy to maximize this value, enhancing exploration through added noise. Recent DRL advancements have improved policy and action-value function approximations, enabling effective solutions for complex decision-making tasks in high-dimensional environments [23] [24].

B. CrossQ Algorithm

Training DRL agents is often time-consuming due to complex environments. A recent algorithm, CrossQ, improves sampling efficiency and reduces training time by removing target networks, normalizing batches, and using wider critic layers [25]. It can be integrated with traditional actor-critic architectures, like Soft Actor-Critic (SAC) [23].

Empirical results show that CrossQ matches or surpasses state-of-the-art algorithms in sample efficiency while significantly reducing computational costs [25], making it a promising approach for continuous control tasks in DRL.

IV. Method

In this paper, we present FRASA: an end-to-end DRL approach for humanoid robot autonomous stand up and fall recovery from external disturbances. The agent is trained in a full-body physics simulator [26], incorporating a robot's detailed collision model. In simulation, the robot's degrees of freedom (DoFs) are position controlled to reflect servomotors behavior. The core contribution of FRASA lies in formulating an environment that enables efficient learning with actor-critic algorithms, such as SAC and, more recently, CrossQ.

The RL environment leverages the inherent symmetrical design of the robot to simplify the learning process. To this end, only 5 of the robot's DoFs are observed and controlled, as shown in Fig. 2. This selection is strategic, as these are the primary DoFs involved in the plane of movement (x, z).

The target posture for the recovery movement is the initial stance associated with walking, noted as neutral pose in Fig 2. This target pose is defined by the angles of the controlled joints q_{target} and the robot's trunk pitch θ_{target} :

$$\psi_{\text{target}} = [q_{\text{target}}, \theta_{\text{target}}] \tag{1}$$

By adjusting the target posture, it is possible to modify the subsequent movement after recovery, enhancing the versatility of the solution.

A. Environment initialization

At the start of each episode, the robot's joint angles q and trunk pitch θ are set to random values within their physical limits. The robot is then released above the ground, simulating various fallen states. The physical simulation is then stepped using current joints configuration as targets, until stability is reached, at which point the learning process begins.

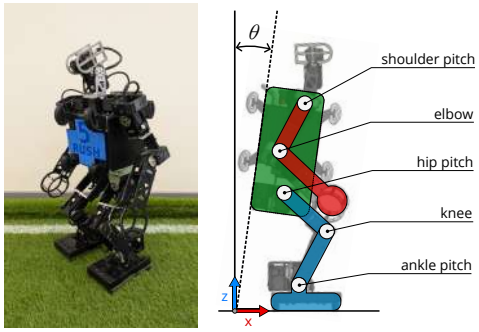

Fig. 2. **Left**: target posture for the recovery movement. **Right**: pose vector ψ_{target} components, including the trunk pitch θ and the 5 explicited DoFs.

This approach prevents overfitting to specific starting positions and promotes exploration of a wide range of configurations, enabling the development of a general policy effective across diverse scenarios. Furthermore, the robot's state is occasionally initialized close to the neutral pose presented in Fig. 2 with a certain probability, p_s. This strategy provides occasional large rewards, reinforcing successful standing behaviors and thus speeding up the learning process.

B. Action Space

As this is an end-to-end model, the action space is the control commands that can be symmetrically applied to the joints on both sides of the robot. The action space is defined as a variation of the desired joints configuration :

$$a_t = \dot{q}_t^{\text{desired}} \tag{2}$$

During each simulation time step t, the desired joint angles q_t^{desired} transition from their initial values to $q_{t+1}^{\text{desired}} = q_t^{\text{desired}} + \Delta t \times \dot{q}_t^{\text{desired}}$, reflecting the incremental control applied by the agent.

C. State Space

The state space captures the robot's posture and movements at each time step. It is designed as

$$s_t = \left[q_t, \dot{q}_t, q_t^{\text{desired}}, \theta_t, \dot{\theta}_t, a_{t-1}, a_{t-2}, \ldots, a_{t-k}\right], \tag{3}$$

where q_t and $\dot{q}_t$ are the joint angles and velocities of the 5 DoFs defined in Fig. 2. They are determined as the average values of the corresponding actuators on both sides of the robot for each degree of freedom. θ_t and $\dot{\theta}_t$ are respectively the trunk pitch and its rate of change. Given (2), $a_{t-1}, a_{t-2}, \ldots, a_{t-k}$ represent the target joint position update rates issued by the agent over the last k time steps.

D. Reward

The reward function is designed to encourage the robot to achieve and maintain the target upright posture (cf. Fig. 2). Our reward function is designed as followed with the different components defined in Table I :

$$R = R_{\text{state}} + R_{\text{variation}} + R_{\text{collision}} \tag{4}$$

In R_{state}, the current state $\psi_t = \left[q_t, \theta_t\right]$ is compared to the target state ψ_{target} defined in (1). A Gaussian function shapes R_{state}, providing higher rewards as the robot's state approaches the desired target. The parameter w_1 shapes the reward, determining how sharply it decreases as the robot's state deviates from the desired state, thus encouraging the robot to achieve and maintain the desired posture. $R_{\text{variation}}$ penalizes high control velocities and large variations in actions, promoting smoother movements. $R_{\text{collision}}$ discourages self-collisions, determined using the physics simulator.

TABLE I
COMPONENTS OF THE REWARD FUNCTION

Reward components	**Formulas**
Desired state proximity reward	$R_{\text{state}} = \exp(-w_1 \times \|\psi_t - \psi_{\text{target}}\|^2)$
Action variation reward	$R_{\text{variation}} = w_2 \times \exp(-\|a_t - a_{t-1}\|)$
No self-collision reward	$R_{\text{collision}} = w_3 \times \exp(-\text{self\_collisions})$

The weights $0 \leq w_2, w_3 \ll 1$ are small penalizations intended to smooth learning and prevent the robot from adopting mechanically feasible but undesirable strategies.

When the robot reaches the desired position, the reward is approximately 1 thanks to R_{state}, providing a clear unit of measure for the agent's performance in terms of maintaining upright posture over time.

E. Episode termination and truncation

An episode termination occurs when the robot reaches an unsafe or irreversible state:

- Trunk pitch (θ) exceeds a predefined threshold, indicating the robot is upside down.
- Trunk pitch velocity ($\dot{\theta}$) surpasses a critical threshold, suggesting very violent displacements of the pelvis.

An episode is truncated if its duration exceed a predefined maximum duration. This allows for an increase in the diversity of encountered states.

F. Domain Randomization

To enhance the robustness and generalization of the learned policy, we employ extensive domain randomization [27] in our simulation environment. The following parameters are randomized:

- Angular errors, at the beginning of each episode, to simulate the fact that parts could be slightly deformed or zero positions slightly incorrect.
- The mass and center of mass positions of the robot's body parts to ensure that the agent could handle variations in its own weight distribution.
- The friction coefficients of the contact surfaces to account for different types of ground interactions the robot might encounter.

- The battery voltage supply, affecting the proportional gain and maximum available torque of the actuators, simulating fluctuations in power that might occur in real-world conditions.
- The friction parameters of the actuators themselves, such as dry friction and damping, to reflect the wear and variability of mechanical components over time.

Table II provides the specific parameters and their ranges used for domain randomization. The randomization ensures that the policy learned by the robot is robust and adaptable to a wide range of real-world scenarios.

The DRL algorithm operates in an end-to-end manner, where raw state inputs are mapped directly to action outputs without any intermediate processing steps.

V. Experiments

To validate the performance of FRASA in a real environment, we deploys it on the Sigmaban robotic platform.

A. Robotic Platform

The Sigmaban is a 70 cm-tall humanoid robot weighing 7.5 kg [28]. Its limbs are milled from aluminum and it is actuated by MX-106 and MX-64 Dynamixel servos with proportional position control. The robot features a camera, an Inertial Measurement Unit (IMU), and eight foot pressure sensors using strain gauge [29]. On-board computations are handled by an Intel NUC (Core i5-7260U, 8 GB RAM) running Ubuntu 22.04 and a custom STM32-based board.

B. Physical Simulation and Modeling

The physics engine used for training is MuJoCo [26]. A precise CAD model of the robot is exported to MuJoCo [30], ensuring accurate geometry, mass distribution, and joint limits, closely matching the real robot.

It is worth noting that in order to manage contacts effectively in the simulated environment, every part of the robot is approximated using basic shapes such as cylinders, cuboids, or combinations thereof. In total, the robot is approximated by 51 boxes and 14 cylinders, the arrangement of which is shown in Fig. 3.

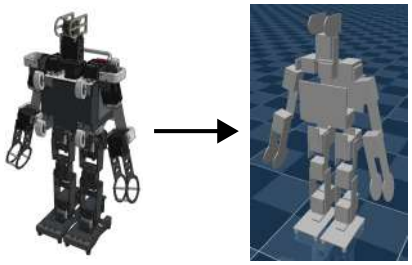

Fig. 3. Approximation of the robot using primitive shapes dedicated to simulating collisions

To ensure a realistic behavior of the actuators in simulation, an identification of their friction parameters, such as dry friction and damping, is performed using the CMA-ES genetic algorithm on logs generated from a test bench. The method used is the same as in [31].

C. Training and Inference

The agent is trained using CrossQ [25] applied to SAC DRL algorithm [23], implemented in Stable Baselines X, a GPU-accelerated version of Stable Baselines 3 [32]. Utilizing the enhanced computational efficiency of this implementation, the training process is remarkably fast, resulting in a fully trained network within only 13 to 37 minutes, with a total of 235,000 to 575,000 timesteps. The training is conducted on a PC equipped with an Intel Core i9-12900 CPU, an NVIDIA GeForce RTX 3060 GPU, 64 GB of RAM, and an SSD.

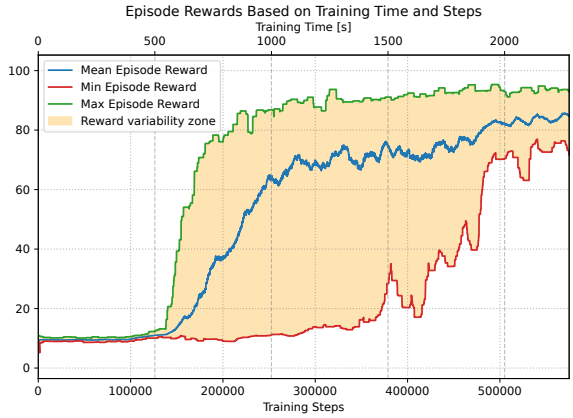

Fig. 4. Reward variability observed during the training of 40 distinct FRASA agents, each trained over 575,000 steps within 37 minutes using the CrossQ+SAC algorithm.

Fig. 4 illustrates the reward progression during the evaluation phase of 40 trained agents of FRASA. The "reward variability zone" highlights the range of rewards experienced across the 40 evaluation sessions. The fastest agents learn to stand up in 13 minutes, with convergence toward a consistent reward observed across all networks after approximately 30 minutes of training. While some agents are better trained than others, this variability is expected and does not detract from the overall efficiency and reliability of the training process.

The hyperparameters used to optimize the learning process include a learning rate set to 0.001, and a large buffer size of 1M transitions to store past experiences. Each training batch consists of 256 samples, and the training utilizes 16 parallel environments to enhance data collection efficiency. The discount factor is set to 0.99, ensuring that the agent considers long-term rewards. Training updates are applied every 4 steps, with 512 gradient descents per update, enabling the model to adjust its parameters 512 times per batch. Training starts after 100,000 steps, ensuring a diverse buffer. The policy network comprises two hidden layers containing 384 and 256 neurons respectively.

To exploit the maximum capacities of our computational power during runtime, we use the open source OpenVINO™ Runtime [33]. The mean on-board inference time to generate the actuators command for one timestep is $40\mu s$.

D. Sim-to-real Transfer

In addition to the precise modeling of the robot in MuJoCo, sensor communication delays, such as those from the IMU and the actuators, are measured on the real robot and modeled

in the simulation. They are crucial to ensure a good transfer to the real robot.

Furthermore, extensive domain randomization is employed during training (see Section IV). Table II lists the specific parameters and their randomization ranges.

TABLE II

PARAMETERS USED FOR SIM-TO-REAL AND DOMAIN RANDOMIZATION

Parameter	Initial Value	Randomization Range
Floor friction coefficient	0.875	± 0.625
Trunk mass	3.28kg	$\pm 20\%$
Trunk CoM position offset	(0, 0, 0)	$\pm 5 \times 10^{-3}$ m
Battery voltage	15 V	[13.8, 16.8] V
Actuators proportional gain*	12.5 / 21	Scaled by battery voltage
Actuators abs. force bound*	5 / 8 N.m	Scaled by battery voltage
Actuators damping*	0.66 / 1.7 N.m	$\pm 20\%$
Actuators friction loss*	0.09 / 0.1 N.m	$\pm 20\%$
Actuator position (q) offset	0	$\pm 1.5°$
Actuator velocities ($\dot{q}$) delay	0.030 s	N/A
Trunk pitch (θ) delay	0.050 s	N/A
Trunk pitch velocity ($\dot{\theta}$) delay	0.050 s	N/A

(*) : MX-64 motor value / MX-106 motor value

During the initial transfers, significant tremors appeared on the robots. The source of these tremors is a substantial discrepancy between the actuators simulated in MuJoCo and the real behavior of the motors. Specifically, the relatively simple friction model used by the simulator seems to not adequately capture the real behavior.

To solve this problem the motor gains in both the simulation and the real robot is reduced to mitigate the unforeseen dynamic effects caused by the significant gap between the simulated and real environments. Some works use downstream low-pass filters [34], which was unnecessary in our case but could lead to the same effects.

Increasing the inference frequency of the agent also help to reduce tremors. The agent is trained with a decision frequency of 20Hz and operates at 100Hz on the actual robot. The nature of the action space allows for this increase without necessitating significant changes.

E. Stand Up Experiment

A first experiment is conducted to compare the ability to stand up from prone and supine positions using both a KFB recovery process and FRASA. A KFB stand up method defines arbitrary key positions determined thanks to expert knowledge for the robot to pass through, interpolating motor positions between them to create a stand up trajectory [17]. The KFB method used in the experiments was developed for RoboCup 2023 by the Rhoban Team and demonstrated both its effectiveness and efficiency [28].

The robot is placed on its back, face down, aiming to stand and begin walking. The metric studied is the time from movement initiation to the start of walking. To initiate walking, the error in the state vector components—comprising the 5 controlled joints and the trunk pitch—must remain below 5° for 0.5 seconds.

F. Fall Recovery Experiment

A second experiment compares the recovery ability of the KFB process and FRASA after being pushed from a static standing pose. The setup, shown in Fig. 5, uses a pendulum mechanism to release a mass from varying distances onto the robot in a repeatable manner. A cord allows retrieval of the weight post-impact, preventing interference during recovery.

The length L of the pendulum and the height H of the stand are respectively 1.28m and 1.74m and the mass m of the weight is 0.9kg. Assuming the pendulum mass is point-like, the kinetic energy of the mass at the moment of impact varies from 4J to 7.3J across the different tested configurations, which are chosen to produce impacts ranging from mild disturbances to guaranteed falls. During the experiments, the velocity of the mass is zero after impact, indicating a complete transfer of energy to the robot.

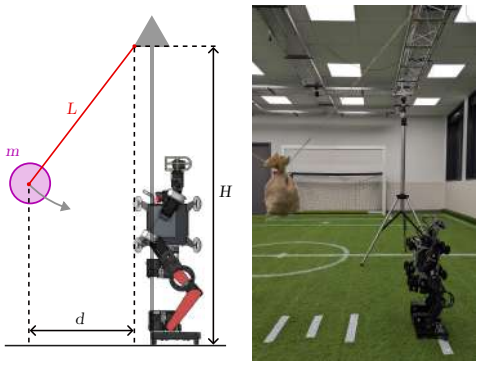

Fig. 5. Experimental setup for inducing repeatable disturbances

In this experiment, the robot is placed under the pendulum and subjected to repeated disturbances with varying impact intensities. After a disturbance, the robot waits to return to a stable position close to the target posture (cf. Fig. 2) before starting to walk in place.

The metric studied is the duration of disturbance rejection. Time measurement begins when any value in the state vector deviates by more than 5° from the neutral walking posture values. Once in this unstable state, all values in the state vector must have an error of less than 5° relative to the target posture for 0.5 seconds for the robot to be considered stable again and able to initiate walking. The time measurement ends at the beginning of the walking phase.

VI. RESULTS

A. Qualitative results

The rapid training of the agents allows for multiple candidates to be trained to select the optimal FRASA candidate. Despite comparable training times and rewards, the behaviors that emerged are sometimes different. Fig. 6.A shows an example of variation in the forward stand up task, where the robot either chose to squat or to remain with legs extended during stand up.

The extensive agent environment exploration sometimes results in highly dynamic solutions, which proved feasible to implement on the real robot. One such solution is presented in Fig 6.B. Although functional, the mechanical stresses and probable wear on the robots from performing this motion make it undesirable.

The emergence of such varied behaviors leads to the selective choice of a candidate for FRASA. The selected candidate, whose reaction to disturbances of different intensities is

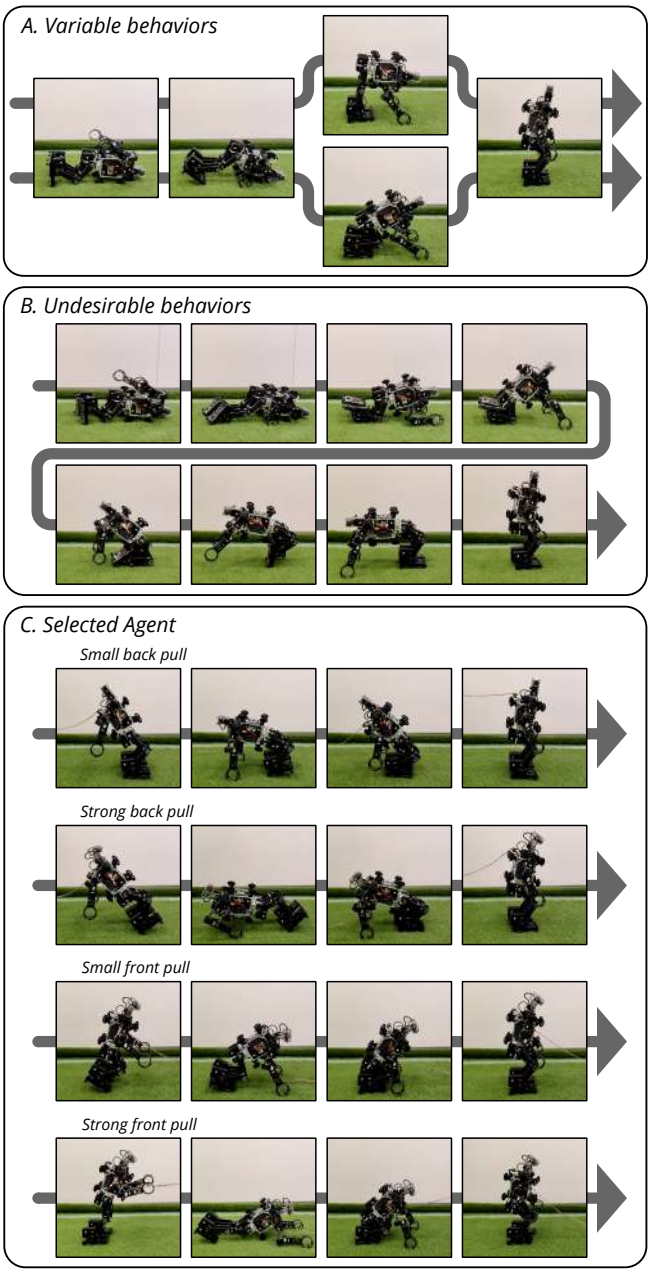

Fig. 6. Agents behaviors on real robot

shown in Fig. 6.C, is chosen based on the following criteria: speed of recovery, robustness of movements, mechanical safety, and versatility of behaviors.

B. Quantitative results

The stand up experiment is conducted by performing the stand up task 20 times for each side and each method. The average stand up times are presented in Table III. We observe that FRASA outperforms the KFB method in both possible configurations: FRASA completes the stand up from the supine position in 53% and from the prone position in 68% of the time required by the KFB method. This corresponds to average gains of 2.38 seconds and 1.02 seconds, respectively.

The results of the fall recovery experiment are detailed in Table IV. Each configuration of distance, method, and side is repeated 10 times, and the mean instability time is calculated.

TABLE III

COMPARISON OF AVERAGE STAND UP TIMES FOR FRASA AND KFB METHOD FROM PRONE AND SUPINE POSITIONS

	Prone position		**Supine position**	
Method	FRASA	KFB	FRASA	KFB
Average Time / s	**2.135** ± 0.042	3.154 ± 0.005	**2.678** ± 0.178	5.06 ± 0.008

The standard deviation is represented using the notation $\pm$

TABLE IV

COMPARISON OF AVERAGE INSTABILITY DURATIONS OF FRASA AND THE KFB METHOD AFTER DISTURBANCES

	Frontwards Disturbance			**Backwards Disturbance**	
d / m	0.56	0.75	0.89	0.75	0.89
Energy / J	4.0	5.5	7.3	5.5	7.3
FRASA / s	0.54 ± 0.02	**2.41** ± 0.04	**2.44** ± 0.03	0.62 ± 0.10	**2.26** ± 0.11
KFB / s	0.57 ± 0.02	5.96 ± 0.11	5.74 ± 0.05	0.49 ± 0.10	4.47 ± 0.23

The standard deviation is represented using the notation $\pm$

An estimate of the kinetic energy transferred at the moment of impact is calculated by considering the pendulum weight as a point mass. The weakest backwards disturbance does not cause any imbalance in the robot.

In all configurations, FRASA achieves comparable or shorter instability times than the KFB method. The only configuration where the KFB method has a lower mean instability time than FRASA is for a 5.5J disturbance from the back. However, the standard deviation ranges for the two methods overlap, indicating that the difference is not statistically significant.

For the most significant impacts ($d = 0.89$m representing 7.3J), FRASA is able to reject the disturbance in 42% of the time required by the KFB method for a front impact and in 51% of the time for a back impact. FRASA also demonstrates superior performance for 5.5J frontwards disturbance, where the return to stability takes only 40% of the time compared to the KFB method.

Overall, FRASA demonstrates its superiority both in rejecting significant disturbances and in recovering from a prone or a supine position, surpassing the KFB method.

VII. CONCLUSION

Experiments on the Sigmaban platform demonstrate FRASA's superiority over the KFB method in both standing up and fall recovery, achieving these tasks with increased efficiency. The integration of the CrossQ algorithm and the use of the robot's symmetry result in training times of around 30 minutes, which, to our knowledge, is unprecedented for successfully transferable humanoid stand up tasks.

The end-to-end nature of FRASA allows it to adapt to various disturbance intensities without requiring expert tuning. However, while leveraging the humanoid robot's symmetry accelerates training, it also limits the versatility of emerging behaviors, particularly in handling lateral disturbances.

To further enhance FRASA's performance, a promising direction would be to finalize its training directly on the robot (online) to potentially reduce the sim-to-real gap. Additionally, improving the modeling of actuators in our simulator could help bridge this gap even further.

REFERENCES

[1] S. Saeedvand, M. Jafari, *et al.*, "A comprehensive survey on humanoid robot development," *Knowl. Eng. Rev.*, vol. 34, p. e20, 2019.

[2] H. Kitano, M. Asada, *et al.*, "Robocup: The robot world cup initiative," in *International Conference on Autonomous Agents (AGENTS)* (W. L. Johnson, ed.), pp. 340–347, ACM, 1997.

[3] E. Krotkov, D. Hackett, *et al.*, "The DARPA robotics challenge finals: Results and perspectives," *J. Field Robotics*, vol. 34, no. 2, pp. 229–240, 2017.

[4] XPRIZE Foundation, "Avatar xprize: A global competition to create the next generation of telepresence technologies," 2021. Accessed: 2024-07-11.

[5] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 1–20, 2003.

[6] J. Bhatti, A. R. Plummer, P. Iravani, and B. Ding, "A survey of dynamic robot legged locomotion," in *International Conference on Fluid Power and Mechatronics (FPM)*, pp. 770–775, 2015.

[7] Z. Cai, Z. Yu, *et al.*, "Self-protect falling trajectories for humanoids with resilient trunk," *Mechatronics*, vol. 95, p. 103061, Nov. 2023.

[8] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, 2023.

[9] N. Scianca, D. D. Simone, *et al.*, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Trans. Robotics*, vol. 36, no. 4, pp. 1171–1188, 2020.

[10] M. Shafiee-Ashtiani, A. Yousefi-Koma, M. S. Panahi, and M. Khadiv, "Push recovery of a humanoid robot based on model predictive control and capture point," vol. abs/1612.08034, 2016.

[11] M. Missura, M. Bennewitz, and S. Behnke, "Capture steps: Robust walking for humanoid robots," *Int. J. Humanoid Robotics*, vol. 16, no. 6, pp. 1950032:1–1950032:28, 2019.

[12] J. E. Pratt, J. Carff, *et al.*, "Capture point: A step toward humanoid push recovery," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 200–207, IEEE, 2006.

[13] S. Kajita, F. Kanehiro, *et al.*, "The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, pp. 239–246, IEEE, 2001.

[14] E. Aslan, M. A. Arserim, and A. Uçar, "Development of push-recovery control system for humanoid robots using deep reinforcement learning," *Ain Shams Engineering Journal*, vol. 14, no. 10, p. 102167, 2023.

[15] H. Kim, D. Seo, and D. Kim, "Push recovery control for humanoid robot using reinforcement learning," in *International Conference on Robotic Computing (IRC)*, pp. 488–492, IEEE, 2019.

[16] B. van Marum, A. Shrestha, *et al.*, "Revisiting reward design and evaluation for robust humanoid standing and walking," *CoRR*, vol. abs/2404.19173, 2024.

[17] J. Stückler, J. Schwenk, and S. Behnke, "Getting back on two feet: Reliable standing-up routines for a humanoid robot," in *International Conference on Intelligent Autonomous Systems (IAS)* (T. Arai, R. Pfeifer, T. R. Balch, and H. Yokoi, eds.), pp. 676–685, IOS Press, 2006.

[18] S. Stelter, M. Bestmann, *et al.*, "Fast and reliable stand-up motions for humanoid robots using spline interpolation and parameter optimization," in *International Conference on Advanced Robotics (ICAR)*, pp. 253–260, IEEE, 2021.

[19] H. Jeong and D. D. Lee, "Efficient learning of stand-up motion for humanoid robots with bilateral symmetry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1544–1549, IEEE, 2016.

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

[21] "RoboCup KidSize League." `https://www.robocup.org/leagues/29`. Accessed: 2024-06-05.

[22] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[23] T. Haarnoja, A. Zhou, *et al.*, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," vol. 80, pp. 1856–1865, 2018.

[24] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (ICML)*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591, PMLR, 2018.

[25] A. Bhatt, D. Palenicek, *et al.*, "Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity," *CoRR*, vol. abs/1902.05605, 2019.

[26] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

[27] J. Tobin, R. Fong, *et al.*, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, IEEE, 2017.

[28] J. Allali, A. Boussicault, *et al.*, "Rhoban football club: Robocup humanoid kid-size 2023 champion team paper," in *RoboCup 2023: Robot World Cup XXVI* (C. Buche, A. Rossi, M. Simões, and U. Visser, eds.), vol. 14140 of *Lecture Notes in Computer Science*, pp. 325–336, Springer, 2023.

[29] G. Passault, Q. Rouxel, L. Hofer, S. N'Guyen, and O. Ly, "Low-cost force sensors for small size humanoid robot," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 1148–1148, 2015.

[30] "Onshape-to-robot." https://onshape-to-robot.readthedocs.io.

[31] R. Fabre, Q. Rouxel, G. Passault, S. N'Guyen, and O. Ly, "Dynaban, an open-source alternative firmware for dynamixel servo-motors," in *RoboCup 2016: Robot World Cup XX* (S. Behnke, R. Sheh, S. Sariel, and D. D. Lee, eds.), vol. 9776 of *Lecture Notes in Computer Science*, pp. 169–177, Springer, 2016.

[32] A. Raffin, A. Hill, *et al.*, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, pp. 268:1–268:8, 2021.

[33] "Openvino toolkit." https://github.com/openvinotoolkit/openvino.

[34] Z. Li, X. B. Peng, *et al.*, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *CoRR*, vol. abs/2401.16889, 2024.