A Multi-policy Approach Based on Clustering for Minimizing the Damage on a Falling Ballbot

Giulia Buzzetti[1], Michel Aractingi[2], Davide Zappetti[3] and Giovanni Iacca[1]

***Abstract*— Fall resilience remains a significant challenge for robots capable of locomotion. In the case of humanoid robots, arms can be utilized to mitigate impact damage, thereby improving resilience to falls. Although previous work has primarily focused on damage reduction strategies for legged robots, the question of minimizing fall damage in humanoid ballbots remains largely unexplored. In this paper, we introduce a novel multi-policy approach, combining clustering on the initial pose and Reinforcement Learning, to reduce damage resulting from a fall in a commercial humanoid ballbot. We conduct simulations to compare our method against three baselines: (1) a curriculum learning policy, (2) a single-policy approach enhanced by clustering information, and (3) a standard single-policy baseline. Experimental results demonstrate that our approach achieves higher success rates and greater damage reduction compared to existing alternatives.**

I. Introduction

Legged robots have garnered significant attention and extensive exploration in recent years due to their ability to perform diverse tasks and their remarkable versatility. However, the capability for unrestricted and agile movement is accompanied by the inherent risk of falls, which pose both safety hazards and substantial costs. Addressing these challenges, particularly by mitigating damage during falls and enhancing the resilience of such robots, remains a critical area of focus. The existing body of literature reflects a variety of studies dedicated to addressing this challenge by enhancing control.

Early studies on fall mitigation in humanoids introduced a "safe landing" approach, such as Fujiwara et al. [1] "Ukemi" algorithm, designed to reduce impact via learned or optimized maneuvers. Similarly, Ha and Liu [2] proposed an optimal contact sequence to minimize impact force, demonstrating meaningful reductions in simulation and hardware tests. Goswami et al. [3] focused on controlling a robot's fall direction to avoid harming bystanders, while Samy et al. [4] devised a dual strategy that firstly computes impact points while avoiding obstacles and then uses Quadratic Programming to minimize the impact on those selected points. Reinforcement Learning (RL) methods have also been explored. Kumar et al. [5] developed a control method using RL to identify the optimal contact points to minimize impact force and to control joint actuation for safer falls. Ma et al. [6] proposed a damage-reduction and recovery strategy from falls, while Wang et al. [7] used RL to reduce damage and facilitate recovery post-fall.

Despite these advances, most existing approaches focus on legged robots, leaving ballbots (i.e., ball-balancing robots that perform locomotion via a spherical wheel) as a relatively underexplored area of research. Ballbots rely on a single ground-contact point and maintain dynamic equilibrium via continuous rolling. Due to their configuration, they exhibit exceptional compliance and agile movement, making them particularly suitable for social environments [8]. However, their dynamic equilibrium also poses unique fall-related challenges. Additionally, ballbots lack the ability to bend their bodies, making strategies such as squatting motions inapplicable. To the best of our knowledge, no published method explicitly addresses minimizing damage during a ballbot fall, other than some preliminary results reported in [9].

In this paper, we propose a novel RL-based approach designed explicitly for ballbots to mitigate fall-related damage. Specifically, our main contributions are:

- A novel, multi-policy RL-based fall mitigation strategy targeted at ballbots experiencing different types of falls (i.e., from different initial poses of the arms).
- A clustering methodology for the initial pose of the arms, enabling specialized learned policies for the diverse fall scenarios experienced by the ballbot.
- A comparative study of the proposed multi-policy approach against single-policy and curriculum RL baselines, demonstrating the benefits of clustering.

The remainder of this paper is organized as follows. Section II introduces the background concepts, including RL, clustering, and multi-policy approaches. Section III details the experimental setup and the methods. Section IV details the tested RL approaches. Section V presents the results, and finally Section VI concludes the paper and discusses directions for future work.

II. Preliminaries

A. Reinforcement Learning

In RL with continuous state and action spaces (which is how we model the problem at hand in this paper), an environment is defined by a Markov decision process (MDP) [10], characterized by the tuple $(\mathscr{S}, \mathscr{A}, \mathscr{R}, \mathscr{T}, P_0)$. $\mathscr{S}$ and $\mathscr{A}$ denote the set of continuous states and continuous actions, respectively. Performing an action $a \in \mathscr{A}$ in a state $s \in \mathscr{S}$ results in a reward defined by the function $\mathscr{R} : \mathscr{S} \times \mathscr{A} \to R$.

[1]Giulia Buzzetti and Giovanni Iacca are with the Department of Information Engineering and Computer Science, University of Trento, 38123 Trento {giulia.buzzetti,giovanni.iacca}@unitn.it

[2]Michel Aractingi is with Hugging Face, Paris, France michel.aractingi@gmail.com

[2]Davide Zappetti is with Enchanted Tools, Paris, France davide@enchanted.tools

The dynamics of the environment is captured by the conditional transition probability distribution $\mathscr{T} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to R$, where $\mathscr{T}(s,a,s') = p(s_{t+1} = s' \mid s_t = s, a_t = a)$ specifies the probability of transitioning to the state s' at timestep $t+1$ given the current state s and the action a at timestep t. The initial state distribution is denoted by P_0.

Only the state space $\mathscr{S}$ and action space $\mathscr{A}$ of the MDP are assumed to be known to the learning agent. Through interactions with the environment, the agent collects samples from the reward function and state transition distributions.

We define a policy $\pi_\theta(s,a) = p_\theta(a_t = a \mid s_t = s)$, parameterized by θ, which gives the probability of selecting action a given the state. The objective is to optimize the parameters θ of the policy to maximize the expected discounted sum of rewards, defined as:

$$J(\theta) := E_{\pi_\theta, P_0, \mathscr{T}} \left[\sum_{t=0}^{H} \gamma^t \mathscr{R}(s_t, a_t) \right] \tag{1}$$

where H is the horizon of the episode (i.e., the number of timesteps) and $\gamma \in [0,1]$ is the discount factor.

B. Clustering

In our scenario, the initial pose of the robot's arms plays a crucial role, as different poses can lead to markedly different fall dynamics. To develop a robust and reliable control strategy, we apply a clustering approach to the initial poses, grouping those with similar characteristics.

Clustering is an unsupervised learning method that partitions data into distinct groups based on shared features. Because it is unsupervised, the algorithm does not require prior knowledge or human labeling.

In this work, we employ the K-means algorithm, an iterative clustering method [11]. K-means represents each cluster by its centroid, and data points are assigned to their nearest centroid. The centroid positions are then recalculated based on the assigned points. This process is repeated until convergence. The overall objective is to minimize the within-cluster sum of squares, defined as:

$$WCSS = \sum_{k=1}^{N} W(C_k),$$

where $W(C_k)$ is the within-cluster variation for the cluster C_k, defined as:

$$W(C_k) = \sum_{x_i \in C_k} ||x_i - \mu_k||^2$$

where x_i is the i-th point of the cluster and μ_k is the centroid of the cluster C_k. K-means was chosen as the clustering algorithm because it has been shown to yield good results in identifying patterns in robot trajectories [12].

The quality of the cluster can be evaluated with the silhouette score metric. The silhouette score is a metric assigned to each dataset point that measures how similar the point is to the centroid. It is calculated as follows:

$$\frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{2}$$

where a_i is the average distance to other points in the same cluster, and b_i is the average distance to points outside the cluster. The mean silhouette score across all points in the dataset provides an overall assessment of clustering quality. The score ranges from -1 to 1. A value close to -1 indicates that points may be assigned to the wrong cluster; whereas, values close to 1 signify well-separated clusters, and values near 0 suggest overlapping or ambiguous cluster boundaries.

C. Multi-policy approaches

Multi-policy approaches are often employed to develop control strategies, and they have demonstrated notable reliability, in particular in robotic tasks [13]. In various scenarios, they have also been combined with deep RL [14].

In this work, we refer to multi-policy control when the policy π is composed of different policies π_i from a finite set of policies $\Pi = \{\pi_1, \ldots, \pi_n\}$ [15]. Each policy depends on the state of the system and on a policy parameterization that defines its domain of applicability. In our case, each policy $\pi_i \in \Pi$ is trained separately on initial states sampled from its corresponding cluster C_i, where each cluster refers to a specific initial pose of the robot arms.

Of note, a similar approach to ours was proposed in [12], where authors segmented the trajectory of a robotic manipulator and trained a different policy for each segment. As in our approach, they used the K-means algorithm to divide the data. In contrast to their method, we have different policies for different types of falls. Once we identify the appropriate cluster using the initial pose of the arms, we employ the corresponding policy for the entire duration of the task.

III. Methods

A. Experimental setup

For our experiments, we used the humanoid ballbot developed by the company Enchanted Tools[1], shown in Figure 1. The robot is 115 cm tall, weighs approximately 30 kg, and is equipped with a head for human interaction and two arms with hands to grasp and carry items. It balances on a ball thanks to three omni-wheels that enable agile movements. Each arm has 7 Degrees of Freedom (DOF); in all experiments, we controlled only the 14 DOFs corresponding to the arms, while the other joints were treated as passive.

The control is velocity-based; the control's frequency is 12.5 Hz, while the simulation runs at 200 Hz. At each control step, the action network outputs the target speed for each of the 14 actuated joints. The speed is converted to the final position, and the positions are then divided into increments; at each simulation step, the position increment is applied to each joint.

We built our RL pipeline using Isaac Gym, which allows training on massively parallel vectorized environments [16]. The RL algorithm used to train the policies is PPO [17], which is a type of actor-critic algorithm [18]. We trained the policy on 4096 parallel environments to maximize the

[1] https://enchanted.tools

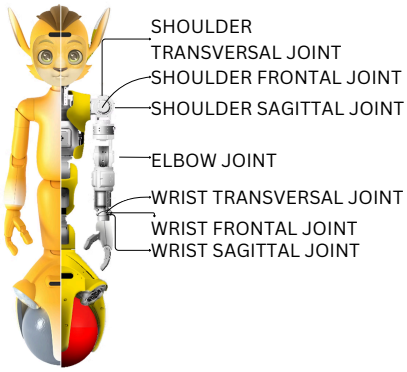

Fig. 1. Front view of Miroki, the robot character developed by Enchanted Tools, which was used for the experiments.

reward function designed in Section III-B. A symmetric actor-critic model is employed, sharing the same network and observations for both the policy and value functions. The network is a multi-layer perceptron with three ELU [19] hidden layers of sizes [512, 256, 128]. Table I details the hyperparameters of PPO used in the experimentation.

TABLE I
HYPERPARAMETERS USED FOR PPO.

Hyperparameters	**Value**
Max. episode length	5 s
Controller frequency	12.5 Hz
No. of environments	4096
Mini-batch size	16384
KL threshold	8e−3
Steps per update	24

The observation vector consists of the position and velocity of each actuated joint, the quaternion representing the root orientation, the linear and angular velocities of the root, and the previous action. The root orientation and velocities are aligned with those of the robot's base. Additionally, for the experiments that involve clustering, the observation vector also included the cluster id.

B. Reward function design

The overarching goal of this work is to develop a control strategy that minimizes damage to the robot during a fall. We seek a strategy that is effective regardless of the initial pose of the arms. An early attempt at applying RL for minimizing the damage of a falling ballbot was presented in [9], but that study considered only a single, fixed initial pose. In contrast, the goal of the present work is to develop a policy that reduces impact damage across a range of initial poses of the arms.

We chose to solve the task using RL to leverage the strengths of model-free solutions, given the difficulty of accurately modeling ballbot falls. The main part of the reward function is given by two different reward components: one is given at each step of the control loop, and the other is given only at the end of the episode. We refer to these two components as the *mid-episode reward* r_m and the *end-episode reward* r_e, respectively. Each episode ends when the robot is on the ground, or when it reaches a maximum duration of 5 s.

The *mid-episode reward* r_m is given by the Euclidean norm $||\cdot||$ of the actuated joint velocities $\dot{q}$:

$$r_m = ||\dot{q}||. \tag{3}$$

The objective of this component is to minimize unnecessary motion, hence reducing energy consumption.

The *end-episode reward* is provided only at the end of the episode and measures the total damage incurred by the robot. We evaluate the damage to the robot by considering both the contact forces on each body (due to ground impact) and the joint yank. The joint yank is the time derivative of the force applied to the joint; minimizing the yank thus helps preserve the joint's motor [20]. The *end-episode reward* r_e is in turn the weighted sum of two components r_{eb} and r_{ej}, the first related to the contact forces and the second related to the joint yank:

$$r_{eb} = \max_{t=0,\dots,H} \sum_{b\in\mathscr{B}} \left\| cf_b(t) \right\|^2 \tag{4}$$

$$r_{ej} = \max_{t=0,\dots,H} \sum_{i=1}^{N} \left\| df_i(t) - df_i(t-1) \right\|^2 \tag{5}$$

where again t is the timestep and H is the horizon of episode, and $\mathscr{B}$ is the set of robot bodies. The term $cf_b(t)$ represents the contact force acting on body b at time t, while $df_i(t)$ is the total force acting on the i-th actuated joint. Similarly to [6], [9], the weights ω_{eb} and ω_{eb} of these components are selected so that the contact forces have a stronger influence than joint yank. The weight for ω_m is chosen so that the mid-episode reward (on the order of 10^{-2}) is about 10^4 times smaller than the end-episode reward (on the order of 10^2). This choice reflects that only the end-episode reward refers directly to damage minimization.

The reward components' weights are given in II; note that all these weights are negative. Since these quantities measure the damage to the robot (which we want to minimize), having negative weights ensures that maximizing the total reward corresponds to minimizing damage.

TABLE II
WEIGHT COEFFICIENTS FOR EACH COMPONENT OF THE REWARD FUNCTION. THESE COEFFICIENTS, AS WELL AS THE REWARD FUNCTION ITSELF, ARE SHARED BY BOTH THE POLICY AND THE BENCHMARK POLICIES CONSIDERED IN THE EXPERIMENTATION.

Reward component	**Weight**
r_{eb}: contact force (end-episode)	$\omega_{eb} = -0.005$
r_{ej}: joint yank (end-episode)	$\omega_{ej} = -0.05$
r_m: velocity (mid-episode)	$\omega_m = -0.005$

C. Clustering on the initial pose of the arms

We analyzed how the robot falls when starting from different arm positions. To do so, at the start of each episode,

we reset the robot in simulation to an upright position with different initial arm positions, then allow it to fall without applying any control. For these different initial poses, we varied only the three shoulder joints and the elbow joint, as the position of the wrist is less relevant to the fall dynamics. Moreover, we considered only symmetric arm positions, reflecting typical real-life scenarios in which both arms are usually positioned symmetrically when moving or standing. Since the various initial poses produced a wide range of fall behaviors, we clustered them into smaller groups with similar characteristics.

We began by discretizing the joint position space. For each of the shoulder and elbow joints, we divided the range of motion into 5 equally spaced intervals, resulting in $5^4 = 625$ unique initial poses of the arms.

Next, for each of these 625 poses, we simulated 20 free-fall episodes (i.e., no policy was applied). For each pose, we then computed the following metrics across the 20 episodes:

- **Time of impact** (mean and median over 20 free-fall episodes): This is the duration from the start of the simulation until the moment of maximum contact force.
- **Body of impact** (mean and median over 20 free-fall episodes): This is the specific robot's body on which the maximum contact force occurs. Since the robot is composed of 64 bodies, this number can assume integer values from 0 to 63.
- **Maximum contact force** (mean and 95$^{\text{th}}$ percentile over 20 free-fall episodes): We recorded the maximum contact force in each of the 20 trials, then computed the mean and the 95$^{\text{th}}$ percentile of these values.

With these metrics in hand, we proceeded to cluster the data based on three features, namely 1) the median time of impact, 2) the median id of the body of impact, and 3) the 95$^{\text{th}}$ percentile of the maximum contact force. We varied the number of clusters from 2 to 68 (in increments of 2). The procedure involved normalizing all three features, applying the K-means algorithm, and evaluating different clustering configurations via the silhouette score. The silhouette score remained around 0.3 for all the clustering setups analyzed, suggesting an overall good separation with minimal overlap. Additionally, scores tended to be slightly lower when the number of clusters exceeded 56, indicating worse clustering quality.

To select the optimal cluster number, we examined three clustering configurations with high silhouette scores—-namely 52, 12, and 4 clusters—-representing small, medium, and large cluster sizes, respectively. For each clustering size, we selected the clusters associated with the maximum contact forces. Separate policies were trained for 300 episodes, considering only the positions within each cluster group.

For evaluation, each policy was tested over 20 episodes across all initial configurations within the corresponding group. In each episode, we compute the absolute value of the maximum contact force by taking the maximum force along the z-axis over all timesteps and all robot bodies as $Mcf = \max_{t\in\{0,\dots,H\},\mathscr{B}} |cf_{bz}(t)|$ where $cf_{bz}(t)$ is the z-component of the contact force acting on the body b at time t. For each cluster group, we record the mean $\overline{M}cf$ of these maximum contact forces. We compute the same metrics for a *free-fall* scenario, where no policy is applied. Furthermore, we define the *mean delta percentage* $\Delta\overline{M}$ of the contact forces relative to the free-fall case as:

$$\Delta\overline{M} = \frac{\overline{M}cf_{free} - \overline{M}cf_{policy}}{\overline{M}cf_{free}}. \tag{6}$$

We then took the average of $\Delta\overline{M}$ across all the initial poses to obtain, for each selected cluster group, the Total Mean Delta $T\Delta\overline{M}$, considered as an indicator of the overall performance of each RL approach.

We also defined a metric of success for the task. An initial pose is considered a *success* if the contact force achieved by the policy tested on that specific initial pose is lower than the contact force observed without the policy, and a *failure* otherwise. With this definition in hand, the Total Success rate TS is calculated by dividing the number of *successes* by the total number of initial poses within the cluster group. If we consider $\Delta\overline{M}$ only for the *successes*, we can calculate the mean $s\Delta\overline{M}$ and the standard deviation of the mean delta percentage for these *successes*. Similarly, we can calculate the mean $f\Delta\overline{M}$ and the standard deviation of the mean delta percentage for the *failures*. These metrics help us better understand the effectiveness of the *successes* and the severity of the *failures*.

TABLE III

METRICS USED TO EVALUATE THE POLICIES FOR EACH CLUSTER GROUP. EACH METRIC IS COMPUTED OVER 20 TEST EPISODES FOR EVERY INITIAL POSE WITHIN THE CORRESPONDING CLUSTER.

Cluster size	$T\Delta\overline{M}$ (%)	TS (%)	**Success** ($s\Delta\overline{M}$ $\pm$ Std)	**Failure** ($f\Delta\overline{M}$ $\pm$ Std)
Large	-2.69	0.54	26.02 $\pm$ 0.1675	-35.87 $\pm$ 0.2778
Medium	33.7	0.92	38.82 $\pm$ 0.1569	-26.0 $\pm$ 0.2245
Small	32.6	0.94	35.58 $\pm$ 0.1523	-15.02 -

As can be observed in Table III, the policies trained on the small and medium-sized cluster groups yield similar performance, whereas the policy trained on the large cluster group performs poorly. Consequently, we focused our analysis on the 12-group clustering configuration, which provides a balanced trade-off between cluster size and damage reduction.

IV. REINFORCEMENT LEARNING APPROACHES

We now introduce the multi-policy approach alongside the set of benchmark methods used for comparison. To assess the effectiveness of our method, we designed a set of baselines aimed at isolating the impact of both clustering and the use of multiple policies. Specifically, we include a single-policy baseline with and without clustering to evaluate their individual contributions.

Curriculum learning was also considered. Not only has it proven effective in handling complex robotic tasks, but it is specifically suitable for the problem we are addressing. In fact, the fall of a humanoid ballot exhibits high variability, but it is not composed of subtasks that require different

skills. This makes curriculum learning more suitable than hierarchical or multi-task reinforcement learning.

We excluded model-based reinforcement learning, as model-free methods are particularly suitable for handling uncertainties and situations that are difficult to model, like the one we are addressing.

A. Multi-policy approach

Considering the 12-group clustering configuration explained in Section III-C, we develop a multi-policy approach by training a separate policy for each of the 12 clusters, each one for 300 episodes, for a total of 3600 episodes. Since clustering is performed on a set of discrete joint positions, we define intervals around each joint's discrete value by calculating the midpoints between consecutive discrete values.

At the start of each episode, we randomly sample an initial pose of the arms from the set of poses used for clustering. We then add random noise to each joint while ensuring that the new positions remain within the intervals corresponding to the initially selected values. In total, the training process of this approach lasts approximately 3.67 days of wall-clock time.

B. Single-policy approach with clustering information

In this case, at the start of each episode, the initial pose of the arms is chosen randomly. We then identify the corresponding cluster for that initial pose as follows: for each joint, we consider the interval in which its value lies and the value in the interval that was used for the clustering. These discrete values form a vector used to query the appropriate cluster. This cluster identifier is subsequently incorporated into the observation vector. In total, 5200 episodes were run, lasting approximately 5.48 days of wall-clock time.

C. Single-policy approach without clustering information

In this case, at the start of each episode, in every environment, the initial pose is chosen by randomly assigning a position to the three joints of the shoulder (shoulder sagittal, frontal, and transversal) and to the elbow joint. In contrast to the previous approach, we did not determine the cluster to which the initial pose belongs. Instead, we trained a policy without this information to assess its relevance. In total, 5200 episodes were run, lasting about 5.4 days of wall-clock time.

D. Curriculum learning approach

Motivated by the success of curriculum learning methods in robotic tasks [21], we also trained a policy using a curriculum-based strategy to establish an additional baseline. To do that, we incrementally increased the difficulty of the fall task by altering the initial pose of the arms as follows:

- **First scenario**: Only the position of the first joint (shoulder sagittal) is randomized.
- **Second scenario**: The positions of the first two joints (shoulder sagittal and frontal) are randomized.
- **Third scenario**: The positions of the first three joints (shoulder sagittal, frontal, and transversal) are randomized.
- **Fourth scenario**: The positions of the first four joints (shoulder sagittal, frontal, transversal, and elbow) are randomized.

Randomizing a single joint tends to produce similar fall behaviors across episodes, whereas increasing the number of joints introduces greater variability and results in a higher-dimensional state-action space, thus making it a more difficult task. To empirically support this, we trained a policy for 500 episodes in two settings: one where only a single joint was randomized, and another where all joints were randomized. The single-joint setup achieved a Total Mean Delta $T\Delta\overline{M} = 45\%$, whereas the full-joint setup showed minimal improvement, with $T\Delta\overline{M} = 21\%$, highlighting the increased difficulty introduced by higher-dimensional control.

In all scenarios, the pose of the left arm mirrors the pose of the right arm. We transition from one scenario to the next, more challenging one either when the reward r_{eb} drops below a specified threshold ($r_eb \leq 2200$) or when the maximum number of episodes for the current scenario is reached. We set this maximum to 1200 episodes per scenario.

Overall, this curriculum learning approach was trained for 5200 episodes, lasting a total of 5.6 days of wall-clock time.

V. Results evaluation

To evaluate the results of the four RL approaches described in the previous section, we tested each policy on 20 test episodes for each of the 625 initial poses used in the clustering process.

We report the results in terms of $T\Delta\overline{M}$, TS, $s\Delta\overline{M}$ and $f\Delta\overline{M}$ for the different RL approaches in Table IV, where the metrics are defined in section III-C.

From the results, we can state that the curriculum learning approach seems not to be particularly suitable for this task, yielding the worst performance with a Total Success rate of only 39%. Furthermore, the single-policy approaches exhibit comparable performance; however, the policy incorporating clustering information outperforms the one without clustering, with $T = 65\%$ (vs. 61%) and a $T\Delta\overline{M} = 11\%$ (vs. 5%).

The best policy is provided by the multi-policy approach, which not only achieves $T = 72\%$ but also attains a $T\Delta\overline{M}$ exceeding 13%. Moreover, it maintains a lower failure mean delta percentage $f\Delta\overline{M}$, and although its success mean delta percentage $s\Delta\overline{M}$ is slightly lower than that achieved by the single-policy approaches, it is still comparable. Overall, this method stands out as the superior choice, particularly due to its faster training time (we purposely chose a much lower number of episodes than for the other methods, 3600 vs. 5200, as we observed much faster convergence with this method).

Demonstrative videos highlighting the improved performance of our approach compared to the baselines can be accessed at: `https://github.com/giu950/Multi-policy-approach-for-damage-reduction`.

VI. Conclusion

In this paper, we investigated the fall resilience of a humanoid ballbot. Our approach exclusively leverages the

TABLE IV

METRICS USED TO EVALUATE THE DIFFERENT RL APPROACHES. EACH METRIC IS COMPUTED ACROSS 20 TEST EPISODES FOR EACH OF THE 625 INITIAL POSES USED FOR CLUSTERING.

Method	$T\Delta\overline{M}$ (%)	TS (%)	**Success** ($s\Delta\overline{M}$ $\pm$ Std)	**Failure** ($f\Delta\overline{M}$ $\pm$ Std)
Multi-policy with clustering	+13.79	72.00	0.34 $\pm$ 0.1804	-0.37 $\pm$ 0.4124
Single-policy with cluster information	+11.26	65.44	0.39 $\pm$ 0.2193	-0.41 $\pm$ 0.3609
Single-policy w/o clustering information	+5.11	61.12	0.38 $\pm$ 0.2133	-0.47 $\pm$ 0.4972
Curriculum learning	-23.63	39.36	0.25 $\pm$ 0.1713	-0.55 $\pm$ 0.5375

arms to control the robot during a fall, since the wheels may lose contact with the ball, making direct ball control infeasible in such scenarios. By applying K-means, we clustered the possible initial poses of the arms and trained a separate policy for each cluster using RL. We validated our multi-policy approach against three baselines: (1) a single RL-based policy, (2) a single RL-based policy incorporating clustering information on the initial pose of the arms, and (3) an RL-based policy enhanced by curriculum learning. Overall, our proposed multi-policy approach achieved the best performance, achieving lower contact forces compared to the free fall, given an initial pose of the arms in $\sim$70% of the cases.

The trained policies could not be validated in a real-world setting. The company providing the robot is still in the startup stage, but once production becomes industrial and the robot's cost decreases, it will be possible to validate the policies in practice. Future work will also focus on extending the proposed method to non-symmetric poses of the arms as well as considering different body orientations.

REFERENCES

[1] K. Fujiwara, F. Kanehiro, S. Kajita, K. Kaneko, K. Yokoi, and H. Hirukawa, "UKEMI: Falling motion control to minimize damage to biped humanoid robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2521–2526.

[2] S. Ha and C. K. Liu, "Multiple contact planning for minimizing damage of humanoid falls," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 2761–2767.

[3] A. Goswami, S.-k. Yun, U. Nagarajan, S.-H. Lee, K. Yin, and S. Kalyanakrishnan, "Direction-changing fall control of humanoid robots: theory and experiments," *Autonomous Robots*, vol. 36, pp. 199–223, 2014.

[4] V. Samy, K. Bouyarmane, and A. Kheddar, "QP-based adaptive-gains compliance control in humanoid falls," in *IEEE International Conference on Robotics and Automation*. IEEE, 2017, pp. 4762–4767.

[5] V. C. V. Kumar, S. Ha, and C. K. Liu, "Learning a unified control policy for safe falling," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 3940–3947.

[6] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *IEEE International Conference on Robotics and Automation*. IEEE, 2023, pp. 12 149–12 155.

[7] Y. Wang, M. Xu, G. Shi, and D. Zhao, "Guardians as You Fall: Active Mode Transition for Safe Falling," *arXiv preprint arXiv:2310.04828*, 2023.

[8] C. Cai, J. Lu, and Z. Li, "Kinematic analysis and control algorithm for the ballbot," *IEEE Access*, vol. 7, pp. 38 314–38 321, 2019.

[9] G. Buzzetti, D. Zappetti, and G. Iacca, "A Reinforcement Learning Method to Minimize the Damage on a Falling Ballbot," in *European Robotics Forum*. Springer Nature, 2024, pp. 81–86.

[10] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[11] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[12] Y. Li, Y. Wang, G. Wang, and T. Shi, "A Multi-Policy Framework for Manipulator Trajectory Tracking Based on Feature Extraction and RL Compensation," in *International Conference on Automation, Robotics and Applications*. IEEE, 2024, pp. 191–195.

[13] M. Hossny and J. Iskander, "Biomechanic Posture Stabilisation via Iterative Training of Multi-policy Deep Reinforcement Learning Agents," *arXiv preprint arXiv:2008.12210*, 2020.

[14] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "Multi-agent behavioral control system using deep reinforcement learning," *Neurocomputing*, vol. 359, pp. 58–68, 2019.

[15] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multi-policy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, vol. 41, pp. 1367–1382, 2017.

[16] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning," 2021.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems*, 2000, pp. 1008–1014.

[19] D.-A. Clevert, "Fast and accurate deep network learning by exponential linear units (ELUs)," *arXiv preprint arXiv:1511.07289*, 2015.

[20] J. Ruiz-del Solar, R. Palma-Amestoy, R. Marchant, I. Parra-Tsunekawa, and P. Zegers, "Learning to fall: Designing low damage fall sequences for humanoid soccer robots," *Robotics and Autonomous Systems*, vol. 57, no. 8, pp. 796–807, 2009.

[21] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, "Curriculum learning: A survey," *International Journal of Computer Vision*, vol. 130, no. 6, pp. 1526–1565, 2022.