# Accessibility-Based Clustering for Efficient Learning of Locomotion Skills

# Accessibility-Based Clustering for Efficient Learning of Locomotion Skills

Chong Zhang[1*], Wanming Yu[2*], and Zhibin Li[2†]

***Abstract*— For model-free deep reinforcement learning of quadruped locomotion, the initialization of robot configurations is crucial for data efficiency and robustness. This work focuses on algorithmic improvements of data efficiency and robustness simultaneously through automatic discovery of initial states, which is achieved by our proposed K-Access algorithm based on accessibility metrics. Specifically, we formulated accessibility metrics to measure the difficulty of transitions between two arbitrary states, and proposed a novel K-Access algorithm for state-space clustering that automatically discovers the centroids of the static-pose clusters based on the accessibility metrics. By using the discovered centroidal static poses as the initial states, we can improve data efficiency by reducing redundant explorations, and enhance the robustness by more effective explorations from the centroids to sampled poses. Focusing on fall recovery as a very hard set of locomotion skills, we validated our method extensively using an 8-DoF quadrupedal robot *Bittle*. Compared to the baselines, the learning curve of our method converges much faster, requiring only 60% of training episodes. With our method, the robot can successfully recover to standing poses within 3 seconds in 99.4% of the test cases. Moreover, the method can generalize to other difficult skills successfully, such as backflipping.**

## I. Introduction

Among robot locomotion skills, trotting and some other tasks are easy in terms of exploration, while fall recovery is difficult due to many possible robot states. Thus, the ability to recover from a fall is critical for legged robots to improve their robustness against potential failures.

For fall recovery, manually-designed joint trajectories [1] are laborious, and their robustness in different environments is not guaranteed. Optimization-based [2] methods require a significant amount of time to obtain a feasible solution since dynamic models and complex contact situations need to be considered [3], which makes it difficult to achieve real-time fall recovery. To overcome the limitations of these methods, model-free deep reinforcement learning (DRL) methods [4] serve as a promising alternative.

However, learning fall recovery via DRL suffers from hard exploration, i.e., to explore in domains with sparse, delayed, or deceptive rewards, and redundant exploration, i.e., certain regions of the state space being too frequently visited while training because of skewed data collection. Both hard exploration and redundant exploration can detriment the data efficiency and learning performance, and initial state distribution is one of the key factors that can affect the efficiency of exploration.

Currently, there are three common ways to design initial state distributions for learning fall recovery via DRL: 1) initialization from demonstrations [5], 2) initialization from random distributions [6], and 3) initialization from predefined poses [4]. For initialization from demonstrations, the performance is limited to the demonstrated examples. For initialization from random distributions, it is not data efficient [6] because of the redundant exploration. Also, corner cases may suffer from insufficient exploration because of skewed data collection. As a result, the diversity of exploration is not guaranteed. For initialization from predefined poses, states that lack heuristics are very likely to be missed, and the generalization can be a problem despite high data efficiency.

In this work, we aim to automatically discover initial states that can help achieve high robustness while still being data efficient. We propose to achieve this by clustering the static poses of the robot and applying centroids as initial states.

### A. Related Work

Regarding DRL, model-free DRL has been a commonly-used method for fall recovery and other locomotion tasks [7] [8] [9]. In contrast to model-based methods such as model predictive control [10] [11], trajectory optimization [12] [13] and Bayesian optimization [14], model-free methods do not require explicit knowledge of complex dynamics, and support fast online computation without mathematical optimization. The most popular model-free DRL algorithms for robot locomotion are the PPO algorithm [15] and the SAC algorithm [16].

Regarding clustering, common clustering methods have successful applications in RL, such as value function approximation [17], but they fail to achieve ideal clustering of diverse robot states. For centroid-based methods such as K-Means [18] and density-based methods such as DBSCAN [19], the problem is the metric for clustering, which will be discussed later. For pattern-based methods [20], state transitions of the robot can be regarded as edges in a directed graph. However, the existing methods could not achieve the desired effect as well, which will be discussed in I-B.

Regarding the metric, we usually adopt the Euclidean distance assuming orthogonal coordinates and undirected distances. However, both of the assumptions are problematic for robot poses. For the state space of static poses, we tend to use the combination of the normalized gravity vector and the

*These authors contributed equally to this work.
†Corresponding author. Email: zhibin.li@ed.ac.uk
[1]Chong Zhang is with Department of Precision Instrument, Tsinghua University, Beijing, 100084 China.
Email: chong-zh18@mails.tsinghua.edu.cn
[2]Wanming Yu and Zhibin Li are with the School of Informatics, University of Edinburgh, 10 Crichton St, Edinburgh EH8 9AB, United Kingdom.
Email: wanming.yu@ed.ac.uk, zhibin.li@ed.ac.uk
Accompanying video: https://youtu.be/7cZMThUn0rU

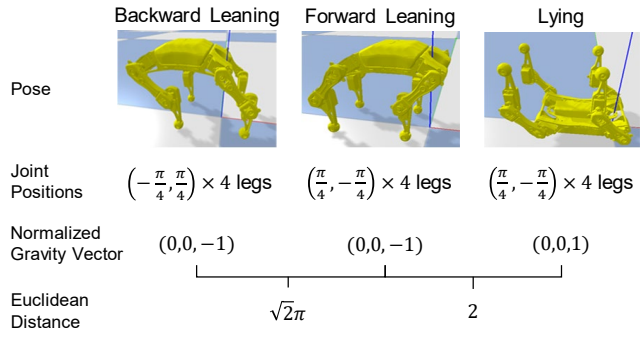


Fig. 1. An erroneous case of using Euclidean distance metric. The distance between a backward leaning stance and a forward leaning stance should be smaller than that between a forward leaning stance and a lying pose.

joint positions [4], and these feature dimensions are coupled. Also, it is difficult to determine the scale of the features. A failure case of the Euclidean distance is shown in Fig. 1. The undirected distances are inapplicable because the robot's transitions are directed. For instance, it is easy to fall from a standing pose, but difficult to recover from a lying pose.

There are also some metric learning methods, but they are time-consuming and difficult to obtain values for each start-end pose pair. In [21], the distance metric is approximated for state-space RRTs [22], but it is impractical to construct an RRT for each pose. In [23], the metric is learned during the training process, but we do not want to learn the metric for a certain DRL model before clustering.

### B. Motivation and Our Contribution

In this paper, we aim to find the initial states that can help reduce hard exploration and redundant exploration during the training process to learn robust fall recovery efficiently. To ensure robustness, the initial states need to cover as much of the state space as possible. We expect the centroids of the static-pose clusters to serve as the ideal initial states.

In terms of the metric for clustering, we propose the accessibility metric in II-A to overcome the shortcomings of the Euclidean distance mentioned in I-A.

For pattern-based clustering, we tried several common methods but they all failed. The directed Louvain method [24] failed because it cannot properly distinguish the direction of the edges (state transitions in this paper) [25]. The Infomap method [26] failed because unreachable states could exist in the same cluster, which means that the agent was expected to explore states that can hardly be reached from the initial states. The spectral clustering method [27] failed because the number of clusters was too small, which indicates low robustness. To obtain the ideal clustering effect, we propose a new accessibility-based clustering method, i.e., K-Access, in II-C.

Our contributions in this paper include:

- An accessibility metric to quantify the level of difficulties for a robot to transition from one physical state/configuration to another;
- A K-Access algorithm based on the accessibility metric for state-space clustering, which is adapted from the K-Means++ algorithm;

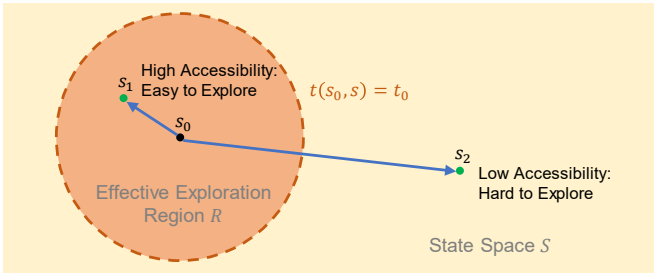


Fig. 2. An illustration of accessibility. The accessibility value corresponds to the difficulty of a state being explored from the initial state.

- A pipeline of automatically discovering feasible initial states, based on their inter-connected transitions, for efficient learning of robot fall recovery and other tasks.

With our proposed method, the data efficiency of DRL models can be greatly improved by avoiding repeated and redundant explorations, and the robustness can be greatly enhanced because of the wide range of searched states and their explored inter-connections.

## II. Methodology

In this section, the concept of accessibility is firstly introduced in II-A. The criteria of good initial state distributions are discussed in II-B, and the K-Access algorithm is presented in II-C. In II-D, the DRL is applied for fall recovery, and the entire pipeline is shown in Fig. 4. Generic principles to apply our proposed method to learning other locomotion tasks are presented in II-E.

### A. Accessibility

To model the difficulty of transitions from one state to another, we propose the accessibility metric. Figure 2 demonstrates the concept of accessibility. Consider an initial state $s_0$ in the state space $S$. There is a region $R \subseteq S$ that can be easily and effectively explored from $s_0$, and this region can be mathematically defined as

$$R = \{s | s \in S, t(s_0, s) < t_0\}, \tag{1}$$

where $t(s_0, s)$ is the minimal time cost (in seconds) of the transition from $s_0$ to $s$, and $t_0$ is a positive value. $R$ is called the effective exploration region of $s_0$.

Consider another two states $s_1 \in R$ and $s_2 \notin R$. We can say that the accessibility from $s_0$ to $s_1$ is high, and it is easy to explore $s_1$ from $s_0$. In contrast, the accessibility from $s_0$ to $s_2$ is low, and it is difficult to explore $s_2$ from $s_0$.

Mathematically, we define the accessibility from $s_i \in S$ to $s_j \in S$ as

$$\text{access}(s_i, s_j) = e^{-t(s_i, s_j)}. \tag{2}$$

In practice, it is difficult to get the minimal time cost $t(s_i, s_j)$, and we approximate it with PD tracking (see III-B). The range of accessibility is $[0, 1]$. If the accessibility is zero, $s_j$ is unreachable from $s_i$. If the accessibility is one, $s_i = s_j$.

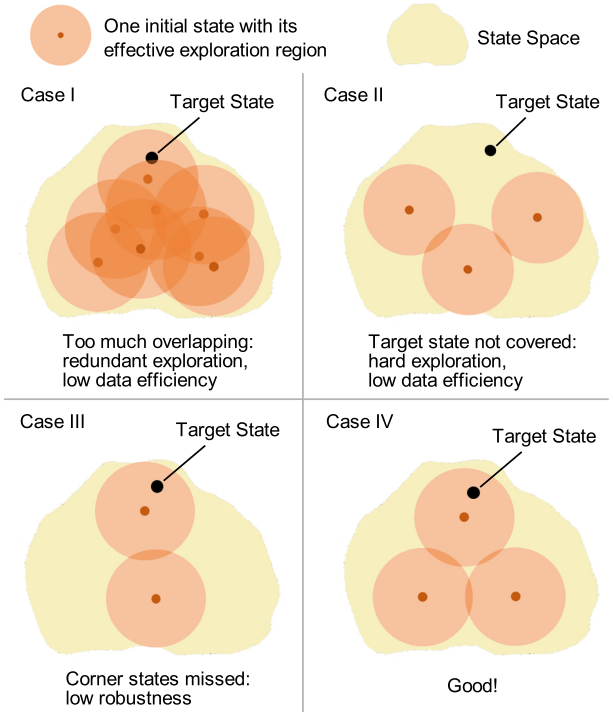


Fig. 3. Different cases of initial state distributions. The target state is what the agent is expected to achieve during exploration (see II-E). Randomized initialization corresponds to case I, and we expect the auto-discovered initial states are similar to case IV.

### B. What Makes Good Initial States

With the concept of accessibility, we can evaluate whether a distribution of initial states is good for the DRL exploration. Figure 3 shows different cases of initial state distributions. For data efficiency, redundant exploration and hard exploration are detrimental. For robustness, the effective exploration regions should cover as much state space as possible. For the trade-off between data efficiency and robustness, we need to select the initial states and their effective exploration regions with sufficient coverage but minimal redundancy.

To ensure enough coverage, we propose to randomly sample a wide range of static poses. Then we cluster the sampled poses to reduce the redundancy. We expect that the centroids of the obtained clusters can make good initial states. Their effective exploration regions should cover most of the state space, and there should be little overlapping.

Based on the discussion above, we can evaluate whether the obtained clusters are good. If the inter-cluster accessibility is too high, there is much overlapping since the centroids can be too close to each other, leading to redundant exploration. If the intra-cluster accessibility is too low, the coverage is low since many samples are not in the effective exploration region of their centroids. Hence, for good clustering results, the inter-cluster accessibility should be low, and the intra-cluster accessibility should be high.

### C. K-Access Algorithm

Based on the accessibility metric proposed in II-A, we refer to the K-Means++ algorithm [28] and propose the K-Access clustering method. The algorithm is presented in

**Algorithm 1** K-Access$(k, A)$

1: $cIndex \leftarrow \text{zeros}(k)$; ◁ indices of centroids
2: $cIndex[0] \leftarrow \text{randInt}(0, k)$;
3: **for** $i = 1$ to $k-1$ **do**
4: $\quad CAccess \leftarrow \sum_{j=0}^{i-1} (A[cIndex[j],:] + A[:, cIndex[j]])$;
5: $\quad cIndex[i] \leftarrow \arg\min_j CAccess[j]$; ◁ initialize $cIndex$
6: **end for**
7: $assignment[l] \leftarrow \arg\max_{c \in cIndex} A[c,l], \forall l < n, l \in \mathbb{N}$;
8: $preassign \leftarrow \text{zeros}(n)$; ◁ previous $assignment$
9: **while** $preassign \neq assignment$ **do**
10: $\quad preassign \leftarrow assignment$;
11: $\quad cIndex[i] \leftarrow \arg\max_{m \text{ s.t. } a_m = c_i} \min_{a_l = c_i} A[m,l], \forall i < k, i \in \mathbb{N}$;
12: $\quad assignment[l] \leftarrow \arg\max_{c \in cIndex} A[c,l], \forall l < n, l \in \mathbb{N}$;
13: **end while**
14: **return** $cIndex, assignment$

Algorithm 1. The inputs are the number of clusters $k$ and the accessibility matrix $A$ for the sampled states $s_0, s_1, \ldots, s_{n-1} \in S$. The size of $A$ is $(n,n)$, and $A[i,j]$ is the accessibility from $s_i$ to $s_j$ $(i, j \in \{0, \ldots, n-1\})$. The outputs are the indices of the centroids $cIndex = (c_0, \ldots, c_{k-1})$ and the centroids of each state's cluster $assignment = (a_0, \ldots, a_{n-1})$.

Similar to the K-Means++ algorithm, the first step of K-Access algorithm is to initialize the centroids, and then we repeat the assignment step and the update step until the assignments no longer change, as described below.

1) *Initialization of centroids:* The first centroid is randomly selected (line 2), and each new centroid is the furthest sample from the already selected centroids (line 3-6).
2) *Assignment of samples to clusters:* Each sample state is assigned to the cluster of which the centroid has the highest accessibility to this state (line 7,12). Note that we only consider the accessibility of single direction here.
3) *Update of the choice of centroids:* The new centroid has the maximal neighborhood accessibility. The neighborhood accessibility of one state is defined as the minimal accessibility value from the state to its neighbors in the same cluster (line 11). To ensure robustness, we do not take the average, which differs from K-Means++.

Finally, the K-Access algorithm can converge to a result that prefers high intra-cluster accessibility and low inter-cluster accessibility. To determine the number of clusters, we propose an index based on the discussions in II-B that good clustering results should have high intra-cluster accessibility and low inter-cluster accessibility. The index $I$ is defined as

$$I = \overline{\log(A_{\text{intra}})} - \overline{\log(A_{\text{inter}})} - \alpha \cdot |\Lambda|, \tag{3}$$

where $A_{intra}$ is the intra-cluster accessibility array of size $k$, $A_{inter}$ is the inter-cluster accessibility of size $(k,k)$, and $\alpha \cdot |\Lambda|$ is the regularization term. The clustering results are better if the index is larger, and the inter/intra-cluster metrics are also defined in a different way from K-Means++. To be specific,

$$A_{\text{intra}}[i] = \min_{a_l = c_i, l < n} A[c_i, l], \forall i < k, \tag{4}$$

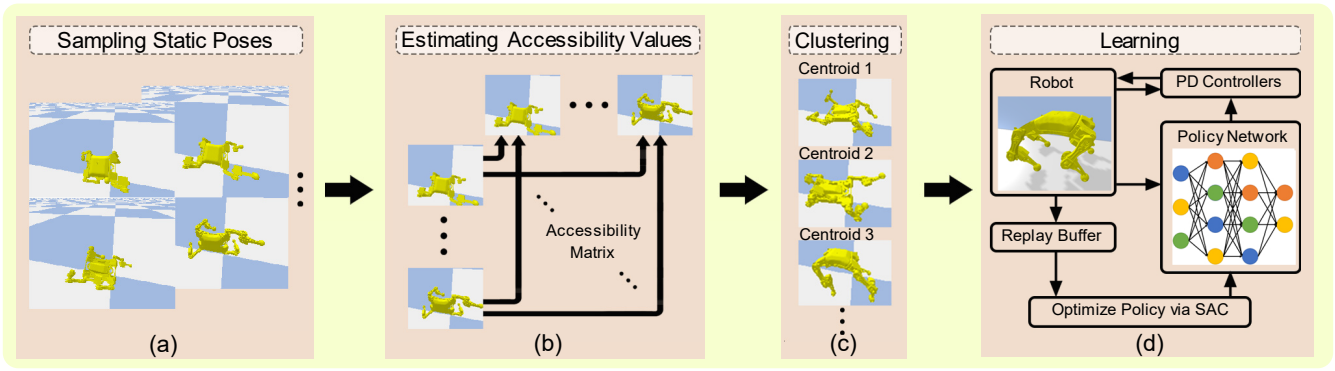


Fig. 4. Pipeline of the proposed method. First, static poses are randomly sampled. Second, the estimated accessibility values are obtained via simulation. Third, the proposed K-Access algorithm selects and discovers the optimal initial states. Finally, the DRL agent learns fall recovery through exploration based on the discovered initial states.

$$A_{\text{inter}}[i,j] = \begin{cases} \underset{a_l=c_i}{\text{mean}}\, A[a_l,c_j], \forall i \neq j, \\ 1, \forall i = j, \end{cases} \tag{5}$$

where $i<k, i\in\mathbb{N}, j<k, j\in\mathbb{N}$,

$$\Lambda = \{c_i \mid i<k, i\in\mathbb{N}, |\{l \mid a_l = c_i, l<n, l\in\mathbb{N}\}| = 1\}, \tag{6}$$

and $\alpha$ is a real value (recommended value: 1). We penalize the number of one-sample clusters because such clusters can hardly be encountered and they may exist because of extreme circumstances. In most cases, one-sample clusters are also accessible from other clusters. Therefore, one-sample clusters should be rare or non-existent, otherwise the training process may suffer from unnecessary or redundant explorations. In Fig. 4 (c), the K-Access algorithm is applied for clustering, after the estimation of accessibility values for the sampled poses.

### D. Fall Recovery Learning

Our DRL framework is shown in Fig. 5. The state space consists of the orientation (represented by the normalized gravity vector), the angular velocity of the body, and the joint positions. Here we adopt the positional control for high learning efficiency and performance according to [29]. The outputs of the policy network are the target joint positions which update at 25 Hz. The PD controllers generate torques based on the target joint positions and the measured joint positions at 300 Hz. The SAC algorithm is applied for learning. In our implementation, we use the 8-DoF quadrupedal robot *Bittle* [30] and the PyBullet [31] simulation environment.

The *Bittle* is equipped with an IMU on its body. The angular velocity can be directly accessed from the IMU. The orientation is represented as the gravity vector in the body frame which is normalized to be of length 1. The gravity vector can be computed using the IMU measurements.

The reward function is the sum of reward terms in Table I. The *jump* regularization term aims to penalize the robot for actions (e.g., spinning) to adjust the orientation in the air, and the action difference term serves to reduce unrealistic large movements. We assign larger weights to the reward terms related to body height and orientation since we characterize a standing pose mainly by the body height and the orientation

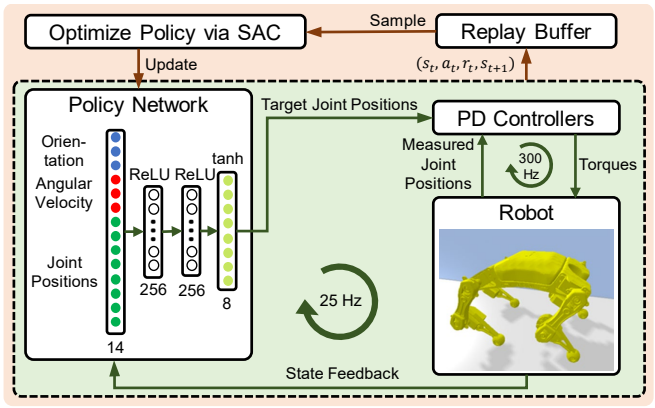


Fig. 5. DRL framework overview. The policy network generates the target joint positions at 25 Hz. The actuators follow the 300 Hz impedance control based on the target and the measured joint positions. The SAC algorithm is applied for learning the feedback control policy.

for the fall recovery task. The radial basis function (RBF) applied in these terms is defined as:

$$\text{RBF}(x,y,\alpha) = \exp\left(\alpha \cdot \|x-y\|_2^2\right). \tag{7}$$

The learning process is the last stage (d) of the pipeline in Fig. 4, where we apply the centroids in (c) as the initial states for DRL of quadruped fall recovery.

### E. Learning Other Tasks

The proposed method is not limited to fall recovery learning. In III-E, we also validated our method in back-flip learning. The generic principles to apply the proposed method are:

1) Perform clustering in a subspace with feasible and necessary dimensions;
2) Ensure that the target state has the maximum expected return in the subspace;
3) Estimate the accessibility values with time-accuracy trade-off.

The first principle is about how we define the subspace for clustering. Some dimensions such as velocities are not suitable for estimating the accessibility values. Also, necessary dimensions to identify the initial states must be included.

TABLE I

REWARD TERMS FOR DRL.

| Symbols | |
|---|---|
| $h$ | body height |
| $g_{\text{ori}}$ | normalized gravity vector |
| $\Omega$ | body angular velocity |
| $\tau$ | vector of joint torques |
| $\omega$ | vector of joint velocity |
| $c_{\text{b}}$ | 0 if the body touches the ground, otherwise 1 |
| $c_{\text{f}}$ | 0.3×the number of feet that touch the ground |
| $h_{\text{f}}$ | vector of the distances from four feet to the ground |
| $d_{\text{s}}$ | the shortest distance from the robot to the ground |
| $p$ | vector of the measured joint positions |
| $p^{\text{t}}$ | vector of the target joint positions |
| $\hat{\cdot}$ | target value |
| **Reward Terms** | |
| Height | $0.667 \times \text{RBF}(h, \hat{h}, -2000)$ |
| Orientation | $0.333 \times \text{RBF}(g_{\text{ori}}, [0,0,-1]^T, -5)$ |
| Angular velocity | $0.067 \times \text{RBF}(\Omega, 0, -0.05)$ |
| Joint torques | $0.067 \times \text{RBF}(\tau, 0, -5)$ |
| Joint velocity | $0.067 \times \text{RBF}(\omega, 0, -0.05)$ |
| Contact | $0.067 \times c_{\text{b}} + 0.033 \times c_{\text{f}}$ |
| Foot lift | $0.067 \times \text{RBF}(h_{\text{f}}, 0, -100)$ |
| Jump regularization | $0.033 \times \text{RBF}(d_{\text{s}}, 0, -100)$ |
| Action difference | $0.033 \times \text{RBF}(p^{\text{t}}, p, -1)$ |

The second principle is about the reward function and the target state. In fall recovery, the target state corresponds to the standing pose. In other tasks such as backflipping, the target state can refer to a good starting pose that can get the maximum episode reward in the future.

The third principle is about the estimation of accessibility values. In fall recovery, we apply the response time of PD control. However, there can be errors in some cases, e.g., when large torques make the robot fly. Also, more complicated and high-level controllers can be necessary for other scenarios. In such cases, a trade-off between complexity and estimation accuracy needs to be considered.

## III. IMPLEMENTATION AND RESULTS

### A. Sampling Static Poses

In the PyBullet environment, we randomly initialized the *Bittle* robot with roll angle $\phi \sim \text{U}(-\pi, \pi)$, pitch angle $\theta \sim \text{U}(-\frac{\pi}{2}, \frac{\pi}{2})$, yaw angle $\psi = 0$, and joint positions $p \sim \text{U}(-\frac{5}{6}\pi, \frac{5}{6}\pi)$. Self-collision cases were abandoned.

The robot was dropped from 0.35 m above the ground. If the robot was stationary within 2 seconds (with negligible velocity, angular velocity, and distance to the ground), we recorded the final joint positions, the final roll angle, the final pitch angle, and the body height. We assume the ground to be flat for sampling.

With 12× multiprocessing, we sampled 2.4k static poses within an hour on an ordinary desktop machine.

### B. Estimating Accessibility Values

To estimate the accessibility from static pose A to static pose B, we initialized the robot with static pose A. Then we sent a command to the PD controllers with the joint positions of B as the target positions. If the robot entered static equilibrium within 3 seconds, we checked whether the state of robot was close to the state of B. If these two states were close enough, we recorded the time cost $t$ and the estimated accessibility from A to B was $e^{-t}$. Otherwise, we set the accessibility to be $10^{-8}$ instead of zero because all the sampled static poses are assumed to be accessible.

We randomly selected 1k samples from the sampled static poses. With 12× multiprocessing, we obtained the 1M accessibility values for the 1k×1k accessibility matrix within 20 hours on an ordinary desktop machine.

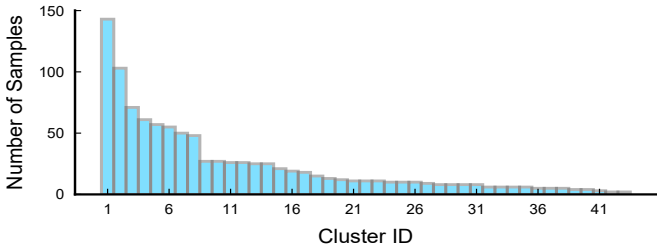


Fig. 6. Number of samples within each cluster when there are $k = 43$ clusters. The clusters are sorted by the number of samples assigned to them.

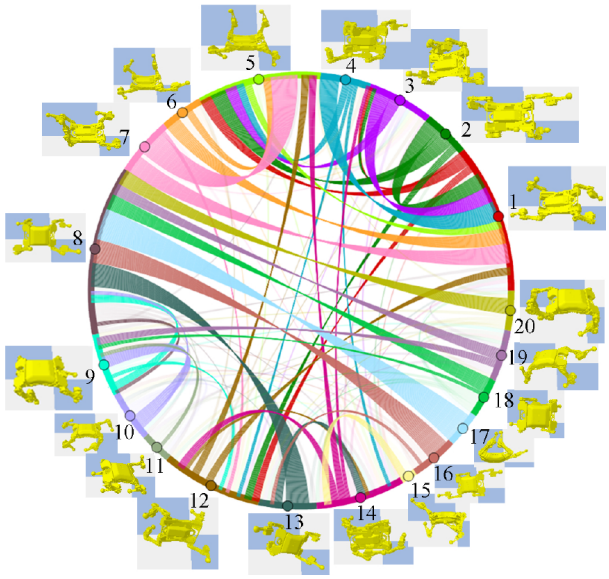


Fig. 7. Visualization of the inter-cluster accessibility of top-20 clusters. Inter-cluster accessibility values above 0.15 are highlighted, and those values below 0.05 are omitted here for clarity.

### C. Clustering

We applied the proposed K-Access algorithm to the 1k×1k accessibility matrix obtained in III-B. The $\alpha$ value for the index is 1. To determine the number of clusters $k$, we tried different $k$ values and obtained the maximum index value when $k = 43$. The number of samples in each of the 43 clusters is shown in Fig. 6, indicating that the static poses of the robot are subjected to a long-tail distribution.

Figure 7 visualizes the inter-cluster accessibility of top-20 clusters via chord diagram, which can also contribute to pose taxonomy analysis [32] [33]. Different clusters correspond to different contact cases, and the inter-cluster accessibility corresponds to the difficulty of transitions.

### D. Deep Reinforcement Learning

We applied six kinds of initial state distributions:

1) Centroids of the obtained 43 clusters by K-Access (abbreviated to KA);

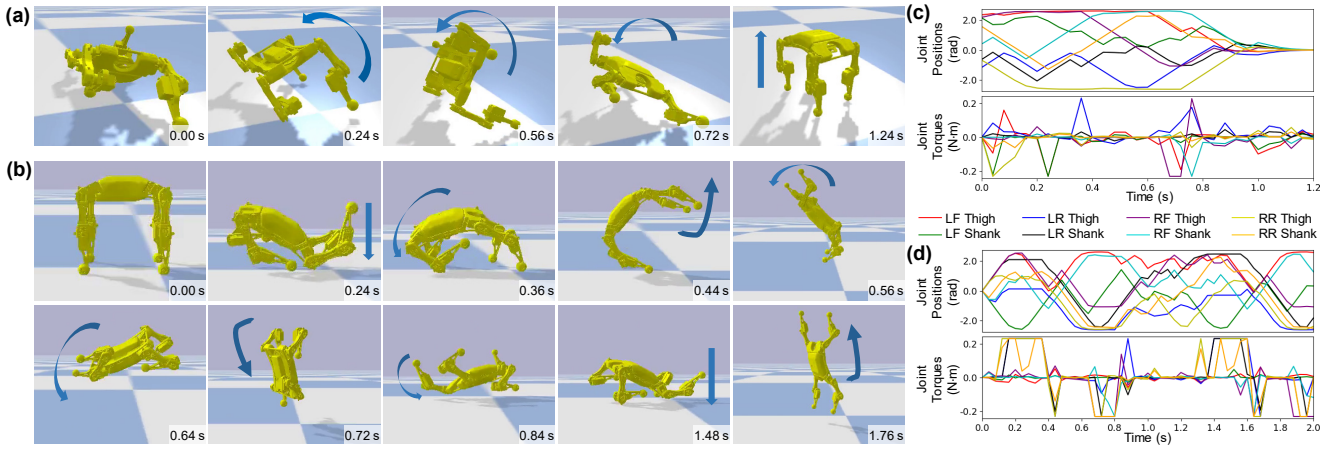


Fig. 8. Snapshots of (a) fall recovery and (b) continuous backflips. (c) shows the joint positions and torques for (a), and (d) shows those for (b).

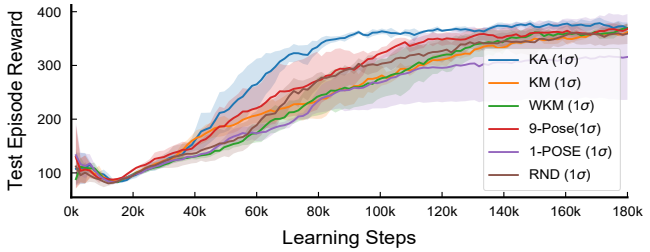


Fig. 9. Learning curves of fall recovery for different initial state distributions. The proposed method (KA) shows the highest data efficiency. There are 300 steps per episode, the discount factor is 0.987. Averaged over 3 random seeds, with ±1 SD. in shadow.

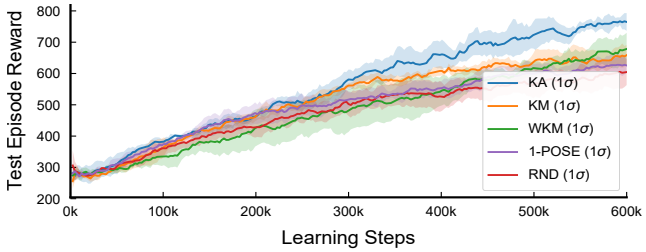


Fig. 10. Learning curves of backflipping for different initial state distributions. Averaged over 3 random seeds, with ±1 SD. in shadow.

2) Centroids of the obtained 33 clusters by K-Means++ based on the generalized Dunn's index $v43$ [34] (abbreviated to KM);
3) Centroids of the obtained 14 clusters by weighted K-Means++ based on $v43$ (abbreviated to WKM, gravity vector weighted by 2);
4) Nine initial poses applied in [4] (abbreviated to 9-Pose);
5) One lying pose (abbreviated to 1-Pose);
6) Random static poses (abbreviated to RND).

We demonstrate the proposed method can greatly improve the data efficiency, based on the learning curves for fall recovery using the same test samples shown in Fig. 9.

For robustness, agents of the best seeds were tested on another 500 static poses. There are 75 steps (3 seconds) per episode during the test. The performance scores are presented in Table II. It can be seen that the proposed method can be used to learn robust fall recovery policies with 25% fewer steps than other distributions. Test runs are shown in the accompanying video. Snapshots are in Fig. 8(a).

### E. Backflip Learning

We applied our proposed method to learn backflipping and the learning curves using the same test samples are in Fig. 10. Here we only clustered the static poses with roll angle less than 60 deg. See snapshots in Fig. 8(b). Test runs and learning details are in the accompanying video.

## IV. Conclusion

In this paper, we propose to automatically discover initial states for DRL of locomotion skills via the accessibility metric and the K-Access clustering algorithm. With the obtained centroids as the initial states, the data efficiency can be greatly improved, and the robustness can be guaranteed. Despite the extra computation to estimate accessibility values before clustering, the centroids are only computed once based on this learning-free metric, and they can boost the training process regardless of the choices of DRL algorithms, hyperparameters, and reward functions.

The K-Access algorithm performs better than the existing clustering methods because: 1) the direction of transitions is taken into consideration; 2) the neighbors in one cluster are all easy to explore from the centroid; 3) the number of clusters can be assigned and determined by the index value. The generalization to backflipping also shows that intermediate and unstable poses can also be better explored with our method.

Future work will focus on the possible extensions to other tasks that can benefit from the proposed technique. We will also do hardware validation and try other methods for better accessibility estimation, e.g., neural networks.

TABLE II

Performance on the Test Poses

| Initial States | Training Episodes | Episode Reward | | Success Rate in 3 s (%) |
|---|---|---|---|---|
| | | Mean | SD. | |
| KA (proposed) | **1200** | **81.93** | **11.84** | **99.4** |
| KM | 1600 | 74.84 | 20.14 | 91.8 |
| WKM | 1600 | 79.04 | 18.20 | 94.4 |
| 9-Pose | 1600 | 73.35 | 17.01 | 91.6 |
| 1-Pose | 1600 | 73.09 | 17.75 | 92.6 |
| RND | 1600 | 79.97 | 17.32 | 94.6 |

## REFERENCES


[1] S. Shamsuddin *et al.*, "Humanoid robot nao: Review of control and motion exploration," in *2011 IEEE international conference on Control System, Computing and Engineering*, 2011, pp. 511–516.

[2] I. Mordatch *et al.*, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[3] O. Melon *et al.*, "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[4] C. Yang *et al.*, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, 2020.

[5] X. B. Peng *et al.*, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[6] D. Reda *et al.*, "Learning to locomote: Understanding how environment design matters for deep reinforcement learning," in *Motion, Interaction and Games*, 2020, pp. 1–10.

[7] J. Yue, "Learning locomotion for legged robots based on reinforcement learning: A survey," in *2020 International Conference on Electrical Engineering and Control Technologies (CEECT)*, 2020, pp. 1–7.

[8] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019.

[9] N. Rudin *et al.*, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021.

[10] M. Neunert *et al.*, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[11] J. Lafaye *et al.*, "Model predictive control for tilt recovery of an omnidirectional wheeled humanoid robot," in *2015 IEEE international conference on robotics and automation (ICRA)*, 2015, pp. 5134–5139.

[12] I. Chatzinikolaidis *et al.*, "Contact-implicit trajectory optimization using an analytically solvable contact model for locomotion on variable ground," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6357–6364, 2020.

[13] J. Wang *et al.*, "Whole-body trajectory optimization for humanoid falling," in *2012 American Control Conference (ACC)*, 2012, pp. 4837–4842.

[14] K. Yuan *et al.*, "Bayesian optimization for whole-body control of high-degree-of-freedom robots through reduction of dimensionality," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2268–2275, 2019.

[15] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[17] X. Xu *et al.*, "A clustering-based graph laplacian framework for value function approximation in reinforcement learning," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2613–2625, 2014.

[18] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[19] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[20] F. D. Malliaros and M. Vazirgiannis, "Clustering and community detection in directed networks: A survey," *Physics reports*, vol. 533, no. 4, pp. 95–142, 2013.

[21] M. Bharatheesha *et al.*, "Distance metric approximation for state-space rrts using supervised learning," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 252–257.

[22] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University, Tech. Rep., 1998.

[23] M. E. Taylor *et al.*, "Metric learning for reinforcement learning agents," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2011, pp. 777–784.

[24] N. Dugué and A. Perez, "Directed louvain: maximizing modularity in directed networks," Ph.D. dissertation, Université d'Orléans, 2015.

[25] Y. Kim *et al.*, "Finding communities in directed networks," *Physical Review E*, vol. 81, no. 1, p. 016103, 2010.

[26] L. Bohlin *et al.*, "Community detection and visualization of networks with the map equation framework," in *Measuring scholarly impact*. Springer, 2014, pp. 3–34.

[27] D. Gleich, "Hierarchical directed spectral graph partitioning," *Information Networks*, vol. 443, 2006.

[28] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[29] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?" in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2017, pp. 1–13.

[30] Petoi. (2021) Bittle robot. [Online]. Available: https://www.kickstarter.com/projects/petoi/bittle

[31] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[32] J. Borras and T. Asfour, "A whole-body pose taxonomy for loco-manipulation tasks," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1578–1585.

[33] J. Borràs *et al.*, "A whole-body support pose taxonomy for multi-contact humanoid robot motions," *Science Robotics*, vol. 2, no. 13, 2017.

[34] J. C. Bezdek and N. R. Pal, "Cluster validation with generalized dunn's indices," in *Proceedings 1995 second New Zealand international two-stream conference on artificial neural networks and expert systems*, 1995, pp. 190–193.