2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

State Estimation Transformers for Agile Legged Locomotion

Chen Yu[1*], Yichu Yang[2], Tianlin Liu[2], Yangwei You[2], Mingliang Zhou[2], and Diyun Xiang[2†]

Fig. 1: We demonstrate the effectiveness of the State Estimation Transformer (SET) in pushing the limit of agile locomotion on several challenging jumping tasks. The Cyberdog2 robot can successfully jump while running (the first row), backflip while running (the second row), and sideflip while running (the third row) in the wild.

***Abstract*— We propose a state estimation method that can accurately predict the robot's privileged states to push the limits of quadruped robots in executing advanced skills such as jumping in the wild. In particular, we present the State Estimation Transformers (SET), an architecture that casts the state estimation problem as conditional sequence modeling. SET outputs the robot states that are hard to obtain directly in the real world, such as the body height and velocities, by leveraging a causally masked Transformer. By conditioning an autoregressive model on the robot's past states, our SET model can predict these privileged observations accurately even in highly dynamic locomotions. We evaluate our methods on three tasks — running jumping, running backflipping, and running sideslipping — on a low-cost quadruped robot, Cyberdog2. Results show that SET can outperform other methods in estimation accuracy and transferability in the simulation as well as success rates of jumping and triggering a recovery controller in the real world, suggesting the superiority of such a Transformer-based explicit state estimator in highly dynamic locomotion tasks.**

I. Introduction

Legged animals exhibit remarkable agility — such as galloping, bounding, and jumping — inspiring researchers in the field of legged robots to replicate their impressive locomotion skills. Nevertheless, achieving comparable capabilities in robotic systems has proven to be a persistent challenge for the robotics community [1]. Reinforcement learning (RL) offers a potent framework for autonomously acquiring skills in robotics.

*Work done during the internship at Xiaomi Robotics Lab.
[1]Center for Robotics and Biosystems, Northwestern University, Evanston, IL, USA
[2]Xiaomi Robotics Lab, Beijing, China
†Corresponding author. Email: xiangdiyun@gmail.com

For quadruped walking control, RL algorithms have demonstrated their effectiveness and robustness in challenging terrains. One of the most popular methods is the student-teacher architecture [2], which trains a teacher policy with privileged information such as height maps and then trains a student policy to reproduce the teacher's output using non-privileged observations. This architecture and its variants have allowed quadruped robots to traverse multiple challenging terrains, such as stairs [3], sand [4], [5], mattresses [6], and curbs [7]. Another promising branch of RL learns behaviors from motion capture data via imitation learning. For example, Adversarial Motion Priors (AMP) [8], [9] uses a discriminator to predict the style reward that encourages physically plausible behaviors. These methods have seen many successes in controlling the robots both robustly and naturally [10], [11], [12].

Besides quadruped walking, these RL algorithms have also endowed quadruped robots with more dynamic skills, such as biped stepping [13], bipedal walking [1], jumping [1], backflipping [13], [14], soccer shooting [15], and even extreme parkour [16], [17], [18], [19]. These agile behaviors show the effectiveness of RL-based controllers, enabling legged robots to navigate complex environments that are typically accessible to humans.

However, these existing control approaches for quadrupedal locomotion depend on accurate state estimation [20], and because such estimations are usually not reliable for dynamic locomotion, they can limit the potential of RL algorithms to explore some more agile behaviors, such as dynamic jumping in the parkour. Ji et al. [20] explicitly train a state estimator while concurrently training

the policy, leading to a more robust running controller even on a slippery plate. Before that, Kim et al. [21] define this problem as a Maximum A Posteriori (MAP) estimation problem and solve it with the Gauss-Newton algorithm; Hartley et al. [22] develop a contact-aided invariant extended Kalman filter using the theory of Lie groups and invariant observer design. Compared with some implicit state estimations such as the student-teacher architecture [2], [6], these interpretable state estimators can be used in conjunction with other modules that also require state information and reduce computational complexity [20]. However, none of these state methods have shown their ability in reliable estimation in the case of agile skills with aerial phases and unexpected contacts such as jumping. Also, none of these previous works have demonstrated the potential of explicit state estimators in transferring to other tasks.

There are recent works formulating RL as a sequence modeling problem [23], [24]. Decision Transformer (DT) [25], [26], [27] uses state, action, and the sum of future rewards as tokens in a Transformer model. Trajectory Transformer [28] uses a Transformer model to learn the dynamics of a robot and uses beam search [29] for planning. Embodiment-aware Transformer [30] applies a variant of DT to shape varying robots and Terrain Transformer [4] applies a variant of DT to student-teacher architecture for robust locomotion control. These Transformer-based methods have attained comparable or superior results in standard evaluation tasks when compared to traditional reinforcement learning algorithms, owing to the model's capacity and the self-attention mechanism.

In this work, rather than directly using Transformers to solve the whole control problems, we take advantage of Transformer models to design a state estimator to allow robots to perform more dynamic and agile behaviors without perfect environment information. In particular, we present the State Estimation Transformer (SET), an architecture that casts the state estimation problem as conditional sequence modeling. SET uses a Transformer architecture to model distributions over robots' non-privileged states and privileged states and outputs the estimated privileged states by leveraging the causally masked Transformer.

We use three jumping tasks to show the advantages of our SET methods compared with traditional MLP estimators; we show the benefits of such an explicit state estimator compared with implicit state estimation by a cross-transferring experiment and additionally deploying a reset policy on the real robots.

The key contributions of this work are:

- Proposing a novel state estimation algorithm, SET;
- Designing and training a set of deployable jumping skills leveraging SET;
- Evaluating the estimation accuracy and transferability of SET and its alternatives.

To the best of our knowledge, this is the first time that Transformers are used for state estimation of a legged robot, and this is the first time that a set of advanced jumping skills including running backflipping and running sideflipping are deployed on a low-cost quadruped robot, as shown in Fig. 1.

II. Preliminaries

The Transformer model, originally introduced by Vaswani et al. [31], is designed to efficiently model sequential data and has demonstrated remarkable performance across a spectrum of tasks, spanning Natural Language Processing [32], [33] and Computer Vision [34], [35]. Its architecture comprises a series of stacked self-attention layers with residual connections.

In each self-attention layer, the model takes an input sequence of symbol representations $(x_1, \ldots, x_n)$ with a context length of n and transforms it into a sequence of continuous representations $\mathbf{z} = (z_1, \ldots, z_n)$. This transformation involves linearly mapping each token to a key k_i, query q_i, and value v_i. The self-attention mechanism computes the output for a given token by weighting the values v_j based on the normalized dot product between the query q_i and the other keys k_j, as described by the equation below:

$$z_i = \sum_{j=1}^{n} \operatorname{softmax}\left(\frac{\{\langle q_i, k_{j'} \rangle\}_{j'=1}^{n}}{\sqrt{d_k}} \right)_j \cdot v_j, \qquad (1)$$

where d_k represents the dimensionality of the queries and keys.

III. State Estimation Transformer

For a deployable highly dynamic locomotion controller, we present a two-stage training framework consisting of training a policy leveraging privileged state information in the simulation and training a state estimator to predict the privileged state information that is not directly deployable on the real robot.

A. Markov Decision Process and State Estimation

We model the control of the robot as a Markov decision process (MDP), described by the tuple $(\mathcal{S}, \mathcal{A}, P_E, \mathcal{R})$. The MDP tuple consists of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, state transition dynamics $P_E(\cdot \mid s, a; e)$, and a reward function $r = \mathcal{R}(s, a)$. We use s_t, a_t, and r_t to denote state, action, and reward at timestep t, respectively. We assume that s_t consists of privileged observations o'_t and non-privileged observations o_t: $s_t = (o'_t, o_t)$. A state estimator $\mho$ can map the history of non-privileged observations to privileged observations:

$$o'_t = \mho(o_t^H), \qquad (2)$$

where o_t^H represents the non-privileged history observation from the timestep $t-H$ to t, with a length of H; o'_t represents the estimated state at timestep t.

B. State Estimation as a Sequence Modeling Problem

In this work, we cast the state estimation as a sequence modeling problem. We expect a trajectory representation to enable the Transformer to learn meaningful patterns between robot history observations and the current privileged observation, and the Transformer to conditionally generate the privileged observation based on history observations at test time.

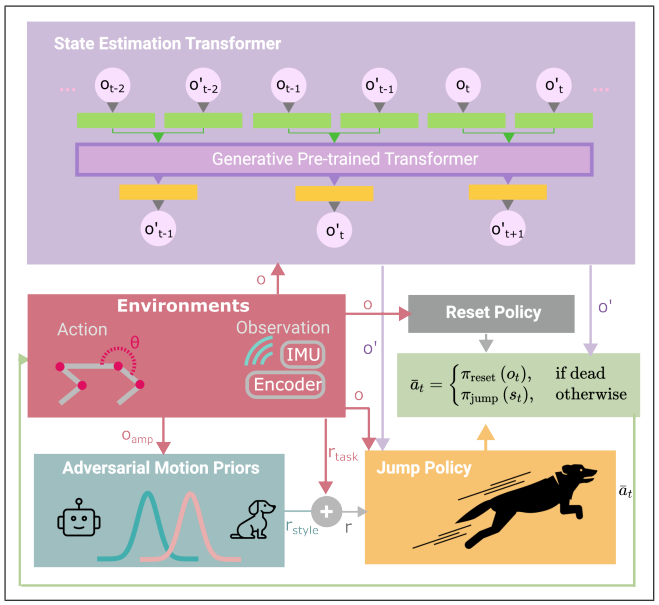

Fig. 2: State Estimation Transformer architecture in our training pipelines of jumping policies. First, we train a *Jump Policy* using the task rewards r_{task} from the *Environments* and the style reward r_{style} calculated by the *Adversarial Motion Priors*; then we train a *State Estimation Transformer* that leverage a GPT model (green blocks represent embedding and position encoding and yellow blocks represent decoders) to predict the privileged observations. To further exploit the benefit of an accurate explicit state estimator, the predicted privileged observations are used as one of the conditions of triggering a built-in *Reset Policy*.

Therefore, we define the following trajectory representation which enables autoregressive training and generation:

$$\tau = (o_1, o_1', o_2, o_2', \ldots, o_T, o_T'). \tag{3}$$

C. State Estimation Transformer

Training. The training process of SET is summarized in Alg. 1. We first train the robot using ground true states s in the simulation and use the trained policy π to generate the initial trajectory dataset D with the representation in Equation (3). For each training step, we sampled H timesteps of each trajectory from D. In terms of token embeddings, we create linear embedding layers for robot history observation o and privileged observation o'. Layer normalization [36] is applied in this process and an embedding for each timestep is added to each token. The tokens, representing these observation pairs, are subsequently fed into a Generative Pre-trained Transformer (GPT) model [37]. This model employs an autoregressive approach to predict future token pairs, specifically (o, o'). The predicted privileged observation o' is instrumental in calculating the mean-squared error during the backpropagation process.

Evaluation.

For evaluating the privileged observations while deploying the policy on a real robot, the robot's initial non-privileged observation will serve as the conditioning information for the initiation of the generation process. This entails supplying

Algorithm 1 State Estimation Transformer (SET)

Input: Initial D, trained policy π
Output: Trained SET Model
1: **for** $i = 0, \ldots, I-1$ iterations **do**
2: Sample H-long (o, o', t) from D
3: Stack embeddings of (o, o') for each timestep
4: Feed the Stacks to the GPT model
5: Update the GPT model
6: **end for**

SET with the most recent H time steps from the ongoing trajectory denoted as τ in order to generate a prediction for the privileged observation for the last timestep.

IV. Learning To Jump with SET

SET is a state estimation algorithm for learning complex locomotion skills. In this section, we focus on robot jumping and its variants as concrete applications for SET and introduce how our framework can be applied to solve them, as summarized in Fig. 2.

A. Robot Setup

We use the Cyberdog2 quadruped robot from Xiaomi [38] as our experimental platform and build our simulation using Isaac Gym [39]. The compact size (0.16 m body length) and relatively lightweight (8.9 kg) of the Cyberdog2 robot enable us to tackle difficult dynamic tasks, while its low-cost actuators with a maximum torque limit of 12 Nm also pose challenges of highly agile locomotion control.

B. Learning to Jump

Here, we describe our implementation of the first stage of our two-stage training framework. We want to train a policy to allow the robot to walk/run at desired velocities and jump at command. Similar to previous work, the action a is the desired joint position of the motors, sent to a PD controller. The state s and the reward r are inspired by [40], as described below.

Observations. As introduced in Section III-A, the state s consists of privileged observation o' and non-privileged observation o. The privileged observation is simply defined as:

$$o' = (h, \mathbf{v}), \tag{4}$$

where h is the height of the robot's Center of Mass (CoM) and $\mathbf{v}$ is the three-dimensional velocities of the robot. The non-privileged observation is defined as:

$$o = (\omega, \phi, \mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}, \mathbf{cmd}), \tag{5}$$

where ω and ϕ are the angular velocity and the base orientation; $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the joint positions and velocities; $\mathbf{p}$ is the Cartesian positions of the feet relative to the robot CoM; and $\mathbf{cmd}$ is a 5-dimensional vector that consists of the velocity commands for the x/y/yaw directions, the target height of jumping, and a Boolean value representing the jump signal. The goal of the robot is to track the commanded

TABLE I: Reward structure.

Reward	Expression	Weight		
		RJ	RB	RS
Linear Velocity	$\phi\left(\mathbf{v}^{\star}_{b,xy} - \mathbf{v}_{b,xy}\right)$	20	20	20
Angular Velocity	$\phi\left(\boldsymbol{\omega}^{\star}_{b,z} - \boldsymbol{\omega}_{b,z}\right)$	6.66	6.66	10
Jump Height	$\begin{cases}\phi(h^{\star}_{jump} - h) \cdot \text{clip}(v_x), & \text{if } sig = 1,\\ \phi(h^{\star}_{walk} - h), & \text{otherwise,}\end{cases}$	5	5	5
Jump Goal	$\begin{cases}\text{clip}(v_x), & \text{if } \lvert h^{\star} - h\rvert < \epsilon_h \text{ and } sig = 1,\\ 0, & \text{otherwise.}\end{cases}$	100	100	100
Jump Height (Roll)	$\begin{cases}\phi(h^{\star}_{jump} - h) \cdot \lvert\phi_{\text{roll}}\rvert, & \text{if } sig = 1,\\ 0, & \text{otherwise.}\end{cases}$	0	0	5
Jump Goal (Roll)	$\begin{cases}\lvert\phi_{\text{roll}}\rvert, & \text{if } \lvert h^{\star} - h\rvert < \epsilon_h \text{ and } sig = 1,\\ 0, & \text{otherwise.}\end{cases}$	0	0	1
Pitch Reward	$\begin{cases}\phi_{\text{pitch}}, & \text{if } falling = 1,\\ 0, & \text{otherwise.}\end{cases}$	10	50	0
Jump Forward	$\left(\text{angle}\left(\phi_{yaw,jumping}, \phi_{yaw,landing}\right)\right)^2$	0	0	-400
Feet Air Time	$\sum_{f=0}^{4}\left(\mathbf{t}_{air,f} - 0.5\right)$	5	5	50
Action Rate	$-\left\lVert\mathbf{q}^{\star}_{j}\right\rVert^2$	0	-10	-10

velocities, and jump to the target height when the jump signal is equal to 1. The jump signal will automatically reset to 0 when the robot reaches the target height.

The observations for the AMP discriminator o_{AMP} is the full state s except for the command vector:

$$o_{\text{AMP}} = (\omega, \phi, \mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}). \tag{6}$$

Data of such observations from the robots and from the motion capture dataset of a jumping and galloping Shepherd dog [41] will be used to train the AMP discriminator.

Rewards. Besides the common tracking terms similar to previous works in locomotion learning [42], [20], [6] and a style reward evaluated by the AMP discriminator [9], [10], we define different reward functions for three different expected jumping gaits that are inspired by human Parkour:

- *Running Jump (RJ):* The robot jumps forward while maintaining its running momentum.
- *Running Backflip (RB):* The robot performs a backflip while running forward and seamlessly continues forward motion upon completing the backflip.
- *Running Sideflip (RS):* The robot executes a side-flip while in forward motion and smoothly resumes running forward after completing the side-flip.

For the *Running Jump* gait, we expect the robot to jump as close as the target height $h^{\star}_{jump}$ when the jump signal $jump\_sig = 1$ and keep close to the walking target height $h^{\star}_{walk}$ when $jump\_sig = 0$. Therefore, we design the jumping reward as follows:

$$\begin{cases}\phi(h^{\star}_{jump} - h) \cdot \text{clip}(v_x, -0.5, 2), & \text{if } jump\_sig = 1,\\ \phi(h^{\star}_{walk} - h), & \text{otherwise,}\end{cases} \tag{7}$$

where $\phi(x) := \exp\left(-\frac{\|x\|^2}{0.25}\right)$ and v_x represents the forward velocity. In addition, to encourage the robot to reach the jumping goal, we add a bonus to the robot while reaching the goal:

$$\begin{cases}\text{clip}(v_x, -0.5, 2), & \text{if } \lvert h^{\star}_{jump} - h\rvert < \epsilon_h \text{ and } jump\_sig = 1,\\ 0, & \text{otherwise.}\end{cases} \tag{8}$$

As the robot has only limited torques, we additionally design a pitch reward to prevent the robot from cheating by standing on two legs as follows in the falling phase of the jumping:

$$\begin{cases}\phi_{\text{pitch}}, & \text{if } falling = 1,\\ 0, & \text{otherwise.}\end{cases} \tag{9}$$

The falling phase $falling = 1$ is defined as the period between the robot reaching the jumping goal and one of its feet contacts with the ground.

We define the pitch angle ϕ_{pitch} as follows: It varies from 0 radians (normal pose) to $\pi/2$ radians (upward tilt) and then abruptly switches to $-\pi/2$ radians, with a gradual return to 0 radians when rotating backward. The beauty of such a pitch reward and definition is that we can use it to switch between the *Running Jump* gait and the *Running backflip* gait: a small weight for this pitch reward can lead to a *Running Jump* gait while a big weight can encourage the robot to backflip when receiving the jump signal.

We also design a variant of the Equation (7) for the side-flipping task while encouraging the rotation of the robot in the roll angle:

$$\begin{cases}\phi(h^{\star}_{jump} - h) \cdot \lvert\phi_{\text{roll}}\rvert, & \text{if } jump\_sig = 1,\\ 0, & \text{otherwise.}\end{cases} \tag{10}$$

Similarly, a successful jump reward for side-flipping is designed as follows:

$$\begin{cases}\lvert\phi_{\text{roll}}\rvert, & \text{if } \lvert h^{\star}_{jump} - h\rvert < \epsilon_h \text{ and } jump\_sig = 1,\\ 0, & \text{otherwise.}\end{cases} \tag{11}$$

Other auxiliary rewards [43] include the feet-in-air reward, action rate penalty, and reward for robot jumping forward in the sideflipping gait are summarized in Tab. I, as well as the weights of all reward terms.

Curriculum in torque limit. While we focus on a low-cost quadruped robot with motors with a maximum torque of only 12 Nm, we find that starting from a more relaxed torque limit in the simulation can significantly help the agents' exploration. Therefore, we set the torque limits of all motors as 300 Nm at the beginning of the training and gradually decrease the maximum torque limit to 12 Nm.

C. Reset Policy

One of the benefits of an explicit state estimator such as SET is that it can be used in conjunction with other modules [20] although hardly previous works have exploited this feature. In this work, the state estimator can share the estimated results between the jumping policy π_{jump} with another built-in reset policy π_{reset}, which can help to reset the robot when it fails the jumping and falls on the ground when the environment is too noisy.

Traditionally, a recovery controller can kick in when the robot is detected as upside-down [44], [45]. However, for our *Jumping Backflip* and *Jumping Sideflip* gaits, additional height information should be used to tell the difference between the normal in-the-air phase or unexpected falling

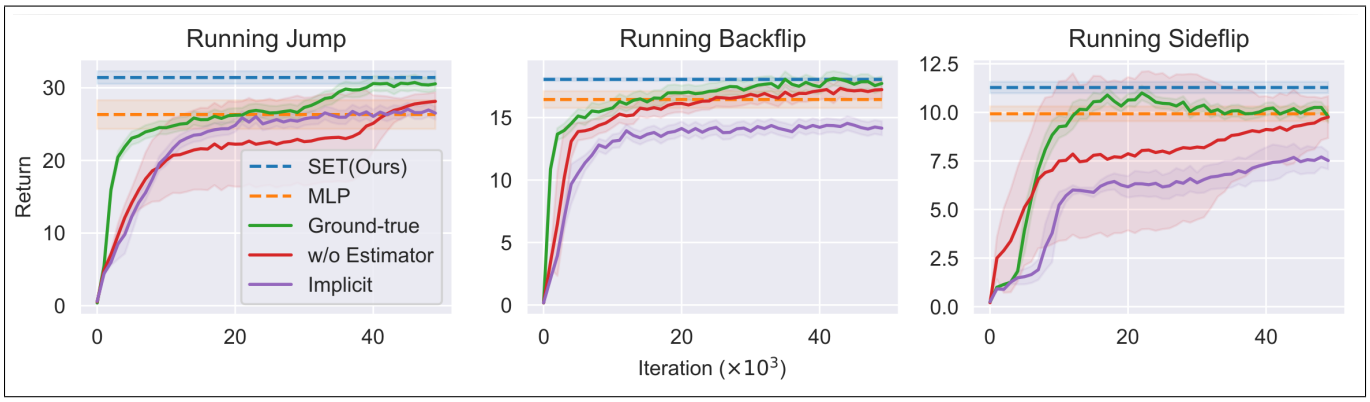

Fig. 3: Training curves of different algorithms on three jumping gaits. We compare the training curves using the ground-true full observations (*Ground-true*), only non-privileged observations (*w/o Estimator*), 20 historical stacked non-privileged observations (*Implicit*), and the results using our SET and MLP estimators. Results show that for highly dynamic locomotion, the privileged observations such as robot velocity and height can significantly increase the efficiency and robustness of policy training; and our deployable method can reproduce the results trained with ground-true privileged observations.

TABLE II: Tracking and estimation error for a walking policy

Task	Model	V_X [m/s]	V_Y [m/s]
Tracking Error	SET (Ours)	0.0606	0.0676
	MLP	0.1621	0.1636
	w/o Estimator	2.620	1.315
	Implicit	1.811	0.9966
	Ground-true	0.0598	0.0610
Estimation Error	SET (Ours)	0.0179	0.0172
	MLP	0.1663	0.1445
	SET-1e4	0.1323	0.06822
	SET-H1	0.08784	0.07742

on the ground. Therefore, we have the final actions $\bar{a}$ as follows:

$$\bar{a}_t = \begin{cases} \pi_{\text{reset}}(o_t), & \text{if } \mathbf{g}^p \cdot \mathbf{g} < \epsilon_r \text{ and } h < h^{\star}_{\text{walk}}, \\ \pi_{\text{jump}}(s_t), & \text{otherwise}, \end{cases} \tag{12}$$

$$\bar{a}_t = \begin{cases} \pi_{\text{reset}}(o_t), & \text{if } dead, \\ \pi_{\text{jump}}(s_t), & \text{otherwise} \end{cases} \tag{13}$$

where $\mathbf{g}^p \cdot \mathbf{g}$ is the cosine similarity between the projected gravity $\mathbf{g}^p$ and the gravity vector (0, 0, -1) $\mathbf{g}$.

V. Simulation Results

A. Walking

Firstly, we evaluate our estimator in a simple walking scenario with a similar experimental setup as in [20]. We train a walking policy using PPO on a flat plane. The observation is the same as our jumping observation with $jump\_sig = 0$ and the reward function only contains the velocity tracking terms. We train the robot with full observation s and then train a state estimator based on the trajectories of the last trained policy. Such a setting is called a *sequential model* in [20]. The number of historical steps H we used for the SET is 20 and for MLP is 5 since the performance of MLP will drop significantly when H increases; we set the number of blocks as 6 for SET and the number of layers as 3 for MLP since this is also the best parameters we have tuned for MLP. We also evaluate the performance of the following methods:

- *Ground-true*: we train a policy with the ground-truth privileged robot states and non-privileged states;
- *w/o Estimator*: we train a policy with only non-privileged states;
- *Implicit*: we train a policy with 20 stacked historical non-privileged states;
- *SET*: we train a SET based on trajectories of the final policy using ground-true privileged observations;
- *MLP*: we train an MLP estimator based on trajectories of the final policy using ground-true privileged observations.

Tab. II shows the tracking and estimation error for the walking policy. SET demonstrates a lower command following errors and estimation errors compared to MLP, *w/o Estimator*, and *Implicit*, which is similar to the *Ground-true*.

We hypothesize that the superiority of SET comes from the context information of previous tokens and the capacity of the model for fitting our diverse training dataset that contains both varying trajectories. To investigate the importance of access to previous non-privileged states, we ablate on the context length H. The performance of SET degrades significantly when the number of historical observations H is 1 (SET-H1 in Tab. II), indicating that past information is essential for this state estimation task.

We further investigate the impact of the training dataset on the performance of SET. We decrease the number of trajectories in the training dataset from 1×10^5 to 1×10^4, denoted as Model *SET-1e4* in Tab. II. This reduction of dataset size also yields degraded performance of SET, suggesting that the high complicity of SET allows it to leverage all the information in the dataset and implicitly build the associations between non-privileged observations and privileged observations via the similarity of the query and key vectors. This makes it superior in privileged state estimation.

TABLE III: Estimated errors of different jumping gaits

	Jumping				Backflipping				Rolling			
	h	v_x	v_y	v_z	h	v_x	v_y	v_z	h	v_x	v_y	v_z
SET (Ours)	**0.001091**	**0.01393**	**0.0065**	**0.0091**	**0.001438**	**0.01873**	**0.007415**	**0.01218**	**0.001113**	**0.00905**	**0.007131**	**0.00686**
MLP	0.007877	0.09994	0.04037	0.06498	0.003593	0.05691	0.02224	0.03335	0.005348	0.04553	0.03514	0.03766

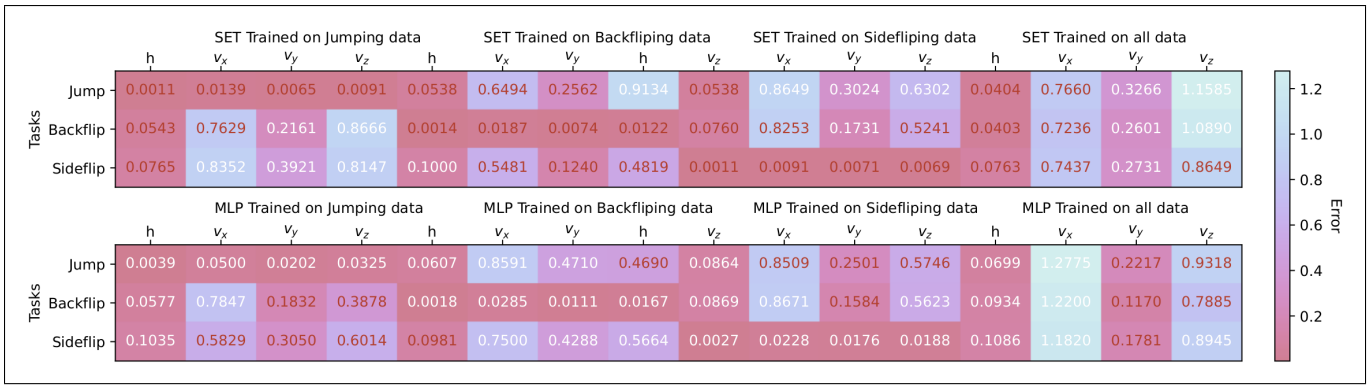

Fig. 4: Prediction error of the estimators that are trained on different data and applied on different tasks. Each pair of training dataset / application task correspond to four adjacent items in the heatmaps, representing errors in the estimation of h, v_x, v_y, and v_z. We show the results from SET and MLP to compare their abilities in generalization, while SET outperforms MLP in 2/3 (32/48, highlighted in red) cases. These results suggest the abilities of a trained explicit estimator to be applied to another task; and demonstrate the superiority of SET over MLP in explicit state estimation.

B. Jumping

Learning to Jump. To evaluate and analyze the proposed methods, we test the performance of SET and alternative methods *w/o Estimator*, *Implicit*, and *MLP* as well as *Ground-true* as described in Section V-A for training three jumping gaits.

Fig. 3 shows the training curves of different algorithms on three jumping gaits, averaged over three trials. Results show that for all of these three jumping gaits, the privileged observations are essential: agents with the explicit ground-true privileged observations show a much more efficient and stable training process compared with using stacked historical non-privileged observation (*Implicit*) or without privileged observations (*w/o Estimator*). The results of SET also show that SET can reproduce the performance of the policy trained with privileged observations, even if it only requires non-privileged observations. These results are consistent with that in Section V-A.

State Estimation. We further compare the estimated errors between SET and MLP estimators on these three jumping tasks, as shown in Tab. III. We demonstrate the RMS errors averaged over 2048 trials on three jumping tasks and four dimensions of predicted privileged observations. Results show that SET can significantly outperform MLP estimators in all of the cases.

C. Transferring

As we have mentioned in Section IV-C, one of the advantages of explicit estimators is that their estimated results can be used for different downstream tasks; this feature can be further highlighted by the transferability of Transformers. To test the transferability of different estimators, we evaluate the performance of estimators on the training tasks that may be different from the training dataset.

In Fig. 4, we show the estimated errors of implementing the estimator that are trained on different training datasets (x-axis) on different tasks (y-axis). Results show that in 2/3 cases SET can predict the privileged observations better than MLP. Interestingly, for estimators that all trained on all data, SET shows a significantly more accurate estimation in body height and x velocity — which are the two most important privileged information for jumping tasks — than MLP, making it more suitable than MLP to train a distilled estimator that can be used for all tasks for such agile and dynamic tasks.

VI. Experimental Results

We directly deploy the policy trained with privileged observations on the real robot and use SET or MLP estimator to estimate those privileged states. Note that although alternative methods such as the *Stacked obs* evaluated in Section V-B also do not require privileged observations for their policy, it is still challenging to deploy them on the real robots since we need the height information to reset the jump signal.

A. Jumping in the Real-world

All three jumping policies with SET can be successfully deployed on the real robots as shown in Fig. 1: Our robots can robustly jump, backflip, and sideflip seamlessly during a dynamic running process in the wild as commanded. More demonstrations can be seen in the supplementary videos.

Quantitatively, we compare the success rates of jumping using SET and MLP estimators. Each jumping policy is tested twenty times, and the overall success rate is shown in Tab. IV. We show the success rates of the *Running Jump*

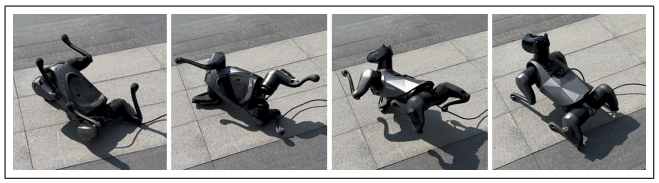

Fig. 5: Recovering from a failed jumping. The estimated robot height from the estimator is used as one of the conditions for triggering the reset policy.

policy, *Running Backflip* policy, and *Running Sideflip* policy paired with SET or MLP estimators on the real robots, as well as the overall success rates using the distilled policy trained with all jumping data and evaluating it on all jumping tasks.

Results show that SET can outperform MLP on the Backfliping and Sidefliping tasks when the policy is trained using a task-specific dataset; SET can also significantly outperform MLP when a general-purpose estimator is trained using training data from all tasks, which suggests the superiority of SET in convenient real-robot deployment. For the jumping gait, the lower success rate observed in SET may be attributed to its longer observation horizon, which makes it more sensitive to real-world noise.

B. Reseting at Failure

As described in Section IV-C, we use the predicted robot height from the estimators as one of the conditions for triggering the reset policy. Note that since our policy can have an almost 100% success rate in the simulators, our estimators have limited training samples of failed jumpings in the training phase. Therefore, this is also a test of the generalization ability and sample efficiency of the estimators.

Fig. 5 shows the snapshots of a robot resetting from a failed jumping. We also compare the successful rate of resetting between using SET and MLP as the height estimator in Tab. IV. SET also outperforms MLP in this application in real-world deployment, which is consistent with the results of estimation accuracy (Tab. III) and generalization ability (Fig. 4) in simulation.

VII. Conclusions

In this work, we propose a state estimation method SET for agile locomotion control. We pose the state estimation challenge as a conditional sequence modeling problem, and use Transformer to fit a data sequence of (non-privileged observations, privileged observations) tuples. The proposed methods are evaluated on three AMP-based jumping tasks in a two-stage training manner: Firstly the policies are trained with privileged observations and then estimators are trained to predict these observations. After showing that privileged observations are essential information for such dynamic and agile locomotion tasks, we demonstrate that SET can show superiority in estimation accuracy, transferability, and real-world success rates compared to MLP. To further exploit the benefits of explicit state estimation, we use the output from the estimator as one of the conditions of performing a recovery controller, showing the practical application of our estimators in real-world applications. It would be our future work to integrate these agile locomotion skills into an autonomous planning framework to unlock the full potential for real-world application of low-cost quadruped robots.

TABLE IV: Success rates in the real world

	Jump	Backflip	Sideflip	Distillation	Reset
SET	75.0%	**85.0%**	**85.0%**	**85.0%**	**100.0%**
MLP	**85.0%**	80.0%	70.0%	60%	75.0%

References

[1] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Learning and adapting agile locomotion skills by transferring experience," *arXiv preprint arXiv:2304.09834*, 2023.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[3] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5078–5084.

[4] H. Lai, W. Zhang, X. He, C. Yu, Z. Tian, Y. Yu, and J. Wang, "Sim-to-real transfer for quadrupedal locomotion via terrain transformer," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5141–5147.

[5] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, p. eade2256, 2023.

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021.

[7] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning*. PMLR, 2023, pp. 403–415.

[8] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[9] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

[10] J. Wu, G. Xin, C. Qi, and Y. Xue, "Learning robust and agile legged locomotion using adversarial motion priors," *IEEE Robotics and Automation Letters*, 2023.

[11] Y. Wang, Z. Jiang, and J. Chen, "Amp in the wild: Learning robust, agile, natural legged locomotion skills," *arXiv preprint arXiv:2304.10888*, 2023.

[12] J. Wu, Y. Xue, and C. Qi, "Learning multiple gaits within latent space for quadruped robots," *arXiv preprint arXiv:2308.03014*, 2023.

[13] Y. Fuchioka, Z. Xie, and M. Van de Panne, "Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5092–5098.

[14] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimminger, and G. Martius, "Learning agile skills via adversarial imitation of rough partial demonstrations," in *Conference on Robot Learning*. PMLR, 2023, pp. 342–352.

[15] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath, "Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1479–1486.

[16] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[17] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[18] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *arXiv preprint arXiv:2306.14874*, 2023.

[19] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.

[20] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[21] J.-H. Kim, S. Hong, G. Ji, S. Jeon, J. Hwangbo, J.-H. Oh, and H.-W. Park, "Legged robot state estimation with dynamic contact event information," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6733–6740, 2021.

[22] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

[23] R. K. Srivastava, P. Shyam, F. Mutz, W. Jaśkowski, and J. Schmidhuber, "Training agents using upside-down reinforcement learning," *arXiv preprint arXiv:1912.02877*, 2019.

[24] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[25] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.

[26] M. Reid, Y. Yamada, and S. S. Gu, "Can wikipedia help offline reinforcement learning?" *arXiv preprint arXiv:2201.12122*, 2022.

[27] Q. Zheng, A. Zhang, and A. Grover, "Online decision transformer," *arXiv preprint arXiv:2202.05607*, 2022.

[28] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, pp. 1273–1286, 2021.

[29] R. Reddy, "Speech understanding systems: A summary of results of the five-year research effort at carnegie mellon university," *Pittsburgh, Pa*, 1977.

[30] C. Yu, W. Zhang, H. Lai, Z. Tian, L. Kneip, and J. Wang, "Multi-embodiment legged robot control as a sequence modeling problem," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7250–7257.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[32] N. Kitaev and D. Klein, "Constituency parsing with a self-attentive encoder," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2676–2686. [Online]. Available: https://www.aclweb.org/anthology/P18-1249

[33] P. J. Liu*, M. Saleh*, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=Hyg0vbWC-

[34] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video swin transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3202–3211.

[35] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8844–8854.

[36] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[37] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[38] [Online]. Available: https://www.mi.com/cyberdog2

[39] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[40] F. S. Huiqiao Fu, Yutong Wu, "Metalhead: Natural locomotion, jumping and recovery of quadruped robot a1 with amp," https://github.com/inspirai/MetalHead, 2023.

[41] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[42] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[43] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[44] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[45] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.