Human-AI Shared Control via Policy Dissection

Quanyi Li[‡◇], **Zhenghao Peng**[§], **Haibin Wu**[◇], **Lan Feng**[†], **Bolei Zhou**[§]
[◇]Centre for Perceptual and Interactive Intelligence, [†]ETH Zurich,
[‡]University of Edinburgh, [§]University of California, Los Angeles

Abstract

Human-AI shared control allows human to interact and collaborate with autonomous agents to accomplish control tasks in complex environments. Previous Reinforcement Learning (RL) methods attempted goal-conditioned designs to achieve human-controllable policies at the cost of redesigning the reward function and training paradigm. Inspired by the neuroscience approach to investigate the motor cortex in primates, we develop a simple yet effective frequency-based approach called *Policy Dissection* to align the intermediate representation of the learned neural controller with the kinematic attributes of the agent behavior. Without modifying the neural controller or retraining the model, the proposed approach can convert a given RL-trained policy into a human-controllable policy. We evaluate the proposed approach on many RL tasks such as autonomous driving and locomotion. The experiments show that human-AI shared control system achieved by *Policy Dissection* in driving task can substantially improve the performance and safety in unseen traffic scenes. With human in the inference loop, the locomotion robots also exhibit versatile controllable motion skills even though they are only trained to move forward. Our results suggest the promising direction of implementing human-AI shared autonomy through interpreting the learned representation of the autonomous agents. Code and demo videos are available at https://metadriverse.github.io/policydissect.

1 Introduction

In recent years, autonomous agents trained from deep Reinforcement Learning (RL) have achieved huge success in a range of applications from robot control [69, 56, 37, 41, 73], autonomous driving [25, 7, 29], to the power system in smart building [44]. Despite the capability of discovering feasible policy under unknown system dynamics in a model-free setting [67], the learning-based agents lack sufficient generalizability [14] in unseen environments, which hinders their real-world deployments. Furthermore, it remains difficult to understand the internal mechanism and the decision making of the neural network [30, 79], especially when abnormal behaviors happen [10, 78, 49]. Thus the absence of generalizability, transparency and controllability limits the application of the end-to-end neural controllers in safety-critical systems.

One potential direction to make the neural controller safer and more trustworthy is incorporating human into the *inference* loop, which we refer to as *human-AI shared control*. With human involvement, the shared autonomy systems can achieve substantial improvement over the performance and safety [54, 15, 55], even on new tasks and unseen environments. Human-AI shared control has been implemented in various forms, such as enforcing intervention [28, 51] or providing high-level commands [15] to the AI models. Previous works mainly explore goal-conditioned reinforcement learning (GCRL) as a form of human-AI shared control, where a high-level goal is fed as an input to the policy during training [60, 2, 15, 25, 57]. Thus the behavior of the learned agent can be controlled by varying the input goal [47, 38, 80, 37]. However, training goal-conditioned policy requires additional modification to the observation design, the reward function and the training

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

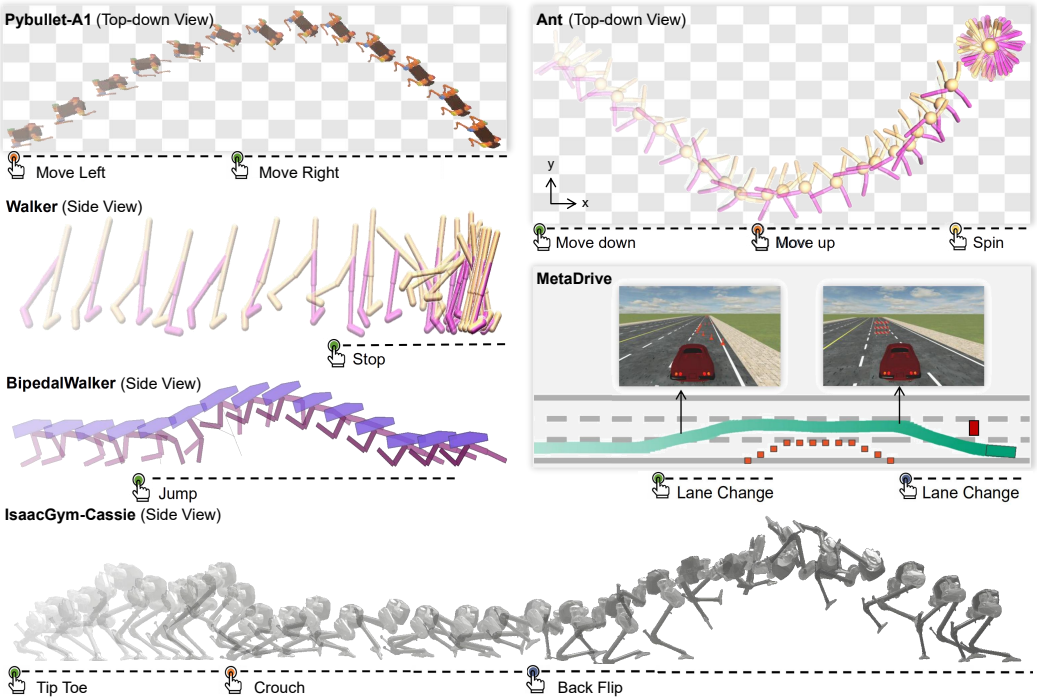


Figure 1: Examples of human-AI shared control enabled by *Policy Dissection*. Each autonomous agent is trained to solve its primal task without changing the environment like the reward function and observation. After training, the *stimulation-evoked map* is then generated by *Policy Dissection* as the interface for human to steer the agent. In various environments, behaviors such as jumping, spinning, redirection, backflipping can be triggered by human subject. These examples demonstrate that *Policy Dissection* is a general method for making a pretrained neural policy controllable.

process. In the case of bipedal robot for example, it is difficult to design reward functions for each behavior like crouching, backflipping, forward jumping. Also, initial state distribution and goal selection criteria should be meticulously designed for a successful GCRL training [50, 57]. One the other hand, the human control only happens at the goal level, still lacking in the interpretability of low-level neural policy.

In this work, we develop a minimalist approach called *Policy Dissection*, to enable human-AI shared control on autonomous agents trained on a wide range of tasks. *Policy Dissection* does not impose assumption on the agent's training scheme like specially-designed reward function, and it requires neither retraining the controller's network nor modifying the environment. *Policy Dissection* achieves human-AI shared control by firstly dissecting the internal representations of the learned policy and aligning them with specific kinematic attributes, and then modulating the activation of the internal representations to elicit the desired motor skills.

Policy Dissection is inspired by the neuroscience studies on motor cortex of primates, where exerting electrical stimulation on different areas of motor cortex can elicit meaningful body movements [21]. Thus, a *stimulation-evoked map* can be built to reveal the relationship between the evoked body movements and the motor neurons located in different areas of motor cortex [21, 20, 34]. In order to replicate such a *stimulation-evoked map* inside a controller powered by artificial neural network, the proposed *Policy Dissection* conducts a frequency analysis and matching strategy for associating the kinematic attribute and the unit activation. The procedure of building a *stimulation-evoked map* by *Policy Dissection* is illustrated in Fig. 2. Concretely, we first roll out the given well-trained policy for many episodes and collect a set of time series of the activities of all units, and the variation in kinematics and dynamics. Afterwards, for each kinematic attribute we identify one unit whose activity align best with the variation of this kinematic attribute following the principle that there is the lowest frequency discrepancy between both the time series. After finishing the alignment for

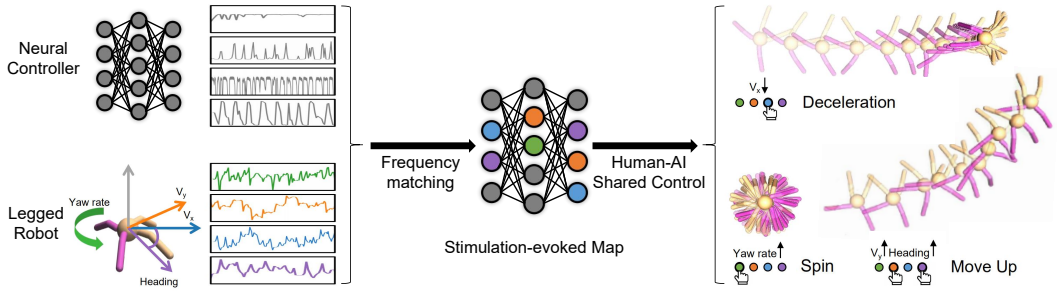


Figure 2: Overview of *Policy Dissection*. Taking Ant robot as an example, our method first identifies the connection between *motor primitives* and kinematic attributes (painted in the same colors). After that, human can activate units to evoke desired behaviors by stimulating one or more *motor primitives*. For example, stimulating *motor primitive* related to yaw rate makes the Ant spin, while activating *motor primitive* related to velocities in different axes brings moving-up or deceleration.

all kinematic attributes, the identified units are then called *motor primitives*[1] whose activation can change corresponding kinematic attributes. Since a behavior can be described by the change of one or more kinematic attributes, the *stimulation-evoked map* can be built for recording the relationship of *motor-primitives*' activation and evoked behaviors. For example, the moving up behavior of the Gym-Ant shown in Fig. 2 can be described by turning upward (increasing yaw) and increasing upward speed (v_y), and thus we can activate corresponding two *motor primitives* to elicit the desired movement. In conclusion, *Policy Dissection* constructs an interpretable control interface on top of the trained agent for human to interact with and evoke certain behaviors of the agent, thus achieving human-AI shared control.

We evaluate the proposed method on several RL agents ranging from locomotion robots to autonomous driving agents in simulation environments. Experimental results suggest that meaningful *motor primitives* emerge in the internal representation. As shown in Fig. 1, novel behaviors can be deliberately evoked by activating units related to certain kinematic attributes, even though they are not the necessary behaviors to solve the primal task. In the quantitative evaluation, we use the human-AI shared control system enabled by *Policy Dissection* to improve the generalization in test-time unseen environment and achieve zero-shot task transfer. In an autonomous driving task, we first train baseline agents under mild traffic conditions and evaluate the driving policy on new test environments containing unseen near-accidental scenes. Different from the poor performance of the baseline agent in these new environments, the human-AI shared control system achieves superior performance and safety guarantee, with 95% success rate and almost zero cost. In quadrupedal locomotion task, a robot with only proprioceptive state input can avoid obstacles with shared control, even though its primal task is moving forward. These results show that the proposed *Policy Dissection* not only helps understand the learned representations of the neural controller, but also brings the intriguing potential application of human-AI shared control.

2 Related Work

Human-AI Shared Control. Existing methods on human-AI shared control can be roughly divided into two categories. The first category is to have human in the training loop and then the well-trained policy can be executed without human in test time [81]. By incorporating human into the training, previous works successfully improve the performance on visual control tasks such as Atari game [1, 58, 71]. Robotic control tasks also benefit from human feedback [28, 43, 65, 51, 72]. The other category is to have human in both training and test time to accurately accomplish human-assistive tasks. A series of works study this problem based on the Atari Game [54, 59, 8, 31]. In addition, the human-AI shared control system is built on autonomous vehicle [19], robotic arms [27] and in multi-agent setting [31]. The reported results on various tasks indicate the effectiveness and efficiency of training and coordination with human-assistive AI. Unlike prior works, our method allows human to collaborate with AI in test-time, while eschewing the need for any training-time

[1] Motor primitive is used by researchers in biological motor control to indicate "building blocks of movement generation" [52, 32]

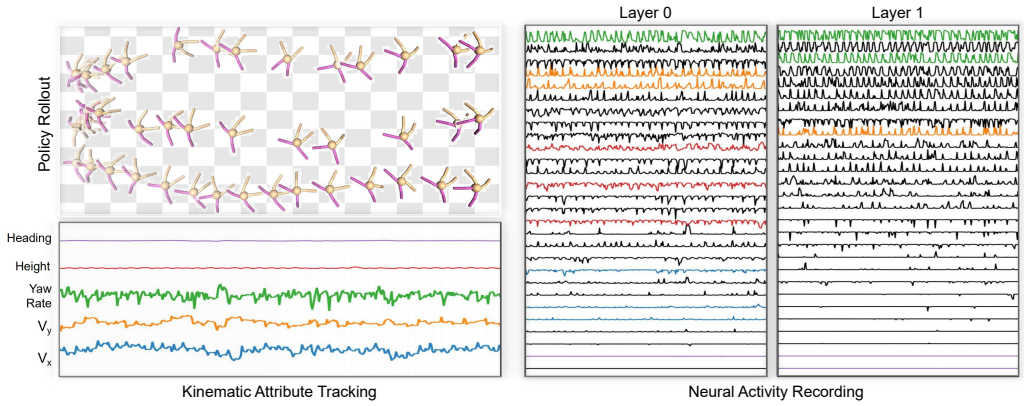


Figure 3: We first roll out the trained policy and record the neural activities and track kinematic attributes, like yaw and velocity. After frequency matching, kinematic attributes are associated with certain units, which are further called *motor primitives*. The curves of kinematic attributes and the aligned *motor primitive* are painted in the same colors. For clarity, we only show the result of one recorded episode and a proportion of units, and the curves of units are sorted by their amplitude.

human involvement. In experiments, shared control systems are implemented on several robotic control tasks to show the effectiveness of our method.

Neural Network Interpretability. There is a growing interest in the AI community to understand what is learned inside the neural network. Previous interpretability methods mostly focus on explaining the networks trained for visual tasks [82, 63, 11, 83, 40]. Various empirical studies have found that meaningful and interpretable structures emerge in the internal representation when trained for the primal tasks. For example, object detectors emerge in scene classifier [6], and disentangled layout and attribute controllers emerge from image generation model [75]. Varying the disentangled hidden features of the deep generative model can edit the content of the generated images. [5, 64, 74]. On the other hand, visuomotor control tasks can also acquire visual understanding to explain the behavior chosen by agent with visual input through saliency map [66, 3, 53, 12, 26, 22].

Neuroscience Inspired Control. The complex behaviors shown by recent autonomous agents prompt researchers to develop new analysis methods. As a source of inspiration, neuroscience provides useful tools [48, 34], and effective procedures [24, 18, 46] for understanding animal brains which are also like black-box systems. Recent works try to develop neural network-based policies from the perspective of neuroscience [17, 4, 77, 13]. In a closely related work on understanding learned agent, Merel et al. [46] use RSA [35] with CKA [33] similarity index to reveal that behaviors spanning multiple timescales are encoded in distinct neural layers. Apart from providing the neuroscientific tools, many concepts from neuroscience, such as *motor primitive* also inspire researchers to design generalizable controllers [70, 68, 45].

3 Policy Dissection Method

Policy Dissection aims at deriving a human-controllable policy from a well-trained RL agent. It has two unique features to enable human-AI shared control: First, it operates on a well-trained RL agent without re-training or modifying the original reward/environment and the policy architecture. We only need to roll out the agent for several episodes and log its neural activity and the change of kinematic attributes. Based on the collected data, kinematic attributes are aligned with certain units which are further termed as *motor primitives*. Thus our method is non-intrusive to the primal task and trained model, and applicable to a wide range of environments. Second, activating certain *motor primitives* can elicit generalizable motor skills that even the agent is not trained to perform. As shown in Fig. 1, the bipedal Cassie robot can be steered to tip toe, crouch and back flip even if it is only trained to move forward in the primal task. In the following subsections, we will introduce the four essential steps of the *Policy Dissection* method in detail.

3.1 Monitoring Neural Activity and Kinematics

Since multi-layer perceptron (MLP) is the most common network architecture used in many RL tasks, we assume the neural controller is an MLP with $\mathcal{Z}$ units, which has L hidden layers and all layers have the same number of I hidden units[2]. We denote a neuron by l, i as $z^{l,i} \in \mathcal{Z}$, where $l = 1, ..., L$ is the index of layer and $i = 1, ..., I$ is the neuron index in each layer, and its output at time step t is $z_t^{l,i}$. Let J be the total number of kinematic attributes. The agent's kinematic attributes are represented by $\mathcal{S} = \{s^j\}$, $j = 1, ..., J$, and the measurement of each kinematic attribute s^j at time step t is s_t^j. We define the *motor primitive* as the unit $m^j \in Z$ that can steers the agent and cause change of the corresponding kinematic attribute. We assume that each kinematic attribute s^j has a corresponding *motor primitive* m^j to drive the attribute to a desired state, and thus there are also J *motor primitives*, denoted by $\mathcal{M} = \{m^j\}$, $j = 1, ..., J$. Our goal is to discover the *motor primitive* m^j by associating the units with certain kinematic attribute s^j.

As shown in Fig. 3, we take the Ant robot as an example. *Policy Dissection* starts by rolling out the pretrained agent for several episodes and tracking the activation of all units as well as the kinematics, such as yaw rate and velocity. We record the $I \cdot L$ time series $\{\mathbf{z}^{l,i} = [z_1^{l,i}, z_2^{l,i}, ...]\}^{I \cdot L}$ measuring the neural activities and J time series of kinematic attributes $\{\mathbf{s}^j = [s_1^j, s_2^j, ...]\}^J$ for further analysis.

3.2 Associating Units with Kinematic Attributes

Observing the neural activity and kinematic measurement recorded in Fig. 3, we can find that several repeated kinematic patterns appear in different frequencies when executing the policy, which is reminiscent of animal behavioral research [21, 20, 34]. It suggests that the motor skills has its predominant frequency component [46], controlling fast kinematics and slow kinematics. Since neural activation signals also encode distinct frequency information as shown in Fig. 3, this resemblance inspires us to make the following hypothesize:

For one kinematic attribute, the unit with minimal frequency discrepancy is the motor primitive.

To this end, we process time series of neural activity and kinematic attributes in frequency domain, where the difference of predominant frequency between two signals can be clearly revealed and the phase discrepancy of two time series are eliminated. As a consequence, this allows us to analyze the recorded data and associate kinematic attributes with units whose activity best matches the variation of target kinematic attributes. Concretely, all time series $\{\mathbf{x}\}$ including the neural activities $\{\mathbf{z}^{l,i}\}$ and change of kinematic attributes $\{\mathbf{s}^j\}$ are transformed into the frequency domain through Discrete-time *Short-time Fourier transform* (STFT):

$$\text{STFT}(d, \omega|\mathbf{x}) \equiv \sum_{t=-\infty}^{\infty} x_t \text{W}(t-d) e^{-j\omega t}. \tag{1}$$

STFT splits the time series $\mathbf{x} = [x_1, x_2, ...]$ into a sequence of temporal window and $d \in [1, +\infty]$ denotes the time step of the center of the temporal window. STFT then conducts Fourier transform in each temporal window. $\omega \in [0, +\infty]$ denotes the frequency. To reduce spectral leakage and reserve the frequency information as much as possible, the window function $\text{W}(t)$ used in *Policy Dissection* is *blackman window*. Moreover, the window length is 64 with hop length 16. We apply STFT instead of naive Fourier transform since it can retain temporal information by breaking the time series into several overlapping chunks of frames. This feature is important since in the following frequency matching we focus on the similarity of resulting spectrograms as well as the temporal correlation of two signals. We use the spectrogram to characterize a measurement:

$$\text{SG}(d, \omega|\mathbf{x}) \equiv |\text{STFT}(d, \omega|\mathbf{x})|^2. \tag{2}$$

After obtaining the spectrograms for all kinematic attributes and neural activities, *frequency discrepancy* can be computed for each neuron-kinematics pair $(\mathbf{z}^{l,i}, \mathbf{s}^j)$:

$$\text{Dis}(\mathbf{z}^{l,i}, \mathbf{s}^j) = \sum_d \sum_\omega \|\text{SG}(d, \omega|\mathbf{z}^{l,i}) - \text{SG}(d, \omega|\mathbf{s}^j)\|_2. \tag{3}$$

[2]To keep the notation concise, we assume all hidden layers of MLP have the same number of units I. MLPs with varying number of hidden units per layer still come to the same conclusion.

For each kinematic attribute, *frequency matching* is applied to select the unit with minimal average discrepancy across N collected episodes as the *motor primitive* responsible for triggering the change of desired kinematic attributes:

$$m^j = \underset{z^{l,i}}{\text{argmin}} \frac{1}{N} \sum_N \text{Dis}(\mathbf{z}^{l,i}, \mathbf{s}^j), \tag{4}$$

3.3 Building Stimulation-evoked Map

A behavior b^k can be described by changing a subset of kinematic attributes $\mathcal{S}^k \subseteq \mathcal{S}$, which can be achieved by activating a set of corresponding *motor primitives* $\mathcal{M}^k \subseteq \mathcal{M}$. Therefore, these movement generation building blocks, *motor primitives*, are associated with a certain behavior, inducing the *stimulation-evoked map* $\{(\mathcal{M}^k, b^k)\}, k = 1, ..., K$, where K is the number of discovered behaviors. Taking the Cassie's back-flip shown in Fig. 1 as an example, this behavior can be described by increasing 1. height 2. pitch and 3. knee force. Therefore, we activate three corresponding *motor primitives* selected by Eq. 4 with output calculated by Eq. 8 to make the robot back flip. In practice, it is possible that more *motor primitives* are required. Again, taking the back-flip as an example, the force for both knee may be different after activating corresponding units, and thus giving the robot a roll velocity, which can be further eliminated by activating roll related *motor primitive*.

3.4 Steering Agent via Stimulation-evoked Map

Now the last thing for evoking a behavior b^k is to determine the output value for *motor primitives* $\mathcal{S}^k$. At time step t, activating the *motor primitive* m^j related to kinematic attribute s^j stands for applying its derivative $\dot{s}^j$ to the agent. For example, activating the speed related units equals to apply the derivative of speed, acceleration, to the agent. Therefore, changing the kinematic attribute s^j at time step t_0 by activating the *motor primitive* m^j with a unit output v^j can be formulated as:

$$s^j_{t_1} = s^j_{t_0} + \int_{t_0}^{t_1} f(v^j) dt, \tag{5}$$

where $T = t_1 - t_0$ is the activation period of the *motor primitive* m^j, constant v^j is the time-invariant output of m^j and $f(v)$ is a mapping from *motor primitive* output value v^j to $\dot{s}^j$.

For steering the agent, our goal is to understand some properties of $f(v)$, especially the correlation between $f(v)$ and v, so that we can amplify or reduce the s^j by choosing a v or $-v$. Taking the Ant robot of Fig. 1 as an example, activating the unit controlling y-axis movement with *positive* value can drive the Ant move left, and a *negative* value makes the Ant move right. We will illustrate the approach to probe the correlation in the following subsection.

Identifying Correlation Coefficient. We compute the correlation coefficient through *spectral phase discrepancy*. we first find the predominant frequency component for each signal:

$$\omega^* = \underset{\omega}{\text{argmax}} \sum_d \|\text{SG}(d, \omega|\mathbf{m}^j) - \text{SG}(d, \omega|\mathbf{s}^j)\|_2. \tag{6}$$

At each window of STFT, we can compute the phase discrepancy of the transforms of $\mathbf{m}^j$ and $\mathbf{s}^j$ at the predominant frequency ω^*. The average phase discrepancy at all temporal windows is computed across N collected episodes:

$$p^j = \frac{1}{D \cdot N} \sum_d \sum_N [\Phi(\text{STFT}(d, \omega^*|\mathbf{m}^j)) - \Phi(\text{STFT}(d, \omega^*|\mathbf{s}^j))], \tag{7}$$

where D is the number of temporal windows and Φ is the function retrieving the phase. We then normalize p^j to get correlation coefficient ρ between m^j and s^j as $\rho^j = 1 - |2p^j|/\pi$. ρ^j tells us the most important thing that whether the s^j will be amplified or reduced by overwriting the output of m^j with a *positive* or *negative* value. After that, v^j is finally determined by

$$v^j = c^j \rho^j, \tag{8}$$

where c^j is a magnitude coefficient determining the scale of v^j and thus $\dot{s}^j$. We usually determine the c^j by observing the performance after trying evoking the desired behavior with different c^j, while it can also be determined by an external PID controller, which takes the error $\hat{s} - s^j_t$ between target state $\hat{s}$ and current state s^j_t as input and outputs c^j automatically.

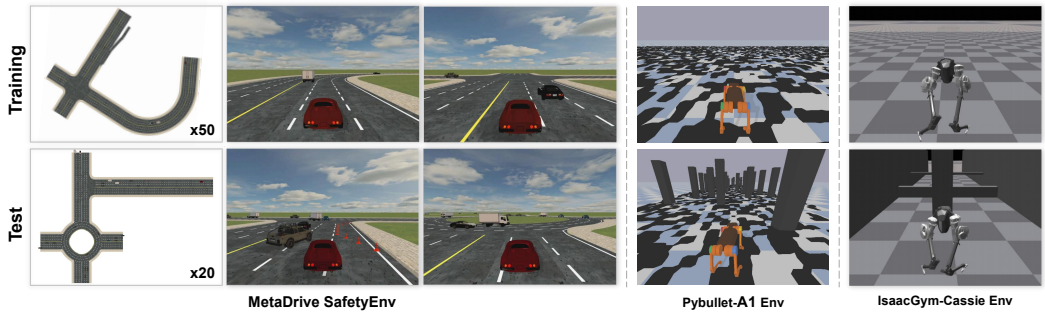


Figure 4: Examples of training and test scenes used in the generalization experiment. In MetaDrive SafetyEnv, the held-out 20 test scenes are more complex compared to 50 training scenarios. For quadruped robot locomotion task, the robot is trained to traverse open but bumpy terrains and tested on the terrain full of obstacles where obstacle avoidance ability is needed to reach the destination. In the Cassie environment, the agent is trained to move forward. However, crouching, jumping and sidestepping are required to reach the destination in the test environment.

4 Experiments

We first apply *Policy Dissection* to an autonomous driving task to study the learning dynamics in Sec. 4.2. We demonstrate that the emergent pivotal *motor primitives* are highly correlated with the success of policy learning. With the trained driving policy, we show a novel application of the shared autonomy in Sec. 4.3: improving the test-time generalization, where Human-AI shared control can be an alternative to overcoming the generalization gap between training and unseen test-time environment. The experiment demonstrates that our method finds not only the correlation but causality between unit output and kinematics. In Sec. 4.4, we conduct experiments on quadrupedal robot and demonstrate that a quadrupedal robot trained to traverse complex terrain with only proprioceptive state can transfer to accomplish obstacle avoidance task with a bit human effort, even if it does not have any information of the surrounding. For qualitatively evaluting our method, we provide a demonstration in Sec. 4.5, where a bipedal Cassie robot trained for moving forward can solve a challenging parkour environment with the help of human subject. In Sec. 4.6, we investigate the coarseness of the goal-conditioned control enabled by *Policy Dissection* through a command tracking experiment , and compare it with the state-of-the-art goal-conditioned controller in IsaacGym [42, 57].

4.1 Experimental Setting

The experiments conducted on Cassie and ANYmal in IsaacGym [57], Ant and Walker in MuJoCo [69] and the BipedalWalker [9] follow the general setting. Other experiments conducted on MetaDrive and Pybullet-A1 environments are introduced as follows:

MetaDrive. We train agents to accomplish autonomous driving task in the safety environment of MetaDrive [39]. In this task, the goal is to maneuver the target vehicle to destination with low-level continuous control actions. An episode terminates only when the vehicle arrives at the destination, drives out of road, or collides with the sidewalk. The observation of autonomous vehicle consists of map, navigation, lidar, and state information such as speed, heading and side distance. More training environment details can be found in appendix.

Pybullet-A1. The legged robots locomotion experiments are conducted on the Pybullet-Unitree-A1 [16] robot. We follow the same environment setting used by Locotransformer robot [76], including terrain shape, obstacle distributions, sensors, reward definition, and termination condition. Two agents are trained. One "insensible" agent with only proprioceptive state is trained to walk on uneven terrain, while the other agent with Locotransformer is directly trained to avoid obstacles. The shared control system will be built on the "insensible" agent, where human subjects help the agent sidestep obstacles.

Training Set and Test Set. As shown in Fig. 4, for both autonomous driving and robot locomotion tasks, we prepare training set and held-out test set. For driving task, the test set is used to benchmark the test-time generalizability of human-AI shared control system and AI-only control system. We train autonomous driving policies in 50 different environments where the traffic condition is mild

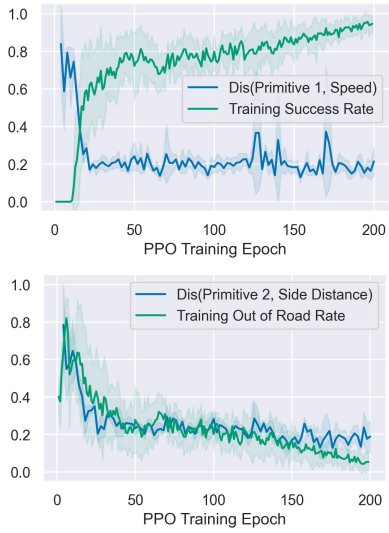


Figure 5: Learning Dynamics

Method	Human Involvement	Episodic Return	Episodic Cost	Success Rate
Human	405.57 $\pm$ 94.79	357.21 $\pm$ 38.3	0.47 $\pm$ 0.24	0.95 $\pm$ 0.03
SAC$^{w/oH}$	-	263.85 $\pm$ 34.74	5.75 $\pm$ 0.25	0.75 $\pm$ 0.12
SAC$^{H^n}$	26.52$\pm$ 1.58	342.0$\pm$ 6.25	0.42$\pm$ 0.32	0.9$\pm$ 0.0
SAC$^{H^*}$	2.85 $\pm$ 0.22	353.19 $\pm$ 13.62	0.80 $\pm$ 0.14	0.95 $\pm$ 0.03
PPO$^{w/oH}$	-	234.81 $\pm$ 23.45	3.37 $\pm$ 2.01	0.75 $\pm$ 0.03
PPO$^{H^n}$	30.58$\pm$ 0.18	352.99$\pm$ 6.59	0.32$\pm$ 0.12	0.92$\pm$ 0.02
PPO$^{H^*}$	**2.55** $\pm$ 0.35	**359.67** $\pm$ 11.20	**0.05** $\pm$ 0.08	**0.95** $\pm$ 0.02
Relu-PPO$^{H^*}$	4.65 $\pm$ 0.3	352.98 $\pm$ 7.35	0.35 $\pm$ 0.1	0.88 $\pm$ 0.07
Deep-PPO$^{H^*}$	4.88 $\pm$ 1.17	334.09 $\pm$ 2.91	0.48 $\pm$ 0.02	0.9 $\pm$ 0.02

Table 1: Improvements on the performance and the safety on test environments with human-AI shared control. H indicates "human". H^n and H^* indicates two takeover methods. H^n requires human to provide steering and throttle in the whole takeover period, while H^* only asks to activate *motor primitives* at the start of takeover. We also do ablation study on the activation function and number of hidden layers of MLP. These results show our method is general and robust to the policy architecture and RL methods.

(averagely 2 cars per map) and all traffic vehicles and the target vehicle can drive smoothly towards their destinations. No obstacles are in these environments. In testing time, the trained agents will be evaluated in the other 20 unseen maps with *higher* traffic density (averagely 6 cars per map). Besides, obstacles like traffic cones and breakdown vehicles will scatter randomly on the road. In this case, the target vehicle needs to be maneuvered delicately.We compare the performance and safety of the trained policy in two scenarios: runs in the test set solely, and runs with the human-AI shared control system built from *Policy Dissection*.

The primal task for the legged robot is to move toward one direction as fast as possible. We design the test environment to be a forest-like environment where tree obstacles are randomly scattered. Therefore, though the overall goal is still to move forward, the agent needs extra effort to sidestep obstacles in the test environment.

Evaluation Metrics. For driving task, we evaluate agents on all the 20 test environments, and report the ratio of episodes where the agent arrives at the destination as the *success rate*. Also, *Out of Road rate* can be calculated if the termination is caused by driving out of road. For benchmarking the safety of agents, the collision to vehicles, obstacles, sidewalk and buildings raises a cost $+1$. In addition, human attention cost is a huge concern of human-AI shared control system. We quantify this using *human involvement*, which is measured by the number of environment steps when the human subject presses keys for activating units, or controls throttle, brake, and steering wheel. We average the sum of reward, cost and human involvement on 20 environments as *episodic return*, *episodic cost* and *human involvement*. The above evaluation process will be repeated 5 times for each agent, and we report the average value and std for all metrics.

For the locomotion task, we use *moving distance* to measure the performance besides reward, since it directly reflects the goal. Moving distance is the displacement of robot on specific direction, i.e. x-axis. The evaluation is also repeated for 5 times on the exclusive test environment.

Human Interface. We use keyboard as interface in all human-AI shared control experiments. When a specific key is pressed for evoking a behavuir, related *motor primitives* will be activated and output v^j. The behavior termination can be triggered by ways: 1. Press another key and change to another behavior 2. The length of activation period $T = t_1 - t_0$ reaches a predefined length. Once a behavior is terminated, the related *motor primitives'* output will not be overwritten. In addition, there is one key mapping to the default behavior that no *motor primitives* are activated.

All experiments without human are repeated 5 times with different random seeds. $N = 5$ episodes are used for applying *Policy Dissection*. For experiments involving human subject, we repeat each experiment with 5 human subjects. For driving task, all human subjects have driving license. Human subject can pause the experiments if any discomfort happens. No human subjects were harmed in the experiments. Human subjects have signed the experiment agreement and get compensation.

4.2 Policy Dissection for Understanding Learning Dynamics

In this section, we verify the correlation between *motor primitives* and relevant kinematic attributes. We first train PPO [61] agent on the MetaDrive's training environments and then conduct *Policy Dissection* on the trained policy and find two *motor primitives* in which **Primitive 1** is associated with **Speed** and **Primitive 2** associated with **Side Distance**. We then plot the evolution of the frequency discrepancy between the identified *motor primitives* and the kinematics alongside with the macroscopic measure of the learning progress in Fig. 5. Note that the frequency discrepancies $\mathbf{d} = [d_1, ..., d_T]$ is normalized to $[0, 1]$ by $\tilde{\mathbf{d}} = \frac{\mathbf{d} - \min(\mathbf{d})}{\max(\mathbf{d}) - \min(\mathbf{d})}$.

The upper plot in Fig. 5 shows Primitive 1 gradually emerges to control speed, indicated by the decreasing discrepancy. With strong negative correlation, the success rate rapidly increases when the discrepancy of speed primitive also rapidly decreases between epoch 20-30. Similarly, in the lower plot in Fig. 5, the decreasing discrepancy between Primitive 2 and side distance implies that the internal unit of the agent is gradually specialized to control side distance. There exists a strong correlation between the discrepancy and the out-of-road rate, which can only be reduced by improving the ability to keep side distance. We thus conclude that the pivotal primitives emerge during training.

4.3 Human-AI Shared Control for Test-time Generalization

Using the *motor primitives* identified above, we can determine the activation value of *motor primitive* for steering associated kinematics according to Eq. 8. We apply the same process to dissect the policy trained from SAC [23] with the same network structure and build a *stimulation-evoked map* for shared control. We then invite human subject to collaborate with the SAC and PPO agents on the test environments with different takeover methods. As shown in Tab. 1, the average human involvement, episodic cost, and success rate in the test environments are reported across policies with or without human involvement. The human involvement greatly improves the performance and safety when deploying the policy in unseen environments where delicate maneuvers such as side-passing and yielding are evoked by human, if necessary, to help finish the task.

Different from the original PPO policy which is a MLP with 2 hidden layers, 256 units per layer and "tanh" activation function, we also repeat the same experiment for Deep-PPO agent which has 6 hidden layers, and Relu-PPO which has "Relu" as activation function. The results reported in 1 show our method is invariant to the network architectures and RL algorithms.

4.4 Human-AI Shared Control for Task Transfer

In addition to the improved test-time generalization, the *stimulation-evoked map* enables RL agent to accomplish new tasks with the help of human. Recent quadrupedal robots trained by RL exhibit promising ability to walk on bumpy terrain via end-to-end control [36]. However, the agent only takes proprioceptive states as input without the observation of surroundings, and thus it can not avoid obstacles by its own. As shown in Fig. 4, we train the agent only to walk on uneven terrain but test it in a new environment full of obstacles. By applying our method, *motor primitives* related to yaw rate and speed control can be found and serve as the interface for human to steer the "insensible" quadrupedal robot to avoid obstacles. This forms the "State$^{w/oH}$" method. We also include "State+Vision" method [76], which trains a visuomotor policy for legged robot directly in the test environment, enabling it to sidestep obstacles. As shown in Table 2, with human involvement, *Policy Dissection* successfully transforms the "insensible" policy that is not fit to this task and greatly improves the performance, achieving comparable performance to the agent specially designed for this task.

Method	Reward	Moving Distance
State$^{w/oH}$	18.23 $\pm$ 10.55	6.55 $\pm$ 0.04
State$^{H^*}$	413.93 $\pm$ 93.33	16.94 $\pm$ 3.08
State+Vision	445.47$\pm$ 190.85	18.64 $\pm$ 6.45

Table 2: Test on Obstacle Avoidance Task.

4.5 Qualitative Result: Parkour Demo

As shown in the Fig. 4, the training objective of the bipedal Cassie robot is to move forward. After training, *Policy Dissection* can discover several skills in the learned representation, as shown in Fig. 6. Furthermore, these skills can be triggered and controlled by human subject for solving the challenging parkour environment. See the demo video for detail.

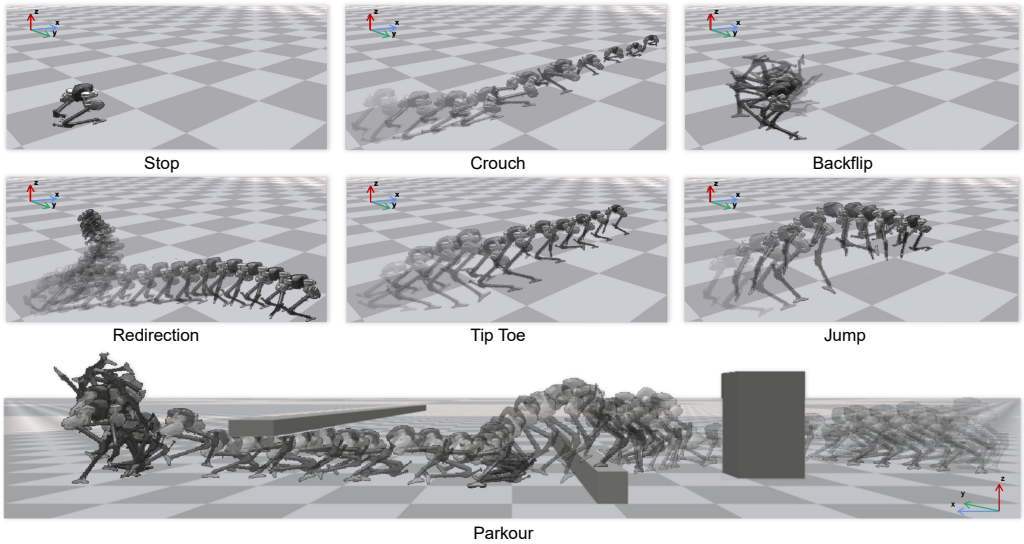


Figure 6: *Policy Dissection* can discover several skills such as crouching, jumping, backflipping in the learned representation of the bipedal Cassie robot, which is originally trained to move forward. These skills can be combined by human subject and overcome a challenging parkour environment.

4.6 Experiment on the Control Precision

We conduct an experiment to evaluate the control precision enabled by our method. We train a goal-conditioned policy for ANYmal-C robot in IsaacGym [57]. The trained robot taking raw rate as input can track a target yaw with a PID controller, and thus serves as the baseline. For a fair comparison, we directly dissect this policy and identify the yaw primitive. Likewise, a PID controller that takes the yaw error as input is used to automatically determine the output v^j for yaw primitive, enabling tracking target yaw. Note that for the primitive activation control the explicit heading command in input is set to 0. Heading tracking results in Fig. 7 shows that the control achieved by primitive activation has comparable precision to the goal-conditioned control (see demo video for more detail).

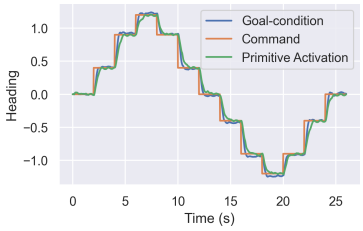


Figure 7: Heading tracking error from the goal-condition control and the primitive activation control.

5 Conclusion and Discussion

Inspired by the neuroscience approach to study the motor cortex in primates, we present a simple yet effective method called *Policy Dissection* to align internal neural representations and kinematic attributes of a pretrained neural policy. The resulting *stimulation-evoked map* allows human to steer the agent and evoke its certain behaviors. *Policy Dissection* enables human-AI shared control systems on top of the models trained from standard RL algorithms, without any re-training or modification. We quantitatively evaluate the human-AI shared control system in autonomous driving and locomotion tasks, and the result suggests the substantial improvement of the test-time generalization for task transfer. We further provide interactive demonstrations in various environments to show the generality of our method and its application for human-AI collaboration.

Limitations and societal impact. One limitation of *Policy Dissection* is that we only evaluate this method in continuous control tasks with the state vector as input. A wider range of tasks and neural network architecture such as discrete control tasks and CNNs can be adopted in future research. One potential negative social impact is that the safety issue remains to be solved when deploying the shared control system in real world application. As shown in the autonomous driving experiment, our method derived from empirical study has no theoretical guarantee on the safety of human subject. The unsuccessful collaboration or conflict between human and AI may lead to potential risks.

References

[1] O. Amir, E. Kamar, A. Kolobov, and B. Grosz. Interactive teaching strategies for agent training. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[2] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[3] A. Atrey, K. Clary, and D. Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743*, 2019.

[4] A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, A. Pritzel, M. J. Chadwick, T. Degris, J. Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557 (7705):429–433, 2018.

[5] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv preprint arXiv:1811.10597*, 2018.

[6] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[8] J. Bragg and E. Brunskill. Fake it till you make it: Learning-compatible performance support. In *Uncertainty in Artificial Intelligence*, pages 915–924. PMLR, 2020.

[9] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[10] Z. Cao, E. Biyik, W. Wang, A. Raventos, A. Gaidon, G. Rosman, and D. Sadigh. Reinforcement learning based control of imitative policies for near-accident driving. In *Robotics: Science and Systems*, 2020.

[11] S. Carter and M. Nielsen. Using artificial intelligence to augment human intelligence. *Distill*, 2(12):e9, 2017.

[12] X. Chen, Z. Wang, Y. Fan, B. Jin, P. Mardziel, C. Joe-Wong, and A. Datta. Reconstructing actions to explain deep reinforcement learning. *arXiv preprint arXiv:2009.08507*, 2020.

[13] J. Cho, D.-H. Lee, and Y.-G. Yoon. Inducing functions through reinforcement learning without task specification. *arXiv preprint arXiv:2111.11647*, 2021.

[14] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[15] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[16] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. *PyBullet*, 2016.

[17] C. J. Cueva and X.-X. Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.

[18] A. K. Dhawale, S. B. Wolff, R. Ko, and B. P. Ölveczky. The basal ganglia can control learned motor sequences independently of motor cortex. *BioRxiv*, page 827261, 2019.

[19] L. Fridman. Human-centered autonomous vehicle systems: Principles of effective shared autonomy. *arXiv preprint arXiv:1810.01835*, 2018.

[20] M. Graziano. *The intelligent movement machine: An ethological perspective on the primate motor system*. Oxford University Press, 2008.

[21] M. S. Graziano, C. S. Taylor, and T. Moore. Complex movements evoked by microstimulation of precentral cortex. *Neuron*, 34(5):841–851, 2002.

[22] S. Guo, R. Zhang, B. Liu, Y. Zhu, D. Ballard, M. Hayhoe, and P. Stone. Machine versus human attention in deep reinforcement learning tasks. *Advances in Neural Information Processing Systems*, 34, 2021.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870, 2018.

[24] R. H. Hahnloser, A. A. Kozhevnikov, and M. S. Fee. An ultra-sparse code underliesthe generation of neural sequences in a songbird. *Nature*, 419(6902):65–70, 2002.

[25] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, et al. Urban driving with conditional imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 251–257. IEEE, 2020.

[26] A. A. Ismail, M. Gunady, H. C. Bravo, and S. Feizi. Benchmarking deep learning interpretability in time series predictions. *arXiv preprint arXiv:2010.13924*, 2020.

[27] H. J. Jeon, D. P. Losey, and D. Sadigh. Shared autonomy with learned latent actions. *arXiv preprint arXiv:2005.03210*, 2020.

[28] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083. IEEE, 2019.

[29] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[30] J. Kim and J. Canny. Interpretable learning for self-driving cars by visualizing causal attention. In *Proceedings of the IEEE international conference on computer vision*, pages 2942–2950, 2017.

[31] P. Knott, M. Carroll, S. Devlin, K. Ciosek, K. Hofmann, A. D. Dragan, and R. Shah. Evaluating the robustness of collaborative agents. *arXiv preprint arXiv:2101.05507*, 2021.

[32] J. Konczak. On the notion of motor primitives in humans and robots. *Proceedings of the Fifth International Workshop on Epigenetic Robotics*, 2005.

[33] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[34] N. Kriegeskorte and J. Diedrichsen. Peeling the onion of brain representations. *Annual review of neuroscience*, 42:407–432, 2019.

[35] N. Kriegeskorte, M. Mur, and P. Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2, 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008.

[36] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[37] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[38] A. Levy, R. Platt, and K. Saenko. Hierarchical reinforcement learning with hindsight. *arXiv preprint arXiv:1805.08180*, 2018.

[39] Q. Li, Z. Peng, Z. Xue, Q. Zhang, and B. Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *arXiv preprint arXiv:2109.12674*, 2021.

[40] X.-H. Li, C. C. Cao, Y. Shi, W. Bai, H. Gao, L. Qiu, C. Wang, Y. Gao, S. Zhang, X. Xue, et al. A survey of data-driven and knowledge-aware explainable ai. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[41] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[42] 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.

[43] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese. Human-in-the-loop imitation learning using remote teleoperation. *arXiv preprint arXiv:2012.06733*, 2020.

[44] K. Mason and S. Grijalva. A review of reinforcement learning for autonomous building energy management. *Computers & Electrical Engineering*, 78:300–312, 2019.

[45] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018.

[46] J. Merel, D. Aldarondo, J. Marshall, Y. Tassa, G. Wayne, and B. Ölveczky. Deep neuroethology of a virtual rodent. *arXiv preprint arXiv:1911.09451*, 2019.

[47] O. Nachum, S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018.

[48] H. Nili, C. Wingfield, A. Walther, L. Su, W. Marslen-Wilson, and N. Kriegeskorte. A toolbox for representational similarity analysis. *PLoS computational biology*, 10(4):e1003553, 2014.

[49] D. Omeiza, H. Webb, M. Jirotka, and L. Kunze. Explanations in autonomous driving: A survey. *arXiv preprint arXiv:2103.05154*, 2021.

[50] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

[51] Z. Peng, Q. Li, C. Liu, and B. Zhou. Safe driving via expert guided policy optimization. In *5th Annual Conference on Robot Learning*, 2021.

[52] J. Peters and S. Schaal. Learning motor primitives with reinforcement learning. *11th Joint Symposium on Neural Computation, University of Southern California*, 2004.

[53] N. Puri, S. Verma, P. Gupta, D. Kayastha, S. Deshmukh, B. Krishnamurthy, and S. Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. *arXiv preprint arXiv:1912.12191*, 2019.

[54] S. Reddy, A. D. Dragan, and S. Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv:1802.01744*, 2018.

[55] S. Reddy, A. Dragan, S. Levine, S. Legg, and J. Leike. Learning human objectives by evaluating hypothetical behavior. In *International Conference on Machine Learning*, pages 8020–8029. PMLR, 2020.

[56] F. Richter, R. K. Orosco, and M. C. Yip. Open-sourced reinforcement learning environments for surgical robotics. *arXiv preprint arXiv:1903.02090*, 2019.

[57] 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*, pages 91–100. PMLR, 2022.

[58] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans. Trial without error: Towards safe reinforcement learning via human intervention. *arXiv preprint arXiv:1707.05173*, 2017.

[59] C. Schaff and M. R. Walter. Residual policy learning for shared autonomy. *arXiv preprint arXiv:2004.05097*, 2020.

[60] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International conference on machine learning*, pages 1312–1320. PMLR, 2015.

[61] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[62] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[63] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. URL http://arxiv.org/abs/1610.02391.

[64] Y. Shen and B. Zhou. Closed-form factorization of latent semantics in gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1532–1540, 2021.

[65] J. Spencer, S. Choudhury, M. Barnes, M. Schmittle, M. Chiang, P. Ramadge, and S. Srinivasa. Learning from interventions. In *Robotics: Science and Systems (RSS)*, 2020.

[66] F. P. Such, V. Madhavan, R. Liu, R. Wang, P. S. Castro, Y. Li, J. Zhi, L. Schubert, M. G. Bellemare, J. Clune, et al. An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents. *arXiv preprint arXiv:1812.07069*, 2018.

[67] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[68] P. S. Thomas and A. G. Barto. Motor primitive discovery. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–8. IEEE, 2012.

[69] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[70] E. Ugur, E. Şahin, and E. Oztop. Self-discovery of motor primitives and learning grasp affordances. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3260–3267. IEEE, 2012.

[71] G. Warnell, N. Waytowich, V. Lawhern, and P. Stone. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[72] J. Wu, Z. Huang, C. Huang, Z. Hu, P. Hang, Y. Xing, and C. Lv. Human-in-the-loop deep reinforcement learning with application to autonomous driving. *arXiv preprint arXiv:2104.07246*, 2021.

[73] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246. IEEE, 2018.

[74] Y. Xu, Y. Shen, J. Zhu, C. Yang, and B. Zhou. Generative hierarchical features from synthesizing images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4432–4442, 2021.

[75] C. Yang, Y. Shen, and B. Zhou. Semantic hierarchy emerges in deep generative representations for scene synthesis. *International Journal of Computer Vision*, 129(5):1451–1466, 2021.

[76] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. *arXiv preprint arXiv:2107.03996*, 2021.

[77] N. Yoshida, T. Daikoku, Y. Nagai, and Y. Kuniyoshi. Embodiment perspective of reward definition for behavioural homeostasis. In *Deep RL Workshop NeurIPS 2021*, 2021.

[78] É. Zablocki, H. Ben-Younes, P. Pérez, and M. Cord. Explainability of vision-based autonomous driving systems: Review and challenges. *arXiv preprint arXiv:2101.05307*, 2021.

[79] Q.-s. Zhang and S.-c. Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[80] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone. Leveraging human guidance for deep reinforcement learning tasks. *arXiv preprint arXiv:1909.09906*, 2019.

[81] R. Zhang, F. Torabi, G. Warnell, and P. Stone. Recent advances in leveraging human guidance for sequential decision-making tasks. *Autonomous Agents and Multi-Agent Systems*, 35(2):1–39, 2021.

[82] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[83] B. Zhou, D. Bau, A. Oliva, and A. Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2131–2145, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5, Limitations
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Hyperparameters are in appendix, and experimental settings are in Section 4
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4. Experimental Setting
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes] See Section 4. Experimental Setting
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] See Section 4. Experimental Setting

Appendix

A Code and Demo Video.

In the project webpage: https://metadriverse.github.io/policydissect, we provide **demonstrative videos** showing the *Policy Dissection* for human-AI shared control and the source code including all trained controllers.

B Workflow of Policy Dissection

Algorithm 1: The workflow of identifying *motor primitives*

Input : MLP policy π with L layers, I units per layer, environment $\mathbf{env}$, episodes to collect N.
Output : *motor primitives* m^j, correlation coefficient v^j related to s^j, where $j = 1, ..., J$.

for $n \leftarrow 0$ **to** N **do**
  Executing policy π in $\mathbf{env}$, recording $\{\mathbf{z}^{l,i} = [z_1^{l,i}, z_2^{l,i}, ...]\}^{I \cdot L}$ and $\{\mathbf{s}^j = [s_1^j, s_2^j, ...]\}^J$.
for $j \leftarrow 0$ **to** J **do**
  for $l \leftarrow 0$ **to** L **do**
    for $i \leftarrow 0$ **to** I **do**
      Calculate frequency discrepancy $\text{Dis}(\mathbf{z}^{l,i}, \mathbf{s}^j)$ following Eq. 3:
  Determine *motor primitive* m^j for kinematic attribute s^j according to Eq. 4:
  Identify output v^j for m^j according to Eq. 8:

After identifying the motion generation block, *motor primitives*, we can pick one or more of them to activate simultaneously and evoke a certain behavior. Therefore, a group of units are aligned with a certain behavior and *stimulation-evoked map* can be built.

C Human Subject's Comments on Shared Control

The performance of shared controls system is largely determined by the effective coordination between human subject and the AI. Some of our experiment participants report that collaboration with quadrupedal AI is more difficult than with self-driving AI. The possible reason is that human subjects are more familiar with driving vehicles than controlling legged robots. For the shared driving task, human subjects have better global planning abilities about which lane to choose and when to stop. However, for controlling the legged robots on bumpy terrain and avoiding obstacles, some of them didn't consider how the complex terrain would influence the pose of legged robots and thus had a hard time interacting with the quadrupedal AI. For example, when the robot is moving down from a small slope and is pitching forward, stopping command should be issued cautiously. Otherwise, the agent may roll or flip forward. Therefore, it would be better if human subjects have prior knowledge about the task, so that they can collaborate better with the AI trained for this task. How to implement effective human-AI teaming in general is an important but less explored direction.

D Summary of Qualitative Results

Besides the results shown in the main paper, we provide more qualitative results of human-AI shared control on policies trained in MetaDrive, Pybullet-A1, Ant, Walker and BipedalWalker environments.

As shown in Fig. 8, in the top-down view we visualize the trajectories of the PPO agent with or without human shared control on the test environment of driving task. Since the PPO agent is trained in the mild traffic density without obstacles, it is incompatible to brake and side-pass. In contrast, with human in the control loop, the lane changing and braking can be deliberately evoked by human, helping the agent solve the near-accidental cases.

We provide a summary on how to identify and use *motor primitives* to evoke behaviors for the trained agents in different environments.

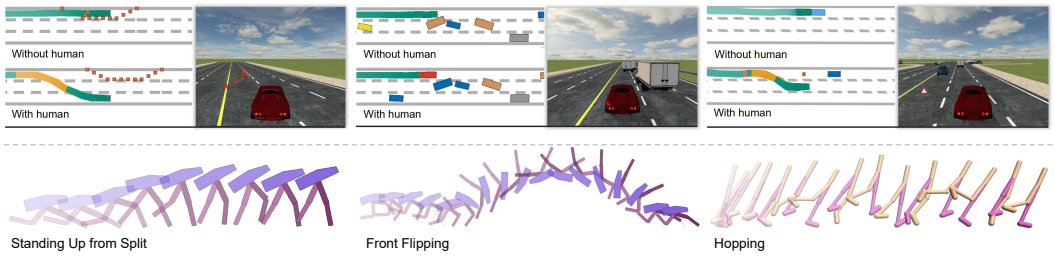


Figure 8: Qualitative demonstration of *Policy Dissection*. First row plots the driving results of the trained PPO agent with or without human-AI shared control in three driving cases. Each case plots the trajectories with human involvement, where different colors represent trajectory segments resulted from different *motor primitives*. Red and orange denote "braking" and "lane changing" respectively. The second row plots more examples of evoking behaviors in different locomotion environments.

MetaDrive: Brake. By activating the *motor primitive* positively related to speed with a negative value, the agent can gradually decrease the speed and still stay in the same lane. The brake can be evoked to avoid collision to obstacles and cut-in vehicles which the agent has never seen in the training phase.

MetaDrive: Lane Changing. Since the agent can observe the distance to left and right side, identifying motor primitives related to sidewalk and yellow solid line can make the distance to side controllable. We use "tanh" as activation function which is symmetric. Therefore identifying only one *motor primitive* related to one side is enough to evoke lane changing behavior, since, for example, increasing the distance to left side equals decreasing the distance to right side. Lane changing is useful for sidestepping obstacles or moving to another lane with lower traffic density.

Pybullet-A1: Turning Left/Right. As shown in Fig. 9, the moving direction of legged robot can be controlled by activating the neurons related to yaw or yaw rate. Similarly, negative and positive stimulation makes the agent turn towards opposite direction.

Pybullet-A1: Stopping. We can also evoke deceleration or stopping behaviors on the legged robot by controlling the activation of speed-related units. Since a huge deceleration may make the agent roll forward, we also add a pitch control to suppress the possible front flipping.

Gym-Ant: Stopping.. The ant will stop if *motor primitive* associated with x-axis speed is activated.

Gym-Ant: Moving Up/Down. The ant can move up/down if we activate two primitives respectively related to: 1) Speed of y-axis and 2) heading direction. Similar to evoking legged robot turning behavior, opposite activation values make the agent move in opposite direction.

Gym-Ant: Spinning. The ant can spin if we activate unis related to yaw rate.

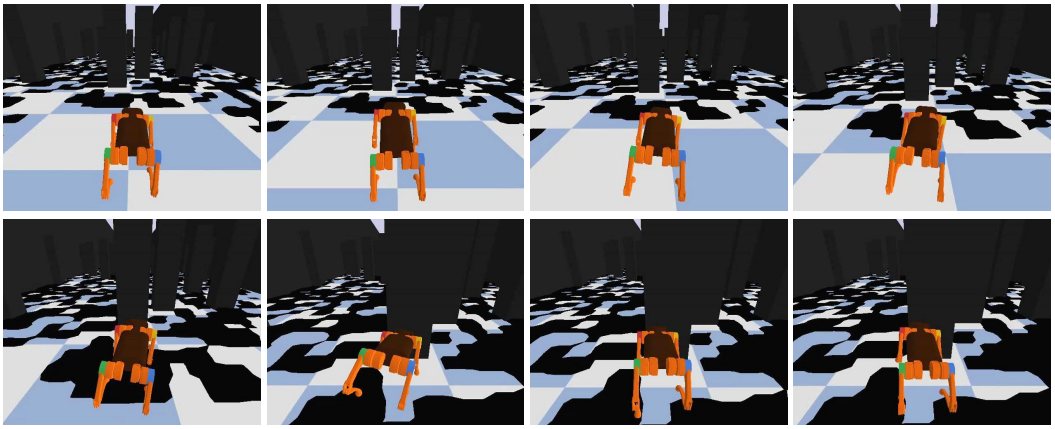
Gym-Walker: Deceleration. The agent trained in Walker stops because we stimulate the *motor primitives* positively associated with velocity with a negative stimulation. As a result, the low-level action sequence for deceleration is executed.

Gym-Walker: Hopping. We manage to find the *motor primitives* associated with torque force. We then disable the movement of the knee in red leg by stimulating the *motor primitive* with negative value. As a result, the agent hops with only the yellow leg.

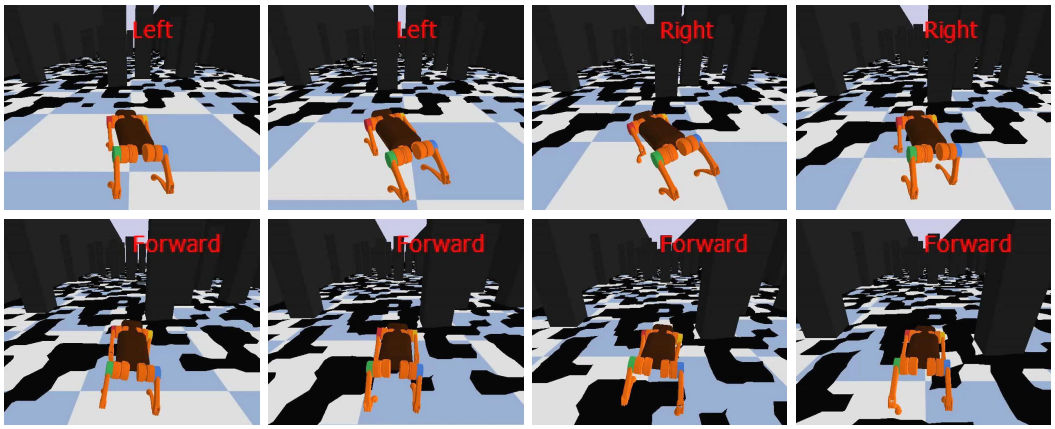
Gym-BidepalWalker: Jumping. For the agent trained in BipedalWalker, jumping is achieved by stimulating *motor primitives* related to V_z, where Z-axis is upward.

Gym-BidepalWalker: Front Flipping. We train the BidepdalWalker walking with larger torque, and thus it can jump higher if we activate units associated with V_z. BipedalWalker can conduct front flipping, which is the combination of jumping and pitching. This is resulted from combining another *motor primitive* related to pitch into control. Therefore, the walker jumps with increasing angular velocity, performing front flipping.

Gym-BidepalWalker: Standing Up from Split. In addition, a common failure mode of agent trained in BipedalWalker is that the agent performs split and can not stand up when both legs touch



(a) Insensible Agent Walking in Test Environment **Without** Human



(b) Insensible Agent Walking in Test Environment **With** Human

Figure 9: Qualitative demonstration of the insensible agents when deployed in test environment with or without human. Facilitated by human, the legged robots can turn left to avoid the collision to the front obstacle, even if it is insensible and trained to move forward as fast as possible.

the ground. We identify the *motor primitives* associated with all the motor torques. Stimulating these *motor primitives*, the agent manages to stand up and continue moving forward.

IsaacGym-Cassie: Crouching and Tiptoe. The crouching and tiptoe behaviors are related to the z-height of the main robot body, so that we can identify z-axis related units, and activate them to change the z-height.

IsaacGym-Cassie: Backflip. This behavior can be described by increasing 1. height 2. pitch and 3. knee force. Therefore, we activate three corresponding *motor primitives* selected by Eq. 4 with output calculated by Eq. 8 to make the robot back flip. Note that we also manually determine the termination time t_1 for this action, preventing it from continuously back flipping

IsaacGym-Cassie: Jumping. The jumping is represented by increasing 1. knee torque force 2. forward speed/moving distance. Therefore, activating corresponding *motor primitives* evoke jumping. Similar as the back-flip, the activation period $T = t_1 - t_0$ is predetermined, so that it only jump once after pressing the key for jumping.

IsaacGym-Cassie: Redirection. The redirection can be achieved in the same way as Gym-Ant, activating y-axis movement related units or heading related units. The forward movement can be evoked by activating x-axis related *motor primitives*.

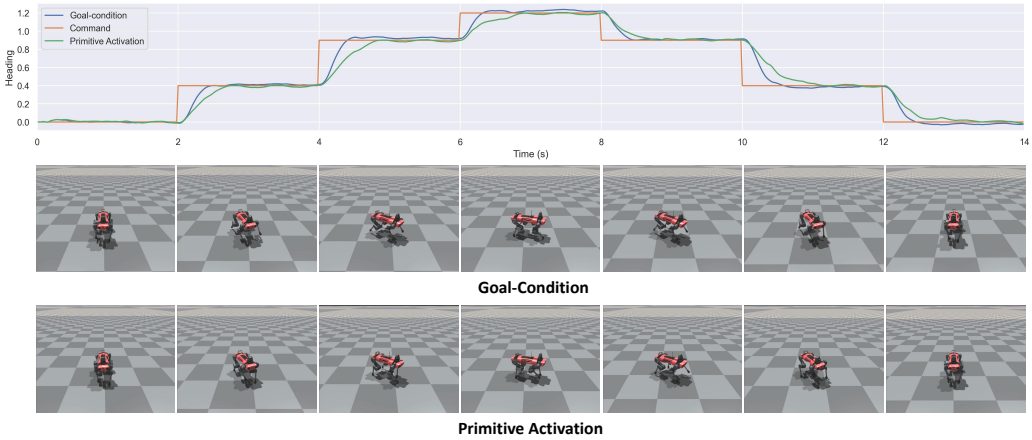


Figure 10: The visualization result of the comparison experiment. It suggests that the goal conditioned control enabled by our method is comparable to the state-of-the-art goal conditioned controller on ANYmal-C robot in IsaacGym simulator.

E Details and Visualization of Control Precision Experiment

We provide a visualization results of the control precision experiments, as shown in Fig. 10. It is obvious that both both goal-conditioned control methods can drive ANYmal-C to track the yaw command smoothly and stably. The main difference is that the goal-conditioned control enabled by *Policy Dissection* has a slightly longer response time when changing command, compared to explicit goal-conditioned control. The demo video is availabe here .

F Environment Details

We provide more environment details for MetaDrive and Pybullet-A1 such as the observations, the design of reward functions, and the termination conditions.

F.1 MetaDrive

In the driving task, the objective of RL agents is to steer the target vehicles with low-level continuous control actions, namely acceleration, brake, and steering.

Observation. The observation of RL agents is as follows:

- A 240-dimensional vector denoting the Lidar-like point clouds with $50m$ maximum detecting distance centering at the target vehicle. Each entry is in $[0, 1]$ with Gaussian noise and represents the relative distance of the nearest obstacle in the specified direction.
- A vector containing the data that summarizes the target vehicle's state such as the steering, heading, velocity, and relative distance to the left and right boundaries.
- The navigation information that guides the target vehicle toward the destination. We sparsely spread a set of checkpoints, 50m apart on average, in the route and use the relative positions toward future checkpoints as additional observation to the target vehicle.

Reward and Cost Scheme. The reward function is composed of four parts as follows:

$$R = c_1 R_{disp} + c_2 R_{speed} + R_{term}. \tag{9}$$

The *displacement reward* $R_{disp} = d_t - d_{t-1}$, wherein the d_t and d_{t-1} denotes the longitudinal movement of the target vehicle in Frenet coordinates of current lane between two consecutive time steps, provides dense reward to encourage agent to move forward. The *speed reward* $R_{speed} = v_t/v_{max}$ incentives agent to drive fast. v_t and v_{max} denote the current velocity and the maximum velocity (80 km/h), respectively. We also define a sparse *terminal reward* R_{term}, which is non-zero

only at the last time step. At that step, we set $R_{disp} = R_{speed} = 0$ and assign R_{term} according to the terminal state. R_{term} is set to $+10$ if the vehicle reaches the destination, -5 for crashing others or violating the traffic rule. We set $c_1 = 1$ and $c_2 = 0.1$. For measuring the safety, collision to vehicles, obstacles, sidewalk raises a cost $+1$ at each time step. The sum of cost generated in one episode is episode cost, a metric like episode reward, but reflecting safety instead.

Termination Conditions and Evaluation Metrics. Since we attempt to benchmark the safety of shared-control system, collision to vehicles and obstacles will not terminate the episode. The episode will be terminated only when: 1) the agent drive out of the drivable area and 2) the agent arrives the destination. For each trained agent, we evaluate it in 20 held-out test environment and define the ratio of episodes where the agent arrives at the destination as the *success rate*. The definition is the same for *out of road*. Also, the average episode reward and cost on 20 test environment produce two metrics: *Episodic Reward* and *Episodic Cost*. Since each agent are trained across 5 random seeds, this evaluation process will be executed for 5 agent which has same training setting but different random seeds. We report the average and std on the 4 metrics mentioned above.

F.2 Pybullet-A1

In the quadrupedal locomotion task, the RL training objective is to move forward on bumpy terrain as fast as possible. We use two environments: the one without obstacles as training environment, and the other one with obstacles as test environment. The environment construction, reward definition, tasks are the same as [76]. Its environment has open-source code at: https://github.com/Mehooz/vision4leg. In this work, we only slightly modify the observation as follows:

- IMU recording Yaw, Yaw rate, Pitch and Roll
- Angle of 12 joints
- Torque applied to 12 joints

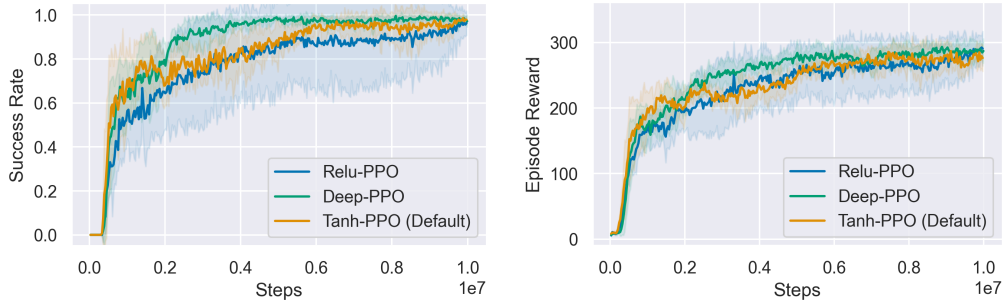
Containing historical output of these sensors over the last 3 steps, the proprioceptive observation is in 84 dimension. For LocoTransformer baseline method directly trained on test environment, the input is this state vector and first-view depth images in 64×64 over the last 4 steps.

F.3 IsaacGym

We use the same setting reported in [57], and the training code is from https://github.com/leggedrobotics/legged_gym.

G Learning Curves of Trained Agents

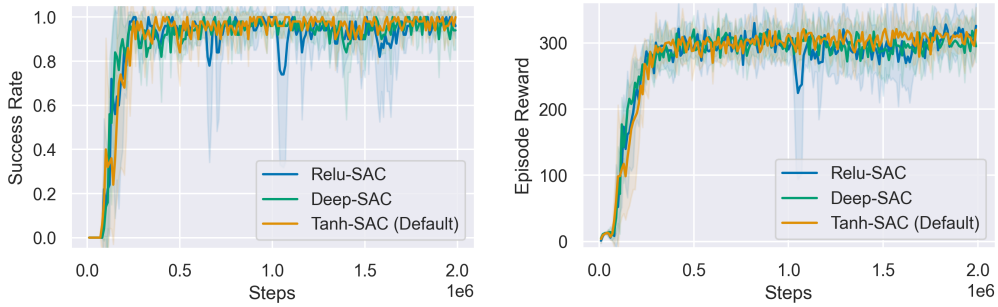
G.1 MetaDrive



(a) Success Rate of PPO Agents

(b) Episode Reward of PPO Agents

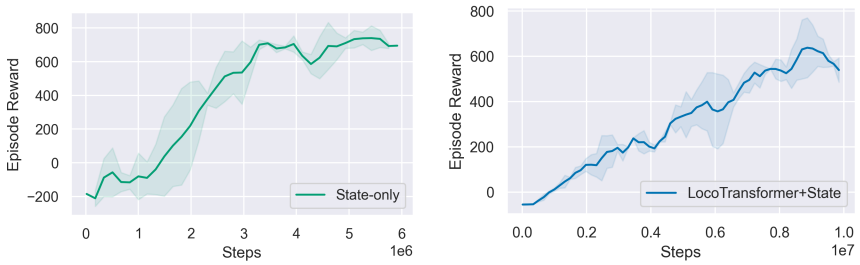
Figure 11: Success rate and episode reward of PPO agents on 50 training environments.



(a) Success Rate of SAC Agents (b) Episode Reward of SAC Agents

Figure 12: Success rate and episode reward of SAC agents on 50 training environments.

G.2 Pybullet-A1 Legged Robots



(a) Insensible Agent on **Training** Environment (b) Visual Input Agent on **Test** Environment

Figure 13: Training episode reward of two quadrupedal locomotion policies.

G.3 Gym Environment

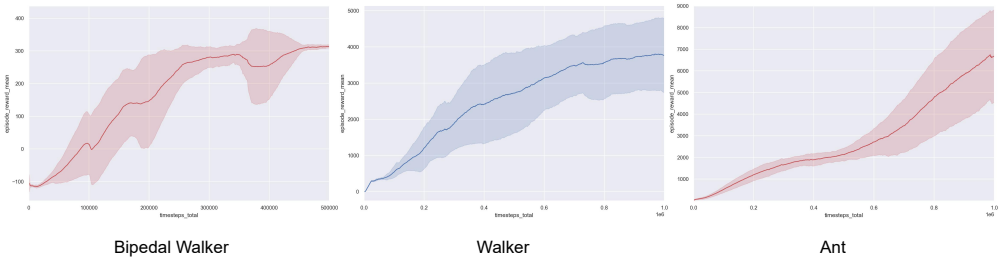


Figure 14: The episodic reward during the learning of SAC algorithms in Ant, Walker and Bipedal-Walker tasks. Though we repeat 5 times for each experiment, *Policy Dissection* is applied to the agent with highest performance to do qualitative study.

H Hyper-parameters

H.1 MetaDrive

Since SAC is not sensitive to the choice of hyterparameters, the learning rate $1\mathrm{e}{-4}$ is also suitable for Deep-SAC (6 Layer) and Relu SAC. For Deep-PPO and Relu-PPO, learning rate should be slightly decreased due to the change of neural network structure. Also note that the number of hidden units is changed to 128 per layer for Deep PPO with 6 layer. This is for keeping the total number of network variables approximate to the network used by default PPO (around 250,000 variables).

Table 3: SAC/ Deep SAC/ Relu SAC

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for target network update	0.005
Learning Rate	$1e-4$
Environmental horizon T	1500
Steps before Learning start	10000
Activation Function	"tanh" or "relu"
Prioritized Replay	False
Target Network Update Frequency	1
Soft Update τ	$5e-3$
MLP Hidden Units	256
MLP Layers	2 or 6

Table 4: PPO

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [62]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	20
Train Batch Size	30,000
SGD mini batch size	256
Learning Rate	$3e-4$
Clip Parameter ϵ	0.2
Activation Function	"tanh"
MLP Hidden Units	256
MLP Layers	2

Table 5: Deep PPO

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [62]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	20
Train Batch Size	30,000
SGD mini batch size	100
Learning Rate	$1e-4$
Clip Parameter ϵ	0.2
Activation Function	"tanh"
MLP Hidden Units	128
MLP Layers	6

Table 6: Relu-PPO

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [62]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	20
Train Batch Size	30,000
SGD mini batch size	100
Learning Rate	$5e-5$
Clip Parameter ϵ	0.2
Activation Function	"Relu"
MLP Hidden Units	256
MLP Layers	2

H.2 Gym Environments and Pybullet-A1 Legged Robots

Agents of Gym environments (Walker/Ant/Bidepadel Walker) are trained by SAC and share the same hyperpatrameter setting. Quadrupedal robots are trained by PPO, and we follow the same codebase and configuration used in [76].

Table 7: SAC for Gym agents

Hyper-parameter	Value
Discounted Factor γ	0.99
τ for target network update	0.005
Learning Rate	$3e-4$
Environmental horizon T	1500
Steps before Learning start	1000
Activation Function	"tanh"
Prioritized Replay	False
Target Network Update Frequency	1
Soft Update τ	$5e-3$
MLP Hidden Units	256
MLP Layers	2

Table 8: PPO for Legged robots

Hyper-parameter	Value
KL Coefficient	0.2
λ for GAE [62]	0.95
Discounted Factor γ	0.99
Number of SGD epochs	3
Train Batch Size	16384
SGD mini batch size	1024
Learning Rate	$1e-4$
Clip Parameter ϵ	0.2
Activation Function	"tanh"
MLP Hidden Units	256
MLP Layers	4