Reinforcement Learning for Quadruped Locomotion with Visual Signals

Kaifeng Zhang*
IIIS, Tsinghua University
Beijing, China
zhangkf19@mails.tsinghua.edu.cn

Xingzhuo Guo*
IIIS, Tsinghua University
Beijing, China
gxz19@mails.tsinghua.edu.cn

Abstract

Locomotion learning for legged robots is becoming an increasingly promising field. In this paper, we focus on several practical training methods. We start with reproducing LocoTransformer, an RL method for quadruped locomotion with visual input. Next, we add the evolutionary trajectory generator to improve on this end-to-end RL model, which achieves efficiency gains and better results. We also study the effect of shaping the reward function and tuning environment parameters. Finally, we show that the learned policy can successfully make the robot walk in the given direction and avoid pillar-shaped obstacles.

1 Introduction

Learning legged locomotion is a long-standing problem in robotics. Traditional approaches start from physical models and cybernetics to enable robots to learn basic motor skills, such as walking, running, and jumping. However, traditional approaches rely heavily on the precision of physical models and manual parameter tuning. In addition, it is hard to apply traditional methods to noisy environments, especially the real-world scenario.

With the rapid development of reinforcement learning and deep neural networks, learning locomotion through RL is becoming more promising. Different from traditional methods, RL does not require explicit physical models. Complex strategies are learned automatically by neural networks, which reduces heavy manual tuning. Moreover, by introducing noise to the environment, a model can generalize better and become more applicable to real-world scenarios. RL can also deal with multi-modal input, such as RGB images or depth maps, by aggregating them into the policy network.

However, RL for legged locomotion also faces many difficulties. First of all, Training RL models can be sample-inefficient, time-consuming, and unstable. Secondly, the detailed setting and design of an RL model are often crucial for performance. Different settings and tricks are not transferrable among environments, making the model environment-sensitive.

There are several previous works in the field of quadruped locomotion. For this project, We mainly focus on two of the latest: ETG-RL [**?**] and LocoTransformer [**?**]. The setting of the two works is somehow complementary: ETG-RL is an RL-based method for learning versatile locomotion skills without visual input; LocoTransformer focuses on learning to walk and avoid obstacles with visual signals. ETG-RL's code is publicly available, while LocoTransformer's code is not released. In our work, we propose to build LocoTransformer using the environment of ETG-RL. Further, we combine the two methods to achieve better performance.

*Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

2 Related Work

RL for quadrupedal locomotion It is intrinsically hard to learn quadruped locomotion directly through reinforcement learning. The reason is, the degree of freedom is not only large but also highly entangled for a quadruped. Moreover, the quadruped must adopt a quasi-periodic gait which is hard to learn. Therefore, most works use trajectory control instead of torque control.

[**?** **?**] divide the trajectory into a periodical term and a residual term. The periodical term can generate basic gaits, while the residual term can adapt to different physical environments. [**?**] applies an adaptation module to enable fast adaptation to unseen terrains; [**?**] uses a centroidal model to simplify the reward design and enables efficient simulation; [**?**] enables robots to learn agile locomotion skills by imitating animals.

ETG-RL The ETG-RL [**?**] borrows the idea of periodical terms from [**?**]. Concretely, it proposes the CPG-RBF network, a generator for various kinds of periodical trajectories from sine signals. It uses an evolutionary strategy to update the CPG-RBF network, forming the evolutionary trajectory generator (ETG). A policy network learns the residual trajectory with the SAC algorithm. ETG-RL successfully learns complicated skills such as walking upstairs, jumping, walking a balance beam, and walking in a cave.

Visual signal and LocoTransformer Theoretically, with visual signals, a robot can predict and adapt to a more complex environment, and perform more complex behaviors, such as avoiding obstacles and planning the route. Introducing vision signals into the RL framework requires aggregating visual signals and proprioceptive states, which is hard given the high dimensionality of visual images.

LocoTransformer [**?**] is a successful work to train vision-based quadrupeds. The model equips the robot with a LiDAR. It divides the depth map into patches and feeds both the processed proprioceptive states and the patches into a Transformer. Unlike previous works, LocoTransformer does not rely on periodical motion prior; the model directly outputs the actions in an end-to-end manner.

Reward Design The reward for quadruped locomotion is usually defined as the robot's displacement along the target direction minus the energy consumed by the control actions. For some complex tasks, additional terms are introduced, such as punishment on the height of legs, the roll, pitch, etc.

3 Problem Formulation

Our problem setting mainly follows LocoTransformer [**?**]: designing a framework to train the quadrupedal robot to learn locomotion strategies (e.g. walking, avoiding obstacles, overcoming complex terrains) with both proprioceptive states and visual inputs.

The locomotion task is modeled as a finite-horizon, discounted Markov decision process $\mathcal{M}$. Given the action space $\mathcal{A}$, the state space $\mathcal{S}$, the state transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the optimal stochastic policy $\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})$ can be determined by maximizing the expected discounted reward $R(\pi)$:

$$R(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T-1} \gamma^t r(s_{t-1}, a_{t-1})\right]$$

where s_t represents the state at time step t, a_t represent the action taken at s_t, $\gamma \in (0, 1]$ is the discount factor, T is the horizon, and $r(\cdot, \cdot)$ is the reward function.

We pay special attention to the reward design of two works: (a) LocoTransformer's reward: the linear combination of displacement reward, the torque regularization reward, and the living reward. (b) ETGRL's reward: a smoothed displacement reward with complex heuristic designs. In our experiments, we try both rewards and also add our modifications.

The evaluation metric for the robot's performance is a bit ambiguous. It is not reasonable to compare rewards under different reward functions. Therefore, we mainly focus on how well the robot performs the task, including the total moving distance, the velocity, the (undesired) lateral displacement, and the gait. We are also concerned about the training efficiency and stability when comparing different training settings and optimization methods.

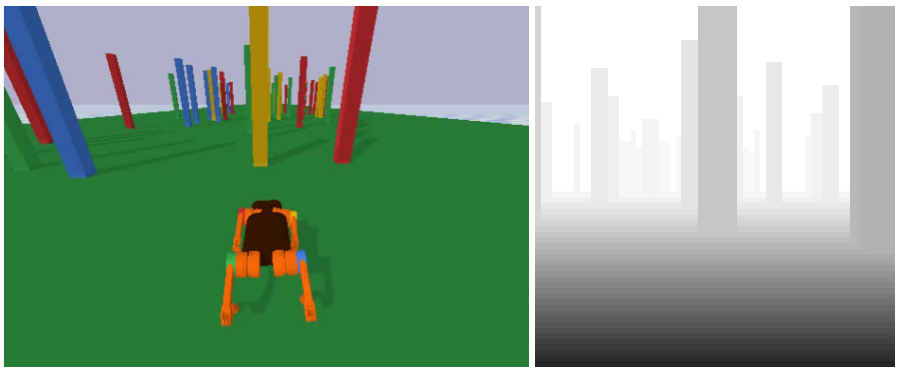

Figure 1: Simulation environment. Left: ground with pillars. Right: the depth image.

4 Method

In this section, we will introduce the main body of our method. We first build the simulation environment based on the environment part of ETG-RL's publicly available code. Then we build the LocoTransformer, combine LocoTransformer with ETG, and perform reward design and parameter tuning to train these models.

4.1 Simulation Environment

We construct the simulation environment with the pyBullet library. In the environment, a 12-DoF Unitree A1 robot is installed on the ground with pillars (see Figure 1). At each timestep, the proprioceptive sensors will give the joint angles, the base displacement, and the IMU information. Meanwhile, the depth sensor equipped on the head of the robot will generate a 64×64 depth image (see Figure 1). An agent gives an action according to the output of sensors. An action is a 12-dim vector representing the target joint angles. The environment will interpolate between the current joint angles and the action, and perform PD control for multiple steps for each action output.

The robot is trained on specially designed terrains. Most of our experiments were held on the pillar environment: the robot is walking on a plain with randomly-placed pillars, and it is expected to walk as far as possible in the forward direction. The pillars are tall cuboids that are slightly wider than the robot. The robot must use visual input to sense the position of pillars and dodge them. We also designed the hill environment: the ground is uneven, and the robot is also expected to walk in the forward direction.

4.2 LocoTransformer Model

To build the LocoTransformer, our main modifications on the ETG-RL code include adding depth sensors, redefining the observation space, modifying the reward function, implementing the Transformer model, and changing the training algorithm.

Proprioceptive states and visual signals Our observation space consists of both the proprioceptive states and the visual signals. For the proprioceptive states, we take features including joint angles, base displacement, IMU information (the roll, pitch, and the corresponding angular velocity), and the last action. The observation space takes the last 3 proprioceptive states, 93 dimensions in total. The observation space for the visual signal takes the last 4 states' depth images.

Multi-modal Transformer model We use a multi-modal Transformer model as a feature extractor for the observation. We feed the proprioceptive states into a two-layer MLP to obtain the proprioceptive embedding. We view the depth input as a 4-channel image and use a 3-layer CNN to get 4×4 patch embeddings. We concatenate the proprioceptive embedding and the patch embeddings and process with a 2-layer Transformer. After, we concatenate the proprioceptive output and the averaged patch outputs, then use a two-layer MLP projection head to obtain the observation feature. Finally, the observation feature derives the values and policies by MLP heads.

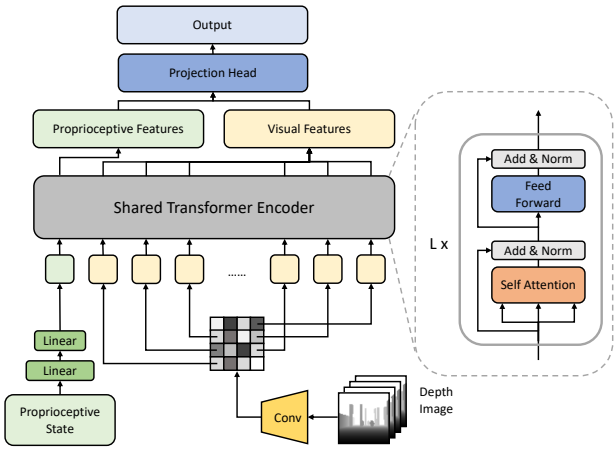

Figure 2: The architecture of the LocoTransformer.

PPO algorithm. We train our model with the proximal policy optimization (PPO) algorithm. (The original ETG-RL uses the SAC algorithm.) We use Stable baseline3 [**?**] as the framework.

4.3 Evolutionary Trajectory Generator

In experiments, we find that LocoTransformer is very hard to train compared with ETG-RL, possibly because it does not use a periodical motion prior. To solve the difficulties of end-to-end training, we seek to combine ETG-RL and LocoTransformer by applying an evolutionary trajectory generator (ETG) to the original LocoTransformer architecture.

ETG structure ETG is a successful approach for introducing periodical trajectory. It uses two sine waves to represent the swing and extension of each leg, then leverages a radial basis function (RBF) network to transform the sine wave into target trajectories. The process can be formalised as:

$$c_0(t) = A * \sin(\omega t), \quad c_1(t) = A * \sin(\omega t + B),$$
$$V_i(t) = \exp\left\{-\frac{(c_0(t) - \mu_{i,0})^2 + (c_1(t) - \mu_{i,1})^2}{\sigma_{RBF}^2}\right\}, \ i = 0, \cdots, H-1,$$
$$P = W \cdot V + b,$$

and the trainable parameters include W and b. We train CPG-RBF with the ES algorithm (the code of the algorithm is available in ETG-RL).

Pretrained model In our early experiments on blind robots with ETG, we find that it is hard to train a robot from scratch. ETG-RL proposes a solution by pretraining the ETG parameters on a simple forward task, then fine-tuning the pre-trained model on more complex tasks. In our work, we also load the pre-trained model provided by ETG-RL.

4.4 Heuristic Reward and Parameter Tuning

The original reward of the LocoTransformer only contains the forward velocity term and the energy term. To better fit the task of dodging pillars, we additionally add some regularization terms as heuristic rewards.

Deviation Regularization The deviation regularization aims to penalize the lateral deviation of the robot. Suppose the lateral offset of the robot is y. Then the reward term can be written as $-\alpha \cdot y^2$. Intuitively, this reward can force the robot to return to the original forward track after dodging a pillar.

Drift regularization The drift regularization aims to penalize the direction inconsistency between the velocity and the pose. Formally, we use the negative value of the robot's velocity component along the horizontal axis. This design can reduce the horizontal drift of the robot.

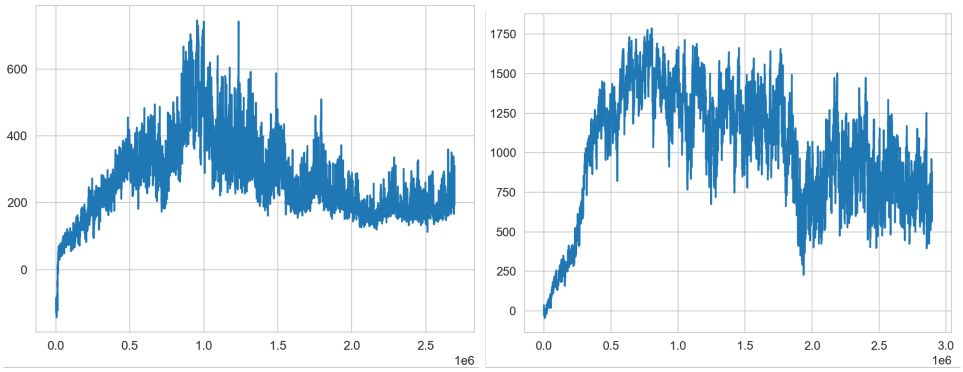

Figure 3: The reward plot of reproducing ETG-RL. Left: training from scratch. Right: training with pretrained ETG weights. The reward of the pretrained scenario is larger and the training is faster as well.

Figure 4: The training result of LocoTransformer. The robot does not learn correct gaits. It uses its knees to contact with the ground.

To find the best setting for training the two works, we also focus on some model details, such as the method to calculate energy (based on torque vs. based on power), the action bound (tighter bounds give less freedom but reduce training complexity), etc. We also tune several crucial parameters, including reward weights, simulation timestep length, etc.

5 Experiment Results

5.1 Reproducing ETG-RL

The ETG-RL is easy to reproduce using open-source code. However, we find that training the model is hard without using the pre-trained ETG weights. Without using the weights, the robot cannot learn to climb upstairs in most of our experiments even after 7 days of training (its front legs will be stuck under the first or the second stair). On contrary, when using pre-trained weights, the model can learn to climb up multiple stairs in a much shorter time. The reward plot is shown in Figure 3.

5.2 Reproducing LocoTransformer

We implemented the model structure of LocoTransformer as introduced in section 4.2, and hyper-parameter settings mostly follow the original paper [**?**]. We train the model for 3 days, but the robot still cannot learn correct gaits (see Figure 4) for walking, and it usually falls to the ground quickly.

5.3 Combining ETG and LocoTransformer

We combine the ETG and LocoTransformer such that the Transformer model only outputs a residual trajectory. Other settings remain unchanged. We find that this combination is successful in accelerating the training. The robot can learn to maintain balance and continuously walk in a shorter period.

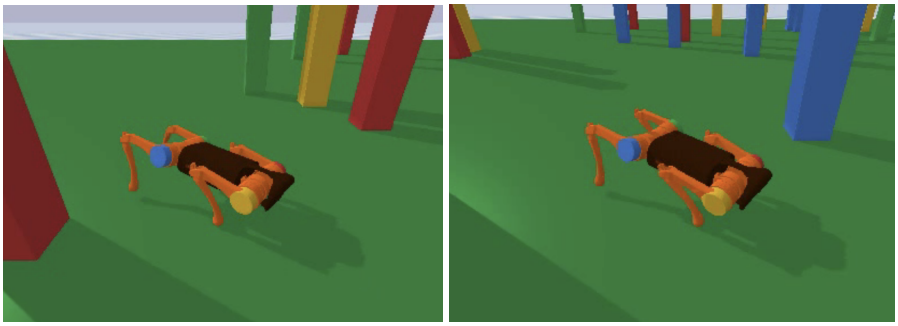

Figure 5: The training result of LocoTransformer with ETG. The robot is able to learn a stable gait and dodge more than 2 pillars.

The gaits of the robot are shown in Figure 5. Although it is not perfect, there is an improvement over the original LocoTransformer.

5.4 Reward Design

We focus on the reward design for combining ETG and LocoTransformer. In our experiments, we vary the coefficient for the energy reward term. We also design deviation regularization reward and drift regularization reward. The main results are shown in Figure 6.

For experiments (a) to (c), we observe that by adding deviation reward and drift reward, the robot can learn to gradually increase both rewards and episode lengths. The reward increases quickly in the initial stages of training, indicating that the training is relatively fast. (d) and (e) show 2 unsuccessful attempts: (d) increases the coefficient for the energy reward, but the robot only learns to end the episode as soon as possible. (e) uses the original ETG-RL reward, but the model fails to learn anything useful.

Figure 9 shows a visualization of the robot's performance, with one of our best models (in experiment (c)). We can see that the gait is better than all previous methods, and the robot can walk for the longest distance. The robot can even restore balance after slightly colliding with a pillar.

Figure 8 shows a visualization of the attention map between the embedding of the proprioceptive states and the embedding of depth maps of the last layer of the Transformer. We can see that the model learns to attend to positions where pillars exist.

5.5 Hill Terrain

Besides the pillar terrain, we also designed the hill terrain. Robots must overcome the uneven shape of the ground and move forward. In the ideal situation, the visual signal can help the robot detect the shape of the terrain and plan the route. However, training on this terrain is harder than the pillar terrain. The episode length remains relatively low during training. (see Figure 6).

We also find that by transferring the learned policy in the pillar terrain directly to the hill terrain, the robot can walk forward for a short distance. Namely, the model shows some basic robustness.

6 Conclusion & Discussion

In this project, we studied the topic of quadruped locomotion and implemented a model for learning quadruped locomotion with both visual signals and an evolutionary trajectory generator. We start from the open-sourced code of ETG-RL and reproduced the model architecture of LocoTransformer. Then we combine LocoTransformer and ETG-RL to form our new model. In the experiments, we find it hard to train LocoTransformer from scratch. After equipping LocoTransformer with the ETG structure and using the pre-trained weights from ETG-RL, the training becomes more efficient. Further, by adding additional heuristic rewards, the performance is greatly improved.

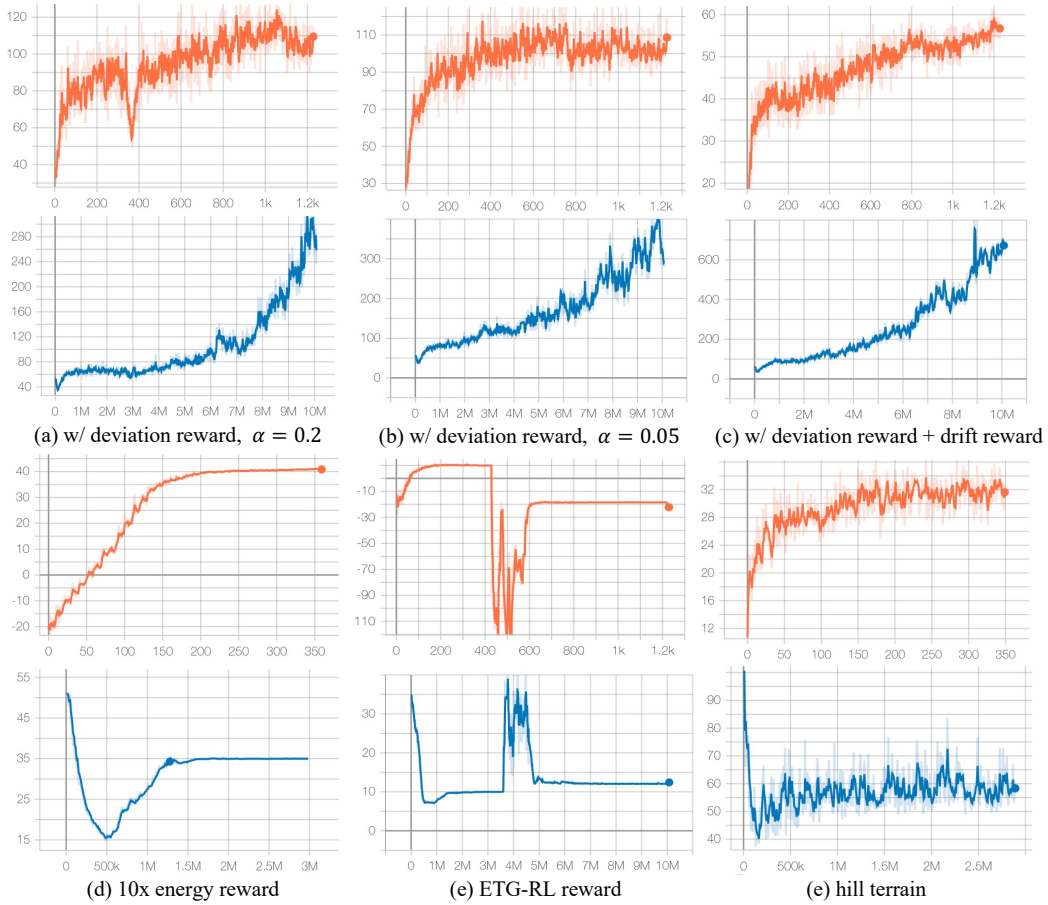

Figure 6: The reward curves and episode length curves of experiments. (a)∼(e): reward design experiments. (In all our experiments, we take $\gamma = 0.99$.) (f): the training reward curve on the hill terrain.

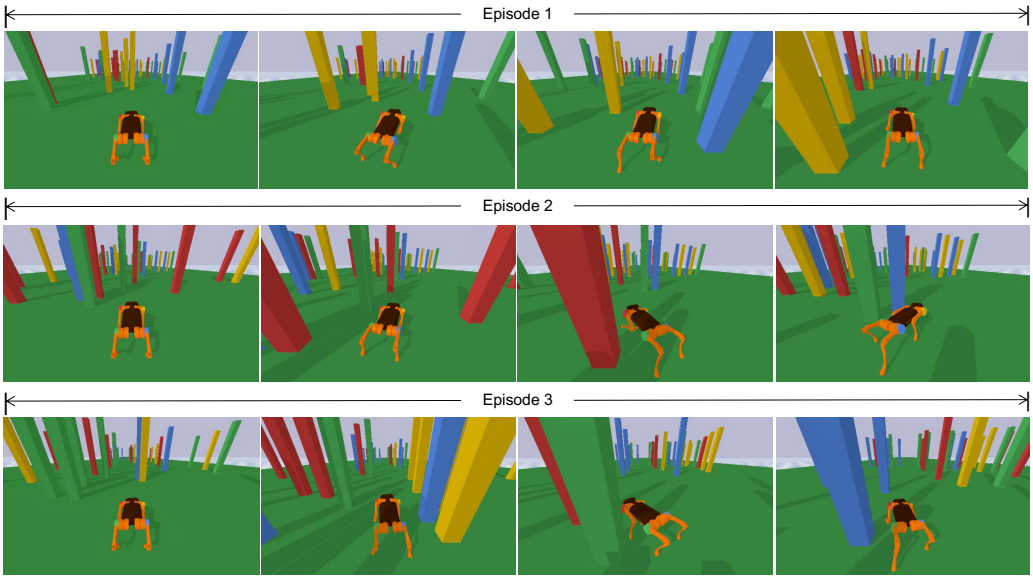

Figure 7: The visualization result of robot evaluation. The robot is able to maintain balance and continuously dodge pillars in an episode.

Figure 8: The visualization result of the self attention map(red). The attention values have been normalized by Softmax.

Figure 9: Visualization of the robot on the hill terrain, and the depth map of the hill terrain. The robot still easily loses balance after 3 days of training (middle).

To summarize, the crucial factors for building a successful RL model for locomotion include periodic trajectory, environment design, pre-trained weights for initialization, and reward design. Environments with uneven grounds are more challenging than environments with flat grounds. It is important to use careful reward design to guide the robot to maintain balance.

Currently, our model is still imperfect. Failure patterns include suddenly losing balance and colliding with obstacles. The hill terrain and more complex terrains are still unsolved. Possible future works include:

- designing better rewards for the hill terrain.
- designing other terrains and tasks (e.g. complex-shaped obstacles).
- comparing between different model architectures and different learning algorithms; try modifying some model details (e.g. learning rate, PD control parameters).

References

A Workload distribution

We have approximately equally divided the workload, listed as follows.

- Xingzhuo Guo: processing the ETG-RL source code, implementing the LocoTransformer, performing ETG-RL and LocoTransformer experiments, presentation, report
- Kaifeng Zhang: surveying previous works, implementing the terrains, performing reward design experiments, visualization, proposal, presentation, report