Look Before You Leap: Safe Model-Based Reinforcement Learning with Human Intervention

Anonymous Author(s)
Affiliation
Address
email

Abstract: Safety has become one of the main challenges of applying deep reinforcement learning to real world systems. Currently, the incorporation of external knowledge such as human oversight is the only means to prevent the agent from visiting the catastrophic state. In this paper, we propose MBHI, a novel framework for safe model-based reinforcement learning, which ensures safety in the state-level and can effectively avoid both local and non-local catastrophes. An ensemble of supervised learners are trained in MBHI to imitate human blocking decisions. Similar to human decision-making process, MBHI will roll out an imagined trajectory in the dynamics model before executing actions to the environment, and estimate its safety. When the imagination encounters a catastrophe, MBHI will block the current action and use an efficient MPC method to output a safety policy. We evaluate our method on several safety tasks, and the results show that MBHI achieved better performance in terms of sample efficiency and number of catastrophes compared to the baselines.

Keywords: Safety RL, Model-based RL, Model Predict Control

1 Introduction

Deep reinforcement learning (DRL) is proposed as an automated framework for intelligent decision-making problems, and has shown tremendous progress in various domains such as video games [1, 2], board games [3, 4] and robotic control [5, 6]. However, most of these success were achieved in the idealized simulator where agents can interact with the environment without restriction. In real-world applications, random exploration may lead to equipment damage or even worse results [7]. Consequently, when applied in the real world, DRL algorithms need to ensure the security throughout the whole learning process, especially in safety-critical scenarios.

This has encouraged the research on safety in DRL, which aims to maximize the cumulative rewards while minimizing safety violations during the training processes [8, 9]. Safe RL can be divided into two categories: trajectory-based [9, 10, 11] and state-based [12, 13, 14]. The former uses the cost function to evaluate the safety of the whole trajectory, while state-based methods place restrictions on a state-wise basis. Safe RL algorithms such as CPO [10] and HIRL [12] have shown that it is possible to significantly reduce the number of constraint violations in safety-critical tasks.

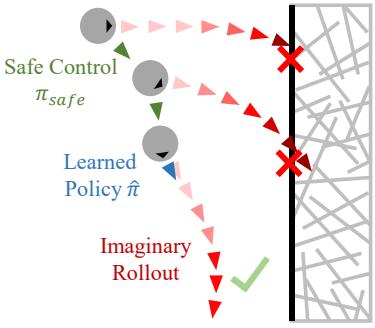


Figure 1: Imaginary safety detection and replacement of catastrophic actions in MBHI.

Unfortunately, most of existing methods have high sample complexity, which means that a large number of interactions with the real world are required, resulting in the increase of the number of catastrophe. Another shortcoming is that most of existing approaches use the explicitly defined constraint or safety cost function, while safety is subjective and always vague. For example, in autonomous driving, the overseer takes over control when he judges that the car is going to crash. The overseer makes judgement based on subjective feelings rather than vehicle dynamic model. Finally, it is more strict and natural to define safety over the state. However, existing state-based approaches only consider

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

the catastrophe of locally avoidable, that is, catastrophes can be avoided by choosing the appropriate action just before the catastrophe would have happened [12, 13, 14].

To address the above-described challenges, in this paper, we introduce a novel safety-aware model-based framework, called Safe Model-based Reinforcement Learning with Human Intervention (MBHI), which incorporates safe reinforcement learning with model-based methods and human knowledge. Intuitively, before performing an action, people will estimate the safety of a period of time in the future. In MBHI, an ensemble of deep neural networks is used to model a supervised Blocker to imitate the human overseer or explicit constraints, output the catastrophe probability of the state and evaluate the model uncertainty. As shown in Figure 1, at each time step, we roll out a trajectory with a certain number of steps in the learned dynamics to check the safety of the agent in the future. If dangerous, the MPC controller will take over and output a safe action to avoid possible future catastrophes in advance. Furthermore, safe active learning is integrated to enable grounded exploration only in safe regions, thereby accelerating model training on the premise of safety.

To summarize, the primary contributions of our work are as follows: (1) a novel safe model-based RL framework with human intervention, (2) safe active learning mechanism that only encourages agents to explore in safe regions, (3) a state-based action correction mechanism that perceive possible catastrophes in the future and avoid them in advance, (4) efficient MPC solving method that outputs a safe alternative policy in continuous action space.

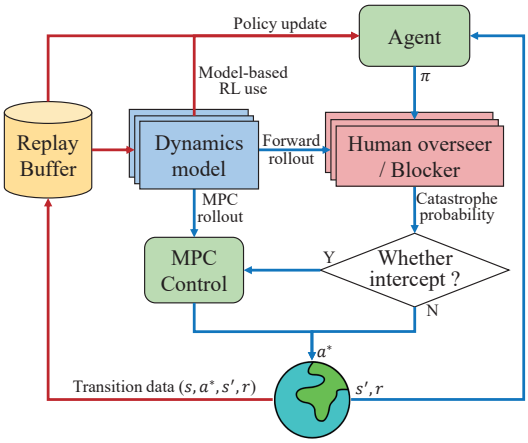


Figure 2: Overall framework of MBHI. The interaction phase is shown in blue and the policy learning in red.

2 Related Work

Safe RL: An interaction process is called safe if the agent rarely accesses the dangerous state [9, 11, 14]. Introducing external knowledge is the only way to ensure safety [12, 15]. Recently, various kinds of explicit constraints are used to impose some form of safety measures. To name some, constraints on expectation [9, 10], constraints on uncertainty [7, 16], and Conditional Value-at-Risk [11, 17, 18]. Unlike the above trajectory-based safety guarantees, [14] and [19] solve an optimization problem in the policy-level to ensure the safety of states. In the setting of implicit constraints, HIRL [12] and [13] train a supervised learner to imitate the human's intervention decisions and use it to evaluate the safety of the current state-action pairs. However, these state-based approaches can only prevent the catastrophe that can be avoided just before the catastrophe would have happened. This work is also focus on state-based safety and ensures safety with a Blocker that imitates human overseer. The main difference of our method is that we incorporate the model-based method to accelerate the policy learning, and can avoid non-local catastrophes.

Model-based RL: Model-based RL exhibits low sample complexity and fast convergence. The learned dynamics model can be used for planning [4, 20], value expansion [21, 22], or imaginary training [23, 24, 25]. In planning approaches, MBMF [26] and PETS [20] use Model Predictive Control (MPC) to select actions based on the learned model. MVE [22] and STEVE [21] incorporate value expansion to improve the targets for temporal difference learning. ME-TRPO [23] and MBPO [24] are directly trained on imaginary data to accelerate policy learning. To limit the influ-

ence of the model error, the uncertainty estimation is widely used in MBRL. For instance, PETS [20] use uncertainty-aware models for trajectory propagation and planning. STEVE [21] interpolates between different imaginary rollout lengths to favor those targets with lower uncertainty. M2AC [27] uses the masking mechanism to only use the model when the model error is small.

In this work, STEVE is used as the base learning algorithm, but our framework is generally applicable to other model-based algorithms with uncertainty quantification, such as MBPO, M2AC, etc. We choose STEVE for three main reasons. Firstly, it explicitly estimates uncertainty for both dynamics and Q-function. Secondly, if the dynamics model cannot be learned or the Q-value network converges better, STEVE can reduce to model-free algorithm instead of learning from inaccurate imaginary data. Thirdly, since STEVE is an off-policy algorithm, the safety detection process in MBHI can be integrated with value expansion at each time-step to reduce calculation amount.

3 Preliminaries

3.1 Markov Decision Processes

We focus on Markov Decision Process (MDP) with continuous states and action spaces, defined as $(\mathcal{S}, \mathcal{A}, r, p_0, p, \gamma)$, in which $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $p_0 : \mathcal{S} \to \mathbb{R}_+$ is the initial state distribution, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the dynamics function, and $\gamma \in [0, 1]$ is the discount factor. The agent starts from an initial state $s_0 \sim p_0$, at each time step, the agent chooses an action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ according to a policy $\pi_\theta(s_t)$ parameterized with θ, transitions to a successor state $s_{t+1} = p(s_t, a_t)$ and receives a reward $r_t = r(s_t, a_t, s_{t+1})$. The goal of reinforcement learning is to find an optimal policy π_{θ^*} that maximize the expected discounted sum of rewards $\theta^* = \arg\max_\theta \mathbb{E}_{s_0 \sim p_0}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

3.2 Stochastic Ensemble Value Expansion

In model-based RL, the agent learns the world model from the replay buffer $\mathcal{D}$ to approximate the true dynamics function. In STEVE, the dynamics model consists of the following modules:

$$\begin{aligned} \text{Transition model}: &\quad s_{t+1} \sim \hat{T}_{\boldsymbol{\xi}}(s_{t+1}|s_t, a_t) \\ \text{Reward model}: &\quad r_t \sim \hat{R}_{\boldsymbol{\phi}}(r_t|s_t, a_t, s_{t+1}) \\ \text{Termination model}: &\quad d_t \sim \hat{D}_{\boldsymbol{\psi}}(d_t|s_t). \end{aligned} \tag{1}$$

An ensemble of parameters is maintained to estimate uncertainty in the learned dynamics (i.e. $\boldsymbol{\xi} = \{\xi_i\}_{i=1}^{N_T}$, $\boldsymbol{\phi} = \{\phi_i\}_{i=1}^{N_R}$, $\boldsymbol{\psi} = \{\psi_i\}_{i=1}^{N_D}$). The entire dynamics model is optimized as follows:

$$\min_{\boldsymbol{\xi},\boldsymbol{\phi},\boldsymbol{\psi}} \mathbb{E}[||s' - \hat{T}_\xi(s, a)||^2 + ||r - \hat{R}_\phi(s, a, s')||^2 + \mathbb{H}(d(s'), \hat{D}_\psi(\hat{T}_\xi(s, a)))] \tag{2}$$

$\mathbb{H}$ is the cross-entropy, and the expectation is calculated over collected data $\{(s, a, s', r)\}$.

For actor-critic methods or Q-learning methods, action-value function $\hat{Q}_\varphi(s, a)$ is a critical quantity to guide policy updating. The Temporal-Difference target is written as:

$$T^{TD}(r, s') = r + \gamma(1 - d(s'))\hat{Q}_{\varphi^-}(s', \pi_\theta(s')) \tag{3}$$

where φ^- means target network. STEVE improves TD targets by combining the short-term value estimate using the dynamics and long-term value estimate using $\hat{Q}_{\boldsymbol{\varphi}}$. An ensemble of Q-functions is also maintained to evaluate the relative uncertainty between the dynamics and Q-function. Through these uncertainty estimates, STEVE dynamically interpolates between imagined rollouts of different horizon lengths, favoring those candidate targets with lower uncertainty, and give up utilizing them when significant errors are introduced. Thus, the interpolated target is written as

$$T_h(r, s') = r + (\sum_{i=1}^{h} D_i \gamma^i \hat{R}_\phi(s'_{i-1}, a'_{i-1}, s'_i)) + D_{h+1}\gamma^{h+1}\hat{Q}_{\varphi^-}(s'_h, a'_h) \tag{4}$$

$$T_H^{STEVE} = \sum_{h=0}^{H} \frac{\sigma^2(T_h)^{-1}}{\sum_j \sigma^2(T_h)^{-1}} T_h^\mu \tag{5}$$

where $s'_i = \hat{T}_\xi(s'_{i-1}, a'_{i-1})$, $D_i = d(s') \prod_{j=1}^{i}(1 - \hat{D}_\psi(s'_j))$. If the environment is too complex or noise, STEVE will ignore the model and reduce to model-free algorithm immediately.

4 Methods

In this section, we present a novel model-based safe RL framework, MBHI, as shown in Figure 2. In the interaction phase, at each time-step, we use the current policy network π_θ to roll out multiple steps in the learned dynamics, and check the safety of the imaginary trajectories with the supervised Blocker. If the catastrophe probability of the imaginary future rollout is greater than the threshold, the output action will be intercepted and replaced by the MPC controller. In policy learning phase, the model-based method is used to update the policy. Moreover, to overcome catastrophic forgetting problem of the network [12], we split replay buffer into safe and unsafe categories.

4.1 Catastrophe Prediction Network

Similar to HIRL [12], a supervised learner is trained to imitate the human overseer and block actions that are unsafe, but we maintain an ensemble of parameters for the Blocker $\mu = \{\mu_i\}_{i=1}^{N_B}$ to evaluate the uncertainty. In concrete terms, the human-imitator method has the following advantages. 1) Blocker can learn from data logs, human intervention data or explicit safety constraints by supervised learning. 2) Blocker is modular, it can be applied in distributed system and different tasks. 3) As binary network, the output value of the Blocker can be regarded as the distribution distance between the input state and the catastrophe zone. 4) Blocker can provide gradient information.

During human oversight phase, we store (s, a, r, s', c) at each time-step, where c is a binary label for whether the current state is dangerous defined by humans. Thus, the Blocker can be trained in a supervised manner by minimizing the cross entropy loss as:

$$\mathcal{L}_{\mu_i} = \mathbb{H}(c, \hat{B}_{\mu_i}(s')) \qquad 1 \leqq i \leqq N_B \tag{6}$$

Each $\hat{B}_{\mu_i}$ is initialized with various weights and trained on different input sequences. At this stage, for model-based RL, the collected data can also be used to train the dynamics. Then, the loss function can be rewritten as:

$$\mathcal{L}_{\boldsymbol{\mu},\boldsymbol{\xi},\boldsymbol{\phi},\boldsymbol{\psi}} = \mathbb{H}(c, \hat{B}_\mu(\hat{T}_\xi(s,a))) + ||s' - \hat{T}_\xi(s,a)||^2 + ||r - \hat{R}_\phi(s,a,s')||^2 + \mathbb{H}(d(s'), \hat{D}_\psi(\hat{T}_\xi(s,a))) \tag{7}$$

For the sake of fairness, in the experiment, we did not pre-train the dynamics model in the human oversight phase.

4.2 Safe Active Explorationm

In order to reduce model uncertainty, we aim to query those points with large entropy, or where the disagreement would be most reduced in a posterior model. In our settings, the dynamics model takes the form of ensemble, and the disagreement can be quantified by calculating the empirical variance across the output of each model. Therefore, the optimization objective can be written as:

$$\pi_\theta = \max[\mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}((1-\lambda)r_t + \lambda c_a \sigma^2(\hat{T}_{\boldsymbol{\xi}}(s_t, a_t)))]] \tag{8}$$

where c_a is the coefficient of the active learning reward and λ is a policy dependent weighting. The second term of Eq. 8 can also be regarded as maximizing the differential entropy of the transition model. The model outputs is in the form of empirical Gaussian distribution, its differential entropy can be easily obtained, given by $H(x) = \frac{1}{2}(\ln(2\pi\sigma^2) + 1)$. Therefore, maximizing variance is equivalent to maximizing model entropy. It is also possible to maximize the variance of the predicted reward, as proposed in [28], but we find it perform poorly in the sparse reward environment.

However, for safe RL, we do not want agents to visit unsafe regions, even if these areas have greater uncertainty. A simple yet effective method is to mask the exploration reward near the dangerous area. The catastrophe probability predicted by the Blocker can be regarded as the distribution distance to the unsafe region. Thus, Eq. (8) can be rewritten in the form of safety-aware.

$$\pi_\theta = \max[\mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T}((1-\lambda)r_t + \lambda c_a \sigma^2(\hat{T}_{\boldsymbol{\xi}}(s_t, a_t))(1 - \mathbb{E}_{\boldsymbol{\mu}}[\hat{B}_\mu(s_t)])^\alpha)]] \tag{9}$$

where α is the weighting of safety, which controls the masking rate. The value of λ determines the level of exploration guided by the learned transition model and it should be asymptotically

consistent. While encouraging exploration, the final policy should be to maximize environmental rewards. In this work, we use the uncertainty measure of the dynamics to adjust λ adaptively. Formally, λ is defined as $\lambda_{t+1} = \frac{\bar{\sigma}_t^2(\hat{T}_\xi(\tau))}{2\cdot\max_k \bar{\sigma}_{k<t}^2(\hat{T}_\xi(\tau))}$, where $\bar{\sigma}_t^2$ denotes the mean variance. This restricts λ to $[0, 0.5]$ and the value of λ should decrease over time when different models converge.

4.3 Replacement of Catastrophic Actions

Before the agent execute the action to the environment, a C step trajectory from current state is rolled out in each dynamics. The catastrophe probability of the imagination is computed as follows.

$$p = \max_{c\in C} \mathbb{E}_{i\sim N_B, j\sim N_T}[\hat{B}_{\mu_i}(\hat{T}_{\xi_j}(s_{c-1}^j), \pi_\theta(s_{c-1}^j))] \tag{10}$$

If p is greater than the safety threshold, it is considered that the agent will have a disaster in a certain number of steps under the current policy, and it is necessary to replace the action in advance. In HIRL [12], the author selects the alternative action based on a lookup table or the ranking of the logit score. However, these methods are not suitable for continuous action space and "non-local" catastrophes. In this work, Model-Predictive Control is used as an expert to output safe actions. Given the current state s_t, the MPC prediction horizon H and the action sequence $a_{t:t+H-1}$, each dynamics model is used to predict the resulting trajectories $\{(s_{t+i}, r_{t+i-1})\}_{i=1}^H$. At each time-step t, the MPC controller solves an optimal control problem of the following form:

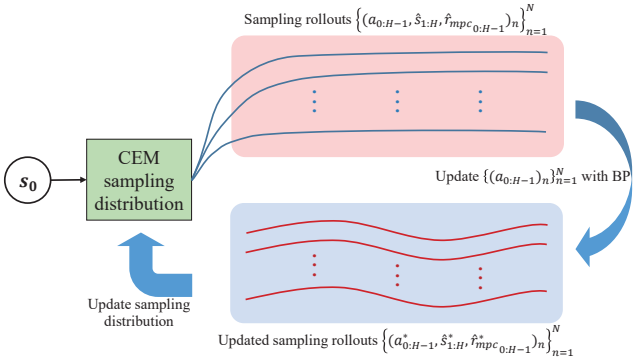


Figure 3: Schematic of MPC optimization method.

$$\max_{a_t,\ldots,a_{t+H-1}} \sum_{h=0}^{H-1} \hat{r}_{MPC}(s_{t+h}, a_{t+h}, s_{t+h+1}) \qquad s.t. \quad s_{t+h+1} = \hat{T}_\xi(s_{t+h}, a_{t+h}) \tag{11}$$

where $\hat{r}_{MPC}$ is the MPC reward. There are two requirements of it. First, guide the agent complete the obstacle-independent task. Second, guide the agent away from the unsafe regions. In consequence, $\hat{r}_{MPC}$ contains the predicted external reward and the predicted disaster probability. In addition, we further introduce a Leave-One-Out safety estimation to make MPC samples as far away from the unsafe region as possible. Given the unsafe dataset $\mathcal{D}_{us}$ and the MPC sampled state s_{t+h}, the estimator is defined as $u_{t+h} = D_{KL}[p(\cdot|\mathcal{D}_{us} \cup s_{t+h})||p(\cdot|\mathcal{D}_{us})]$. A small u_{t+h} indicates that the sampled state is dangerous. Thus, the MPC return can be formulated as

$$\hat{r}_{MPC}(s_{t+h}, a_{t+h}, s_{t+h+1}) = \mathbb{E}_{\boldsymbol{\phi}}[\hat{R}_\phi(s_{t+h}, a_{t+h}, s_{t+h+1})] + \mathbb{E}_{\boldsymbol{\mu}}[\hat{B}_\mu(s_{t+h+1})] + u_{t+h+1} \tag{12}$$

Finally, the controller executes the first action a_t to the environment, advances to the next time-step, and recheck the safety of the agent.

To improve the efficiency and performance of MPC, especially in high-dimensional action space, as shown in Figure 3, the gradient of the dynamics is used to guide the update of CEM sampling distribution. Each iteration consists of three steps: 1) Sample action sequences from CEM sampling distribution, and calculate MPC return with Eq. (12) . 2) Fix the dynamics parameters, and update action sequences via gradient ascent method. 3) Update the parameters of CEM sampling distribution. The action replacement mechanism is summarized in algorithm 1 in Appendix A.

A negative reward must be given to guide the agent to learn a safe policy. The interception mechanism in MBHI considers the danger of the current policy in state s_t rather than the state-action pair. Thus, the negative reward should solely depend on the state. In this work, an intrinsic reward signal is generated based on how far the agent is from the unsafe region to encourage the agent to stay away from these areas. Specifically, the output of Blocker is served as this kind of measurement. Besides, to increase the punishment near the unsafe area, a boundary value is introduced. When the

predicted catastrophe probability is greater than this value, a larger scaling factor is used to increase the negative reward. The intrinsic reward signal is computed as

$$r^i(s,a,s') = \begin{cases} -c_l \mathbb{E}_{\boldsymbol{\mu}}[\hat{B}_\mu(s')] & if\ \mathbb{E}_{\boldsymbol{\mu}}[\hat{B}_\mu(s')] < Bound, \\ -c_h \mathbb{E}_{\boldsymbol{\mu}}[\hat{B}_\mu(s')] & else. \end{cases} \tag{13}$$

where c_l and c_h are scaling factors. Finally, the overall optimization problem can be written as,

$$\pi_\theta = \max[\mathbb{E}_{tau\sim\pi_\theta}[(1-\lambda)(r^e(\tau) + r^i(\tau)) + \lambda r^a(\tau)]] \tag{14}$$

where r^e is the environment reward, and r^a is the active learning reward in Eq. (9). The full procedure is outlined in Algorithm 2 in Appendix A.

5 Experiments

5.1 Experiment Setting

In this section, we evaluate our approach and various baselines over five safety-critical continuous control tasks. 1) PuckWorld-L & PuckWorld-H: they differ in acceleration. The catastrophe of PuckWorkd-H is locally avoidable, while PuckWorkd-L is not. The agent is rewarded for reaching the target and is constrained to not touch the barrier. 2) Reacher: the agent is rewarded for reaching the target without touching the vertical bar. 3) Ant-Limit: the agent needs to move forward between two parallel barriers. 4) Ant-Block: the agent needs to move forward and avoid obstacles in the path. The last three tasks are adapted from PyBullet's environments [29]. A visualization of these environments and details of these tasks are given in Appendix B.

We compare our proposed MBHI against both the model-based and model-free baselines, DDPG and STEVE, and human intervention method, HIRL [12]. To ensure a fair comparison, we extend HIRL to the continuous action space and use the same MPC-based method in MBHI for action replacement. Besides, STEVE is also used as the basic learning algorithm of HIRL. In all experiments, we use the same network structure of the policy, value, dynamics and Blocker for all algorithms, and the number of ensemble models is always 4. Details can be found in Appendix C.

5.2 Comparison of Performance

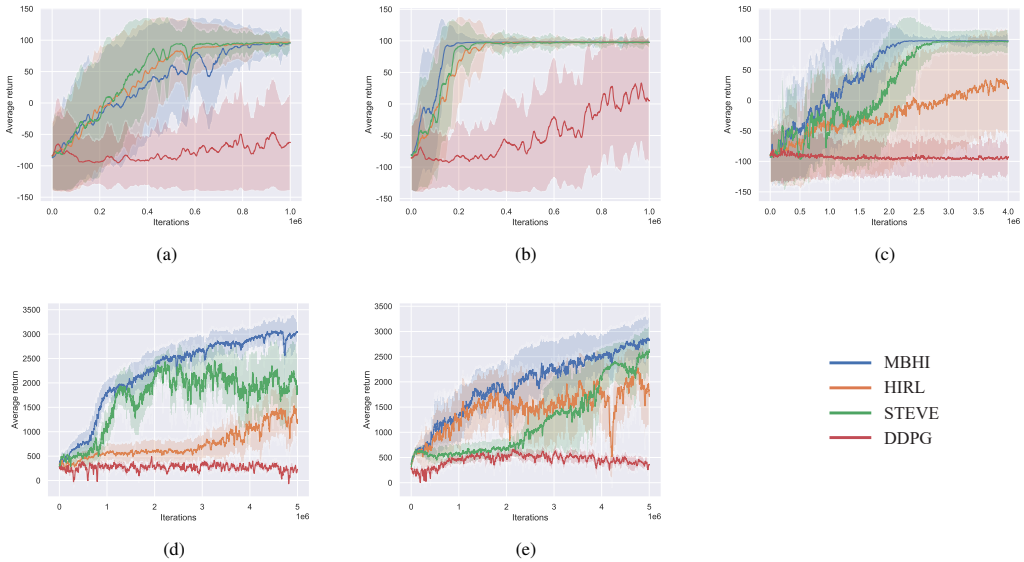


Figure 4: Learning curves of MBHI and baselines on five continuous control tasks over time (mean and standard deviation). (a) PuckWorld-L, (b) PuckWorkd-H, (c) Reacher, (d) Ant-Limit, (e) Ant-Block. The x-axis is environment steps.

Figure 4 shows the cumulative reward in five safety control tasks. The comparison result demonstrate that the sample efficiency of MBHI is better than baselines, especially DDPG. MBHI achieves

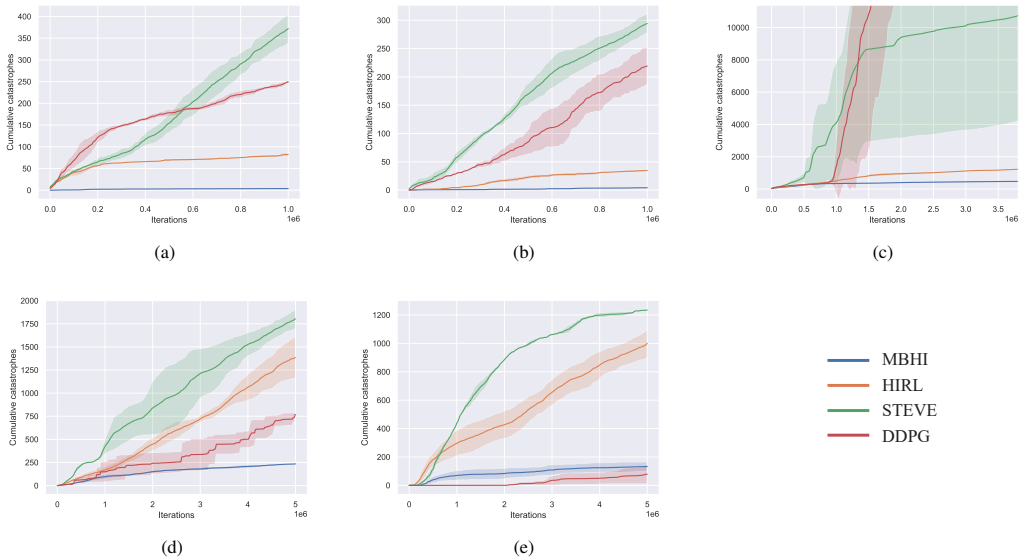


Figure 5: Cumulative catastrophes of MBHI and baselines on five continuous control tasks over time (mean and standard deviation). (a) PuckWorld-L, (b) PuckWorkd-H, (c) Reacher, (d) Ant-Limit, (e) Ant-Block. The x-axis is environment steps.

similar or better performances in all tasks than baselines. In PuckWorld-H, the parameters of MBHI and HIRL in safety inspection are exactly the same, but MBHI can converge faster, which illustrates the importance of active exploration. Frequent action shielding increases the complexity of policy learning. However, as an off-policy method, we do not find perceptible negative effects for MBHI.

The comparison result of cumulative catastrophes is shown in Figure 5. Experiment results of STEVE and DDPG demonstrate that optimal unsupervised behaviors can result in a large number of catastrophes. Note that DDPG has fewer catastrophes in some environments because the agent has not learned how to complete the task, and tends to move near the initial state. For example, in Ant-Block, the agent does not even successfully move to the place where it could encounter the obstacle. Additionally, HIRL is able to successfully resolve locally avoidable catastrophes, but failed in other environments. By contrast, MBHI effectively avoids catastrophes for both locally and non-locally avoidable situation during training. In the simple environment like PuckWorld, MBHI almost achieves zero catastrophes.

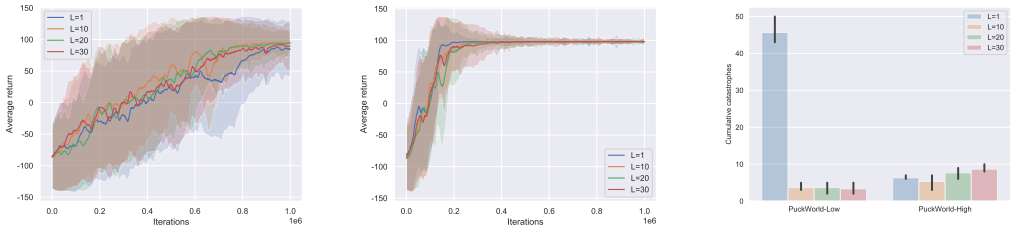
In tasks other than PuckWorld-H, HIRL cannot effectively save the agent. However, from Figure 5, we can see that the number of catastrophes in HIRL is significantly less than STEVE. This is because we divide the replay buffer in HIRL into safe and unsafe parts. Besides, we can also find that even though STEVE has converged, the number of catastrophes is still rising. Our findings are in line with recent results on catastrophic forgetting [12, 30]. MBHI can well restrain this problem.

5.3 Ablation on Safety Detection Rollout Length

Intuitively, the longer the safety detection distance, the more conservative the policy. We evaluate different safety detection rollout lengths on PuckWorld-L and PuckWorld-H. The result is shown in Figure 6. It shows that, surprisingly, when the safety detection distance is sufficient, our method is not so sensitive to the length of safety detection rollouts. This also proves that safety detection in the imagination can effectively avoid the catastrophe that will happen in advance. In addition, the long detection distance does not further reduce the cumulative catastrophes significantly, but it will introduce unnecessary calculation costs.

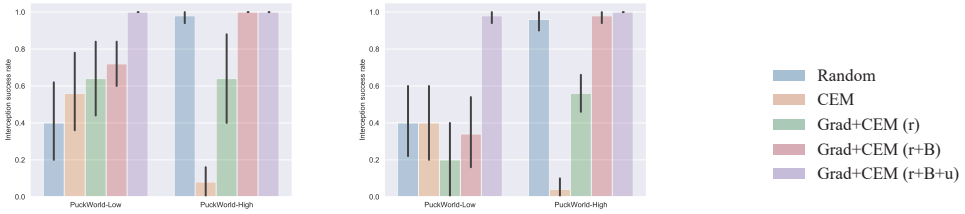
5.4 Comparison of Action Replacement Methods

We study the impact of different MPC optimization techniques and reward settings. The compared optimizers include the uniform random search [26] and the cross-entropy method [20]. The MPC



(a) (b) (c)

Figure 6: Ablation studies of the safety detection rollout length. (a) Average return in PuckWorld-L; (b) Average return in PuckWorld-H; (c) Cumulative catastrophes. Error bars are 95% bootstrap confidence intervals.



(a) (b)

Figure 7: Interception success rate of different methods. (a) Converged dynamics model, (b) pre-trained dynamics model. For fair comparison, random shooting sampled 3200 times, CEM has 25 iterations of sampling 128 candidate actions, and Grad+CEM has 5 iterations plus 5 gradient steps of sampling 128 candidate actions. Shown is the mean performance, with error bars showing 95% bootstrap confidence intervals over 5 runs.

reward settings include three types: 1) the reward $\hat{r}$ predicted by the learned reward function; 2) $\hat{r}$ and the catastrophe probability $\hat{B}$ predicted by the Blocker; 3) $\hat{r}$, $\hat{B}$, and the distribution distance u between sampling samples and unsafe samples. For the sake of fairness, we also use $\hat{r} + \hat{B} + u$ as the MPC reward in Random and CEM. We sample ten points uniformly around the catastrophe as the initial position, and make the agent move toward the catastrophe at the fastest speed. In addition, the converged dynamics model and the model only be pre-trained are used as the simulator of MPC to evaluate the robustness of the proposed method. The results of the interception success rate are shown in Figure 7. It shows that our method has better interception performance and is more robust.

6 Conclusion

In this paper, we proposed a state-based safety oversight mechanism MBHI, which is based on the model-based RL algorithm with active learning method. MBHI is forward-looking, it considers the safety of the current policy in a certain time-step in the future, replaces dangerous actions in advance, and realizes the prevention of both local and non-local catastrophes. We evaluated the proposed algorithm on five continuous safety control tasks, the experiment results demonstrate that MBHI can significantly reduce catastrophes and accelerate training. Furthermore, our solution acts directly at the policy level, so it is independent of the learning algorithm and can be plugged into any existing model-based and model-free continuous control algorithms.

MBHI effectively alleviates the problem of non-locally avoidable catastrophes, but it needs to have a certain understanding of the task. The safety detection distance needs to be long enough to avoid catastrophes in advance. Besides, hyperparameters are sensitive to the performance of the Blocker. Going forward, the safety detection length should be adaptive to the specific task, rather than set a fixed value. Moreover, in future research, we plan to study the theoretical guarantee of MBHI and improve the algorithm's robustness to the hyper-parameters. In terms of computation, since safety evaluation inevitably increases the cost of training process, we plan to study more effective methods to estimate the potential danger of the agent in the future and to integrate these estimates in model-based policy optimization.

References

[1] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney. Recurrent experience replay in distributed reinforcement learning. In *ICLR*, 2019.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[4] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[6] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[7] B. Lütjens, M. Everett, and J. How. Safe reinforcement learning with model uncertainty estimates. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8662–8668, 2019.

[8] J. Garcia and F. Fernández. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.*, 16:1437–1480, 2015.

[9] J. Fan and W. Li. Safety-guided deep reinforcement learning via online gaussian process estimation. *ArXiv*, abs/1903.02526, 2019.

[10] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31, 2017.

[11] A. Cowen-Rivers, D. Palenicek, V. Moens, M. Abdullah, A. Sootla, J. Wang, and H. Ammar. Samba: Safe model-based & active reinforcement learning. *ArXiv*, abs/2006.09436, 2020.

[12] W. Saunders, G. Sastry, A. Stuhlmüller, and O. Evans. Trial without error: Towards safe reinforcement learning via human intervention. In *AAMAS*, 2018.

[13] B. Prakash, M. Khatwani, N. R. Waytowich, and T. Mohsenin. Improving safety in reinforcement learning using model-based architectures and human intervention. In *FLAIRS Conference*, 2019.

[14] G. Dalal, K. Dvijotham, M. Vecerík, T. Hester, C. Paduraru, and Y. Tassa. Safe exploration in continuous action spaces. *ArXiv*, abs/1801.08757, 2018.

[15] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. Safe reinforcement learning via shielding. In *AAAI*, 2018.

[16] G. Kahn, A. Villaflor, V. H. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *ArXiv*, abs/1702.01182, 2017.

[17] A. PrashanthL. Policy gradients for cvar-constrained mdps. In *ALT*, 2014.

[18] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *ArXiv*, abs/1512.01629, 2017.

[19] T. Pham, G. Magistris, and R. Tachibana. Optlayer - practical constrained optimization for deep reinforcement learning in the real world. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6236–6243, 2018.

[20] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[21] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *NeurIPS*, 2018.

[22] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-based value estimation for efficient model-free reinforcement learning. *ArXiv*, abs/1803.00101, 2018.

[23] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *ICLR*, 2018.

[24] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019.

[25] G. Kalweit and J. Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *CoRL*, pages 195–206, 2017.

[26] A. Nagabandi, G. Kahn, R. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, 2018.

[27] F. Pan, J. He, D. Tu, and Q. He. Trust the model when it is confident: Masked model-based actor-critic. In *NeurIPS*, 2020.

[28] P. J. Ball, J. Parker-Holder, A. Pacchiano, K. Choromanski, and S. Roberts. Ready policy one: World building through active learning. In *ICML*, 2020.

[29] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[30] R. Kemker, A. Abitino, M. McClure, and C. Kanan. Measuring catastrophic forgetting in neural networks. In *AAAI*, 2018.

A Appendix: Algorithms

Algorithms 1 and 2 contain the pseudo-code of action replacement mechanism and complete optimization procedure in MBHI, respectively.

Algorithm 1 Action Replacement Mechanism in MBHI

Inputs: Initial state s_0, MPC horizon H, number of CEM iterations T, number of gradient steps I, transition ensemble $\hat{T}_{\boldsymbol{\xi}}$, reward ensemble $\hat{R}_{\boldsymbol{\phi}}$, Block ensemble $\hat{B}_{\boldsymbol{\mu}}$
for $t = 0$ to $T - 1$ **do:**
  Sample K action sequences $\{a^{(t)}_{0:H-1}\}_{k=1}^{K} \sim \text{CEM}(\cdot)$
  for $i = 1$ to I **do:**
    Calculate imaginary rollouts $\{s^{(t)}_{1:H}\}_k = \hat{T}_{\xi}(\{a^{(t)}_{0:H-1}\}_k, s_0)$
    Calculate MPC returns for each action sequence by Eq. (12)
    Update the action sequence $\{a^{(t)}_{0:H-1}\}_{k=1}^{K}$ via gradient ascent
  end
  Recalculate imaginary rollouts and MPC returns
  Update CEM$(\cdot)$ distribution with top N action sequences sorted by MPC returns
end
Output: The first action a^*_0 from optimal action sequence $a^*_{0:H-1}$

Algorithm 2 Actual algorithm of MBHI

Inputs: Number of initial samples N_0, interacting step T, length of safety detection H, safe threshold ε, scaling factors c_l and c_h, Block ensemble $\hat{B}_{\boldsymbol{\mu}}$
Initialize: random initialize policy π_θ, value ensemble $\hat{Q}_{\boldsymbol{\varphi}}$, transition ensemble $\hat{T}_{\boldsymbol{\xi}}$, reward ensemble $\hat{R}_{\boldsymbol{\phi}}$, termination model $\hat{D}_{\boldsymbol{\psi}}$
Collect N_0 samples with random policy, initialize replay buffer $D = \{(s_t, a_t, r_t, s_{t+1}, d_t)\}_{t=1}^{N_0}$ and pre-train $\{\hat{T}_{\boldsymbol{\xi}}, \hat{R}_{\boldsymbol{\phi}}, \hat{D}_{\boldsymbol{\psi}}\}$ N_0 times
for each epoch **do:**
  Get initial state s_0
  for $t = 1$ to T **do:**
    Check the safety p_t of the current policy π_θ with Eq. (10) in imaginary rollouts of depth H starting from state s_t
    if $p_t \geq \varepsilon$ **then** Compute action a_t using Algorithm 1
    else Compute $a_t \sim \pi_\theta(a_t|s_t)$ with policy network
    Execute a_t to the real world and add transition to D
    Train the dynamics model $\{\hat{T}_{\boldsymbol{\xi}}, \hat{R}_{\boldsymbol{\phi}}, \hat{D}_{\boldsymbol{\psi}}\}$ on buffer D with Eq. (2)
    Update π_θ and $\hat{Q}_{\boldsymbol{\varphi}}$ with model-based learning methods on buffer D with Eq. (14)
  end
end

B Appendix: Experiment Setup

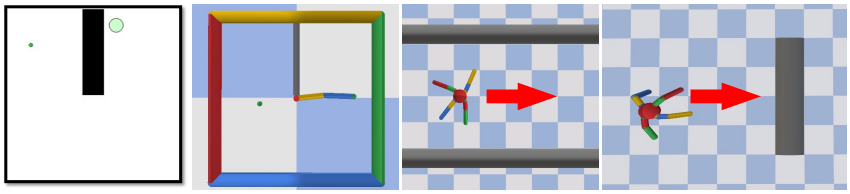


Figure 8: Visualization of the experimental environments. From left to right: PuckWorld, Reacher, Ant-Limit and Ant-Block.

- **PuckWorld-L:** A randomly generated puck has to reach a given random target in two-dimensional space. The maximum speed of the puck is 0.025, and the maximum acceleration is 0.002. Therefore, it needs to decelerate multiple steps in advance before the puck can stop. The constraint is that the puck cannot enter the black area in the middle.
- **PuckWorld-H:** Same as PuckWorld-L, except that the maximum acceleration is 0.025. It means that the puck can be stopped immediately.
- **Reacher:** A two-link arm has to reach a given random target in two-dimensional space. The arm is not allowed to touch the vertical bar.
- **Ant-Limit:** An ant with four legs needs to run as fast as possible along the positive X-axis in three-dimensional space. The ant can only move between two parallel barriers.
- **Ant-Block:** The basic environment is the same as Ant-Limit. The difference is that the ant is not allowed to touch the square obstacle.

Note that the episode will be terminated immediately and receive a large negative reward if the agent enters the catastrophe zone.

C Appendix: Implementation Details

In this work, we follow the same network structure and policy learning method as STEVE. All models are fully connect neural networks optimized by Adam with learning rate of 3e-4. The value network, policy network, reward model and termination model each has 4 layers of size 128. The transition model has 8 layers of size 512.

The first 1e5 frames are sampled by the agent interacting with the environment through random actions. After that, the dynamics model is pre-trained 1e5 times, and then 1 model update and 1 policy update are performed for each frame sampled from the environment. The mini-batch size of all models is 512. We use soft update for the target network instead of hard update used in STEVE. Note that the replay buffer is divided into two parts, safe and unsafe, and sampled from them in equal proportions. This prioritization can introduce bias, which we correct with importance sampling method.

Table 1: Experiment hyper-parameters.

Hyperparameter Name	PuckWorld-Low	PuckWorld-High	Reacher	Ant-Limit	Ant-Block
Safety detection length	10	1	8	10	10
MPC horizon	10	10	10	10	10
Safety weight	1	1	1	1	1
Active learning coefficient	5e4	10	1e3	10	10
Safe threshold	0.96	0.96	0.96	0.99	0.99
Intrinsic reward bound	0.92	0.92	0.92	0.95	0.95
Scaling factor c_l	1	1	1	0.5	0.5
Scaling factor c_h	100	100	100	10	10

D Appendix: Visualization of Motion Sequences

The visualization of motion sequences is shown in Figure 9, MBHI executes the current policy in the imagination to decide whether to block the action.

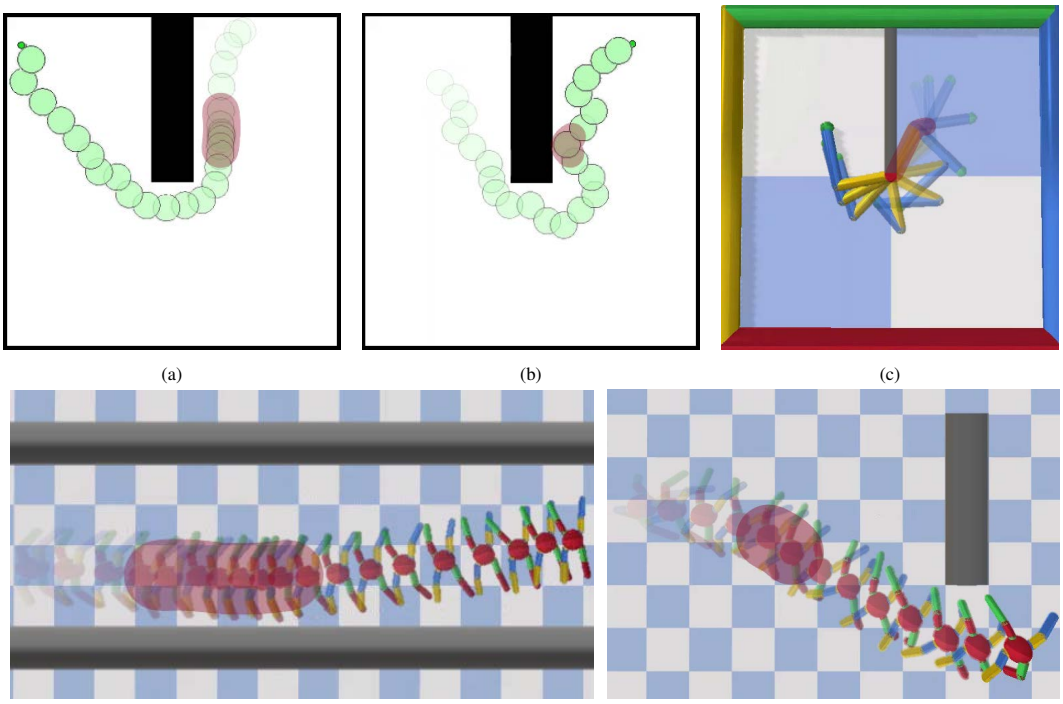


Figure 9: The Visualization of motion sequences in experiments. (a) PuckWorld-L, (b) PuckWorkd-H, (c) Reacher, (d) Ant-Limit, (e) Ant-Block. The red area indicates that the agent is blocked and the action is replaced by the MPC controller.