Supplementary Materials

Appendix A. Proof for Theo. 6

From $V_i(x_i, p_i) \leq c_i(p_i)$ we know that the entering state x_i is within the maximum ϵ-stable level set of equilibrium point x^*, hence the entering state x_i is within the ϵ-RoA of mode i. Next, we show that the next entering state $x_j = h_i(\bar{x}_i, u; p_i, p_j)$ is also within the ϵ-RoA of mode j.

Since x_i is within ϵ-RoA of mode i, we know $||\bar{x}_i - x^*|| \leq \epsilon$. Then from $\alpha_j||x - x^*|| \leq V_j(x, p_j) \leq \beta_j||x - x^*||$ we have

$$
\begin{aligned}
V_j(x_j, p_j) &= V_j(h_i(\bar{x}_i, u; p_i, p_j), p_j) \\
&\quad \text{(Definitions of jump maps and entering/exiting states)} \\
&\quad \leq \beta_j||h_i(\bar{x}_i, u; p_i, p_j)|| \\
&\quad \text{(Lyapunov bounding condition)} \\
&\quad \leq \beta_j||h_i(x^*, u; p_i, p_j)|| + \beta_j K_i||\bar{x}_i - x^*|| \\
&\quad \text{(Local Lipschitz condition for } h_i \text{ at } x^*) \\
&\quad \leq \beta_j||h_i(x^*, u; p_i, p_j)|| + \beta_j K_i \epsilon \\
&\quad \text{(Definition of } \epsilon\text{-RoA for mode } i) \\
&\quad \leq \frac{\beta_j}{\alpha_j} V_j(h(x_i^*, u; p_i, p_j), p_j) + \beta_j K_i \epsilon \\
&\quad \text{(Lyapunov function bounding condition)} \\
&\quad \leq \frac{\beta_j}{\alpha_j}\Big(\frac{\alpha_j}{\beta_j} c_j(p_j) - \alpha_j K_i \epsilon\Big) + \beta_j K_i \epsilon \\
&\quad \text{(The condition in Eq. 2)} \\
&\quad \leq c_j(p_j)
\end{aligned}
\tag{8}
$$

therefore we derive that $V_j(x_j, p_j) \leq c_j(p_j)$, which means x_j is within the ϵ-RoA for mode j. So the whole hybrid system is ϵ-stable according to Def. 3.

Appendix B. Proof for Theo. 7

We consider the lower bound of $||p_i - p_k||$ for every jump. We know that the Lyapunov value at the entering state of mode k (denote the switching $i \to k$) is:

$$
\begin{aligned}
&V_k(h_i(x_i, u; p_i, p_k), p_k) \\
&= V_k(h^+(x_i, p_i) + p_i - p_k, u; p_k) \\
&\text{(definition of the special system)} \\
&\leq \beta_k ||h^+(x_i, p_i) + p_i - p_k|| \\
&\text{(Lyapunov bounding condition)} \\
&\leq \beta_k ||h^+(x_i, p_i)|| + \beta_k ||p_i - p_k|| \\
&\text{(Triangle inequality)} \\
&\leq \beta_k ||h^+(x^*, p_i)|| + K_m \beta_k ||x_i - x^*|| + \beta_k ||p_i - p_k|| \\
&\text{(Local Lipschitz condition)} \\
&= K_m \beta_k ||x_i - x^*|| + \beta_k ||p_i - p_k|| \\
&\text{(Since } h^+(x^*, p_i) = x^*) \\
&\leq \beta_k K_m \epsilon + \beta_k ||p_i - p_k|| \\
&\text{(Definition of } \epsilon\text{-RoA)}
\end{aligned}
\tag{9}
$$

If the optimization is feasible and the optimal p exists, then from the assumption we know that $V_k(h^+(x_i, p_i) + p_i - p_k, p_k)$ for the optimal p must be no larger than $c_k(p_k)$ (zero the first loss in Eq. 7). We are going to show that $V_k(h^+(x_i, p_i) + p_i - p_k, p_k)$ must strictly equal to $c_k(p_k)$. If not, based on the continuity of the V_k, there must exist a $\tilde{p}_k$ around p_k that can also zero the first loss term in Eq. 7, and make $||\tilde{p}_k - p_j|| \leq ||p_k - p_j||$ which brings contradiction. Thus we have $V_k(h^+(x_i, p_i) + p_i - p_k, p_k) = c_k(p_k)$, hence based on Eq. 9, we have:

$$
||p_i - p_k|| \geq \frac{c_k(p_k)}{\beta_k} - K_m \epsilon \tag{10}
$$

For each jump, the step length is lower bounded as shown in Eq. 10. Thus we have the number of jumps is:

$$
N \leq \left\lceil \frac{||p_j - p_i||}{\min\limits_m \frac{c_m}{\beta_m} - K_m \epsilon} \right\rceil \tag{11}
$$

Appendix C. Details for the simulation environments

C.1. Car tracking control

The goal here is to make sure the car can drive on the road under different road conditions. Given a reference state $(x, y, v, \psi)^T$ for a Dubins car model, the state of the car model is $(x_e, y_e, \delta, v_e, \psi_e, \dot{\psi}_e, \beta)^T$, where x_e, y_e represent the Cartesian error, δ denotes the steering angle, v_e denotes the velocity error, ψ_e and $\dot{\psi}_e$ are the heading angle error and angular velocity error, and β is the slip angle. The

dynamics are given by $\dot{x} = f(x) + g(x)u$, with

$$f(x) = \begin{pmatrix} v\cos(\psi_e) - v_{ref} + \omega_{ref} y_e \\ v\sin(\psi_e) - \omega_{ref} x_e \\ 0 \\ 0 \\ \dot{\psi}_e \\ C_1(\dot{\psi}_e + \omega_{ref}) + C_2\beta + C_3\delta \\ C_4(\dot{\psi}_e - \omega_{ref}) + C_5\beta + C_6\delta \end{pmatrix} \tag{12}$$

with

$$\begin{cases} C_1 = -\frac{\mu m}{vI_x(l_r+l_f)}(l_f^2 C_{Sf} g l_r + l_r^2 C_{Sr} g l_f) \\ C_2 = \frac{\mu m}{I_x(l_r+l_f)}(l_r C_{Sr} g l_f - l_f C_{Sf} g l_r) \\ C_3 = \frac{\mu m}{I_x(l_r+l_f)}(l_f C_{Sf} g l_r) \\ C_4 = \frac{\mu}{v^2(l_r+l_f)}(C_{Sr} g l_f l_r - C_{Sf} g l_r l_f) - 1 \\ C_5 = -\frac{\mu}{v(l_r+l_f)}(C_{Sr} g l_f + C_{Sf} g l_r) \\ C_6 = \frac{\mu}{v(l_r+l_f)}(C_{Sf} g l_r) \end{cases} \tag{13}$$

and

$$g(x) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{14}$$

where u is the acceleration and the steering angle output, $I_x, l_r, l_f, C_{Sf}, C_{Sr}, g$ are constant factors, and μ is the road friction factor. More details can be found in Althoff et al. (2017).

The road consists of multiple segments with different road conditions (different μ). Each segment belongs to a system mode with the configuration of reference waypoint (X^E, Y^E), reference velocity v^E and the friction factor μ. Different combinations of friction factor and the velocity will give different traction force for the vehicle. At junctions of the two segments, the mode switching causes the system state jump because of the change of the reference waypoint.

C.2. Pogobot navigation

The state of the pogobot is $s = (x, \dot{x}, y, \dot{y})^T$, where x, y are the 2d coordinate of the pogobot head, and the $\dot{x}, \dot{y}$ are the corresponding velocities. The movement of a pogobot involves two phases. In the flight phase, the pogobot follows a ballistic dynamics $\dot{s} = f(s)$ where:

$$f(s) = \begin{pmatrix} \dot{x} \\ 0 \\ \dot{y} \\ -g \end{pmatrix} \tag{15}$$

here g is the gravity and the stance foot is determined by the pogobot pose θ (which can be controlled instantly, since we assume a mass-less leg). In the stance phase, together with the stance foot

position $(x_f, y_f)^T$, the dynamics becomes

$$\dot{s} = \begin{bmatrix} \dot{x} \\ \frac{x-x_f}{L}\left(k\left(L-l_0\right)+F\right) \\ \dot{y} \\ \frac{y-y_f}{L}\left(k\left(L-l_0\right)+F\right)-g \end{bmatrix} \tag{16}$$

where $L = \sqrt{(x-x_f)^2+(y-y_f)^2}$ denotes the length of the current pogobot, l_0 denotes the original length of the pogobot, k denotes the spring constant factor, and F and θ are the control inputs (the stance force and the swing leg angle). Here we consider the apex-to-apex control strategy. We first collect the simulation data to learn an apex-to-apex dynamic estimator. Then we use this dynamic estimator to train our Lyapunov function and controllers (as well as the RL-based methods).

C.3. Bipedal walker locomotion

The state of the Bipedal walker is $s = (q, \dot{q})^T$ where $q = (q_1, q_2)^T$, and q_1 is the angle between the normal vector of the ground and the stance leg, and the q_2 is the angle between the stance leg and the swing leg. $\dot{q}_1$ and $\dot{q}_2$ are the corresponding angular velocities. Within each mode, the continuous dynamics of the system follows the manipulator equation:

$$\dot{s} = \begin{bmatrix} \dot{q} \\ D_s^{-1}(q)[-C_s(q,\dot{q}) - G_s(q) + B_s(q)u] \end{bmatrix} \tag{17}$$

where D_s, C_s, G_s, B_s are functions of q (and $\dot{q}$), u is the control input (torque in this case), and a state jump will occur when $q_2 + 2q_1 = 0$. Here $B = (1,0)^T$. We denote the leg mass m, the original leg length l with center of mass (CoM) location l_c, the acceleration due to gravity g_0, and the leg inertial about leg CoM as I. Then the $D_s(q)$ can be written as:

$$\begin{cases} (D_s(q))_{1,1} = (l-l_c)^2 m + I \\ (D_s(q))_{1,2} = ml(l-l_c)\cos(q_2) - (l-l_c)^2 m - I \\ (D_s(q))_{2,2} = -2ml(l-l_c)\cos(q_2) + \\ \qquad\qquad (2(l_c^2+l^2) - 2l_c l)m + 2I \end{cases} \tag{18}$$

and the nonzero entries in C_s are:

$$\begin{cases} (C_s(q,\dot{q}))_{1,2} = -ml\sin(q_2)(l-l_c)\dot{q}_1 \\ (C_s(q_2,\dot{q}_1))_{2,1} = -ml\sin(q_2)(l-l_c)(\dot{q}_2 - \dot{q}_1) \\ (C_s(q_2,\dot{q}_1))_{2,2} = -ml\sin(q_2)(l-l_c)\dot{q}_2 \end{cases} \tag{19}$$

and the nonzero entries in G_s are:

$$\begin{cases} (G_s(q_1,q_2))_1 = mg_0\sin(q_2-q_1)(l-l_c) \\ (G_s(q_1,q_2))_2 = mg_0((l_c-l)\sin(q2-q1) - \sin(q_1)(l_c+l)) \end{cases} \tag{20}$$

We recommend readers to Choi et al. (2022) for more details.

Appendix D. Implementation of our approach

As mentioned in Algorithm. 1, we first learn the control Lyapunov function (CLF) and the NN controller, then we estimate the RoA, and finally we conduct online optimization for the planner in the deployment phase.

For the CLF and controller learning phase (take the car tracking experiment as an example), we uniformly sample 1000 states from an initial set, then at every epoch we sample trajectories for 100 time steps from the corresponding environment simulator given the NN controller, We use the trajectories to train our CLF and NN controller. The CLF and the NN controller are 2-hidden-layer NNs with 256 hidden units in each layer and ReLU activation in the intermediate layers. The last layer of the controller uses TanH activation function to clip the output signal in a reasonable range. We train the CLF and NN controller via the loss in Eq. 3. We use RMSprop gradient descent method for the optimization with the learning rate of 10^{-4}. We train CLF and controller for (at maximum) 1000 epochs, where inside each epoch the CLF and the controller will be updated for 500 steps. We stop the training when the validation loss is not dropping significantly.

For the RoA Estimation phase, we use the CLF and the NN controller trained in the previous step to generate $10^3 \sim 10^4$ trajectories in 100 time steps. Using CLF, we are able to find the largest Lyapunov value c_i^* for all the sampled initial states that having the exiting states within the ϵ-ball of the equilibrium. We set the $\epsilon = 10^{-2}$. Then we train the RoA estimator using the loss in Eq. 4 for 50000 iterations, with RMSprop optimizer and learning rate of 10^{-4}.

For the online planning (deployment) phase, we use the differentiable planner to plan for valid configurations, and use the controller to "follow" that configuration to maintain stability. For the differentiable planner, at every mode switching instant, we first randomly generate 1000 configuration hypothesis. Then we use RMSprop optimizer and conduct gradient descent with learning rate of 0.05 for 5 steps. Then we pick the updated configuration hypothesis with the lowest loss.

D.1. Car tracking control

Before entering the i-th segment, we optimize for the configuration p_i, which is the waypoint $W_i = (x_i^{ref}, y_i^{ref})$ at the junction between the i-th segment and the $i+1$-th segment, and the reference velocity v_i^{ref} to track on the i-th segment. And at the i-th segment, we use the environment reference waypoint (x_{i+1}^E, x_{i+1}^E) and the reference velocity v_{i+1}^E for the next configuration. We make sure: (1) the current entering state x_i is within the RoA of the current system under the configuration of $p_i = (W_i, v_i^{ref})$. (2) the next entering state x_{i+1} is within the RoA of the system at segment $i+1$ with configuration p_{i+1}.

D.2. Pogobot navigation

Before entering the i-th segment, we optimize for the configuration p_i, which is the reference apex state height and reference apex state horizontal velocity at the next cycle (during the i-the segment). We can find the last apex state $\tilde{x}_i$ before exiting the i-the segment using the dynamics estimator, and make sure $\tilde{x}_i$ is within the RoA for the $i+1$-th segment under the reference apex state X_{i+1}^E given from the environment.

D.3. Bipedal walker locomotion

In this case, due to the difficulty to synthesize a control Lyapunov function, for the low-level controller, we directly use the QP controller derived from Choi et al. (2022) and the corresponding Lyapuonv function is replaced by an RoA classifier, which outputs value <0.9 if the entering state is within the RoA, and outputs value >1.1 for the rest case. During the planning, we use the differentiable planner but with the loss Eq. 7 in to find the optimal configuration (in this case, the reference gait).

Appendix E. Implementation of baseline approaches

Reinforcement learning approaches: We modify the RL implementation code from https://github.com/RITCHIEHuang/DeepRL_Algorithms, created the RL environments for each experiment, train each method with 3 random seeds for 24 hours each and take the average performance in the testing. For the car experiment, we use the reward as a combination of Root Mean Square Error (RMSE) penalty with the reference state and the valid rate of the trajectory segments. For the pogobot experiment, we use the RMSE, the collision rate with the ceiling/floor, and the distance to the goal as the rewards/penalties. For the Bipedal walker, we use the L2-distance from the current state to the reference state on the target gait as the penalty to guide the controller to converge to the target gait.

Model predictive control approaches: We use the CasADi Andersson et al. (2019) to implement non-linear optimization for each tasks. In each case, the system is simulated under some parameters (controls) and the cost function is computed to optimize the control input. For the car experiment, the cost function is the tracking error within the prediction horizon (T=20). For the pogobot experiment, the cost function is a penalty term with collisions and a tracking error term to the horizontal reference velocity. For the bipedal walker experiment, the cost function is the L2-norm between the leg angle q_1 after the switching and the target gait leg angle q_1^{ref}.

Linear quadratic control approaches: At each segment, we require the car to track the segment endpoint and the designed reference velocity on the current segment given the friction factors. We compute the error dynamics, and synthesize the controller by solving the Algebraic Ricatti Equation similar as in Dawson et al. (2022b).

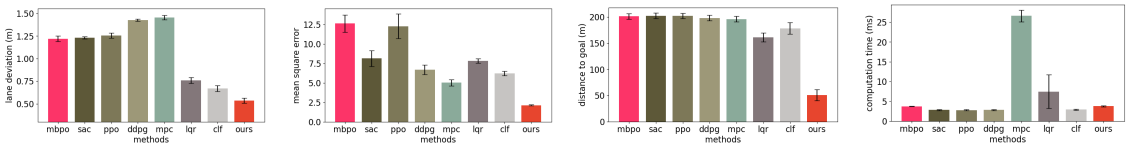
Control Lyapunov function approaches: Followed by Chang et al. (2019), we jointly train a single NN Lyapunov function for all the system modes with a NN controller (that can also take modes as inputs), with the same amount of training time used as for our approach.

CLF-QP approach: We directly use the QP controller derived from Choi et al. (2022) for the Bipedal walker comparison.

Hamilton Jacobian based approach: We directly use the computed result (the value function) from Choi et al. (2022) for the comparison for the target gait with the leg angle $q_1 = 0.13$ rad.

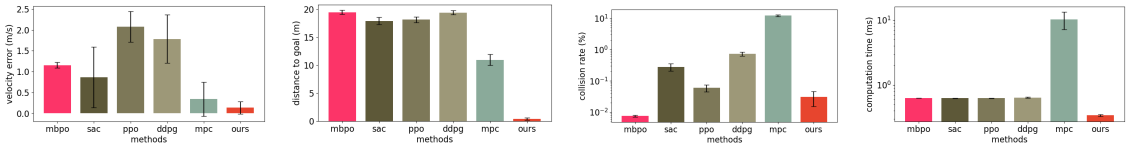
Appendix F. Comparison to Model-base RL method

We compare with the Model-based Policy Optimization (MBPO) method (Janner et al., 2019) implemented by an open-sourced library (Pineda et al., 2021). Based on the hyper-parameters provided in the library configuration files, we then searched for 3$\sim$ 10 different hyper-parameters (learning rates, policy update frequency, etc) and picked the one with the highest task-performance. We finalize the hyper-parameters and train the MBPO policy under 3 different random seeds for 24 hours for each experiment.



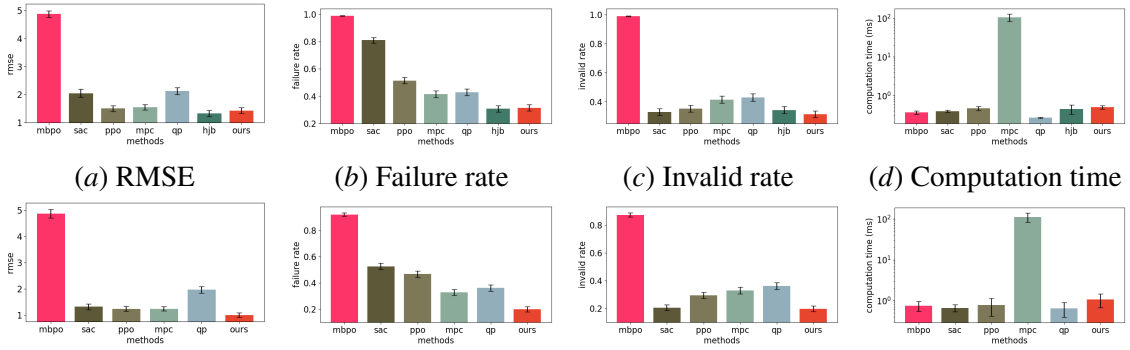
(*a*) Lane deviation (*b*) Mean square error (*c*) Distance to goal (*d*) Computation time

Figure 1: Quantitative result (including model-base RL method) for car experiment



(*a*) Velocity error (*b*) Distance to goal (*c*) Collision rate (*d*) Runtime

Figure 2: Quantitative result (including model-base RL method) for pogobot experiment



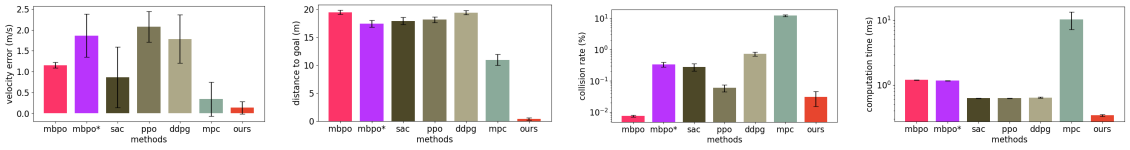
(*a*) RMSE (*b*) Failure rate (*c*) Invalid rate (*d*) Computation time

(*e*) RMSE (*f*) Failure rate (*g*) Invalid rate (*h*) Computation time

Figure 3: Bipedal walker comparison (including model-base RL method) under same ((a)$\sim$(d)) and different target gaits ((e)$\sim$(h)).

As shown in Fig. 1, Fig. 2 and Fig. 3, the MBPO result is on par of other RL baselines. MBPO achieves the lowest collision rate in the pogobot experiment, which is because it jumps out of the valid region before resulting in collision (which can be inferred by the high distance-to-goal measure for MBPO).

To figure out how well, under the pogobot experiment, we further train the MBPO with a pre-trained dynamic model derived from our method, and continue the training for 12 hours. As shown in Fig. 4, though the distance-to-goal performance is slightly better, still far from the result our method can achieve.



(*a*) Velocity error (*b*) Distance to goal (*c*) Collision rate (*d*) Runtime

Figure 4: Quantitative result (including model-base RL method with pretrained dynamic model) for pogobot experiment

Appendix G. Ablation studies for our method in the car experiment

η	κ	Lane deviation	RMSE	Distance to goal
1.2	0.0	2.096	0.518	0.117
1.0	0.0	2.114	0.533	0.166
0.9	0.0	2.107	0.519	0.166
0.8	0.0	2.087	0.519	0.117
0.5	0.0	2.103	0.546	0.117

Table 1: How different η will affect the performance in car experiment.

η	κ	Lane deviation	RMSE	Distance to goal
1.0	0.0	2.114	0.533	0.166
1.0	10^{-3}	2.116	0.530	0.166
1.0	10^{-2}	2.100	0.523	0.166
1.0	10^{-1}	2.110	0.525	0.167
1.0	10^{0}	2.102	0.513	0.117

Table 2: How different κ will affect the performance in car experiment.

Simulation Δt	Lane deviation	RMSE	Distance to goal
0.01	2.114	0.533	0.166
0.005	2.125	0.522	0.216
0.0025	2.118	0.528	0.216
0.00125	2.106	0.531	0.166
0.000625	2.104	0.531	0.166

Table 3: How different simulation Δt will affect the performance in car experiment.

Appendix H. Success rate for Bipedal walker locomotion under different initial conditions

For the bipedal walker experiment, we compare our approach with multiple RL-based approaches (A2C, DDPG, PPO, SAC, TD3, TRPO, VPG) and classic methods (QP, HJB) under different initial gait angles. As shown in Fig. 5, our approach can achieve similar-to-HJB performance, outperforming all the RL-baselines and the QP baseline. The largest improvement (comparing to RL methods) is from the "small initial angles". And our gain compared to QP-based methods is mostly from the "large initial angles", which might because the large deviation from the target gait angle makes the linearization more inaccurate, hence the QP-based method cannot achieve good performance.

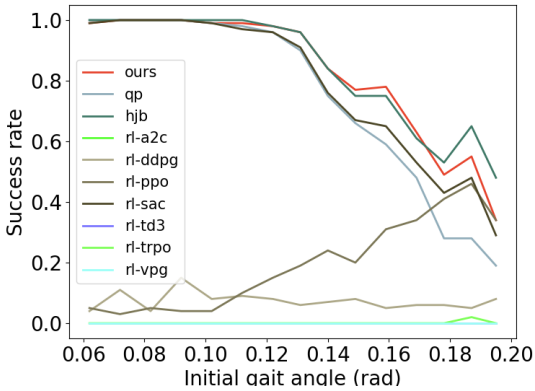
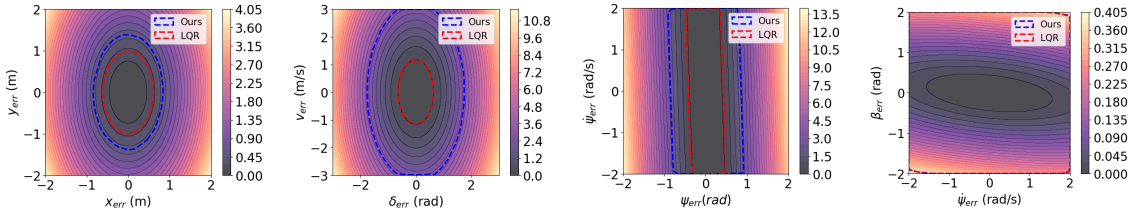


Figure 5: Walker success rate comparison under different initial states

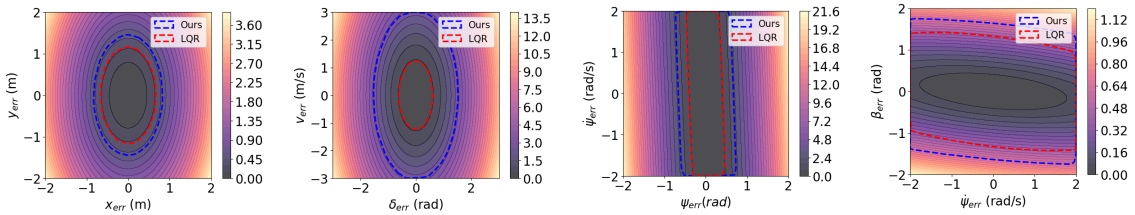
Appendix I. Visualization of learned RoA

From Fig. 6 to Fig. 30, we show the visualization of the learned RoA in all three experiments under different configurations.



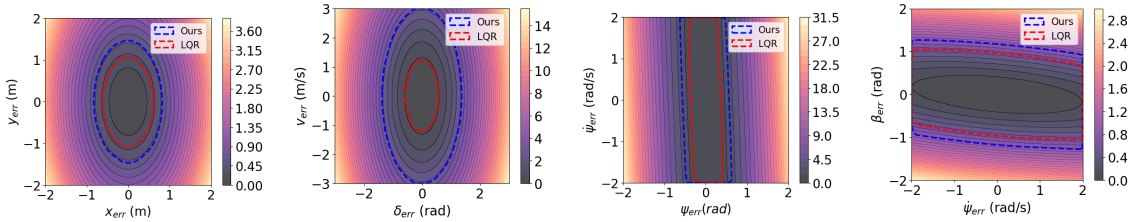
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 6: Car experiment (friction $\mu = 1.0$, reference speed $v = 5m/s$)



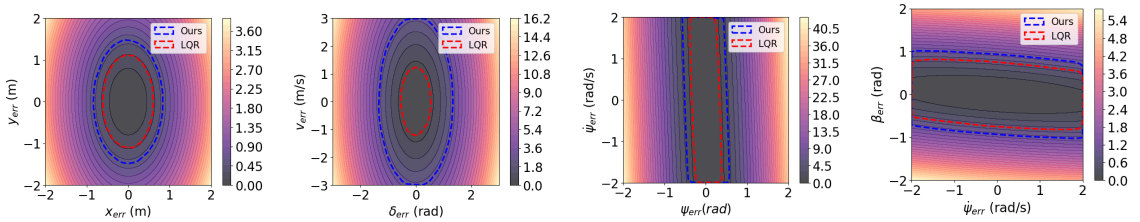
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 7: Car experiment (friction $\mu = 0.1$, reference speed $v = 5m/s$)



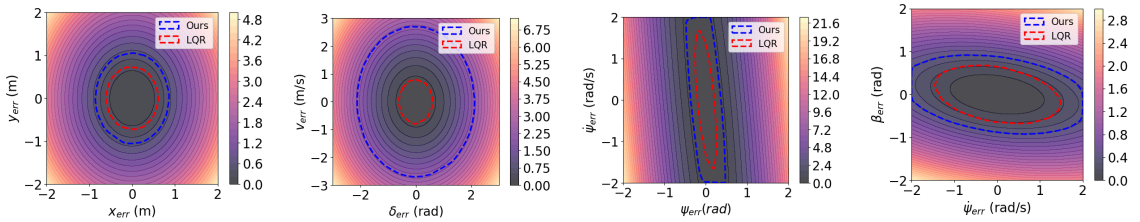
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 8: Car experiment (friction $\mu = 1.0$, reference speed $v = 10m/s$)



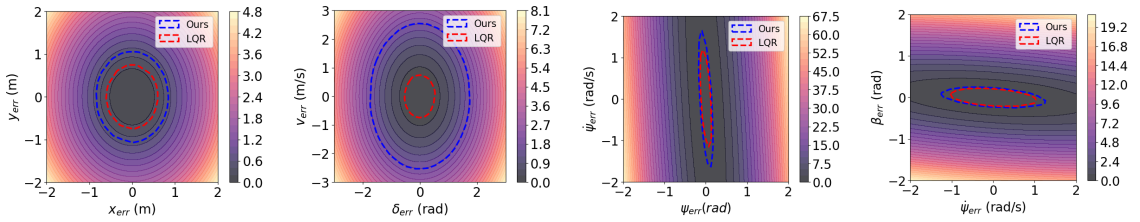
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 9: Car experiment (friction $\mu = 0.1$, reference speed $v = 10m/s$)



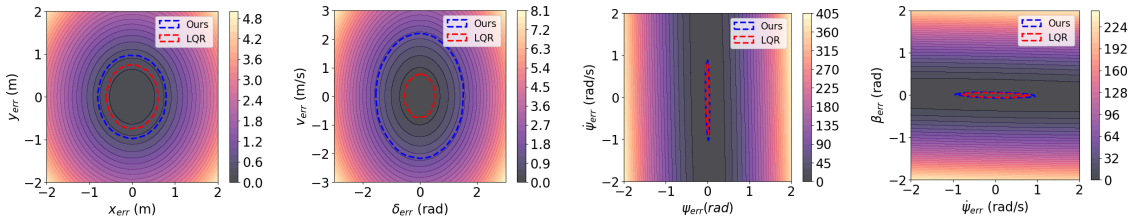
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 10: Car experiment (friction $\mu = 1.0$, reference speed $v = 20m/s$)



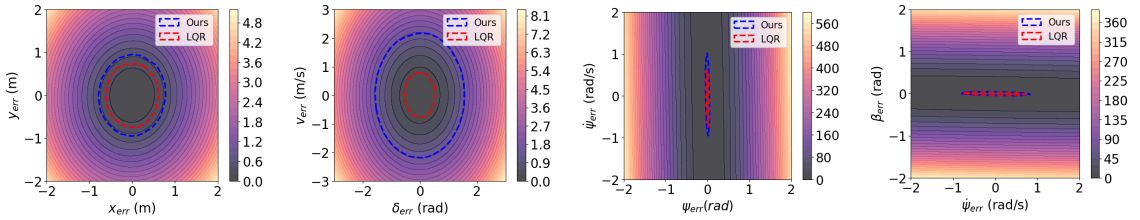
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 11: Car experiment (friction $\mu = 0.1$, reference speed $v = 20m/s$)



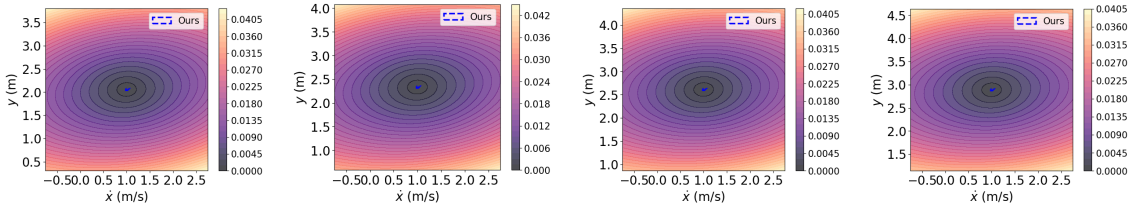
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 12: Car experiment (friction $\mu = 1.0$, reference speed $v = 30m/s$)



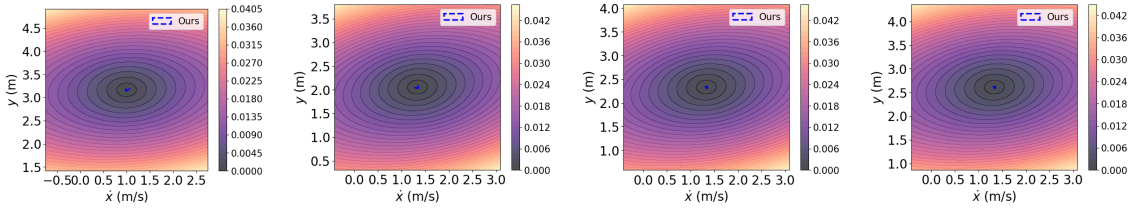
(*a*) x-y plane (*b*) v-δ plane (*c*) ψ-$\dot{\psi}$ plane (*d*) $\dot{\psi}$-β plane

Figure 13: Car experiment (friction $\mu = 0.1$, reference speed $v = 30m/s$)



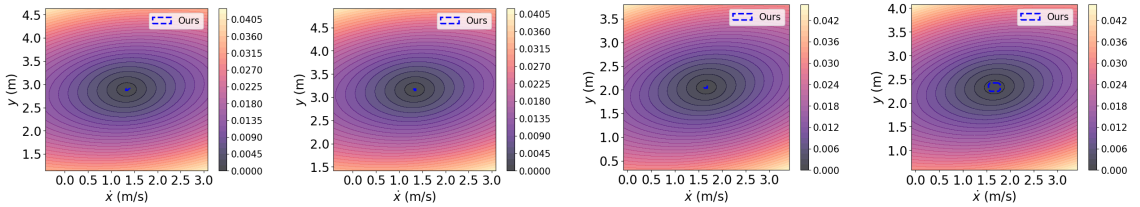
(*a*) apex velocity=1.0 m/s, apex height= 2.1 m (*b*) apex velocity=1.0 m/s, apex height= 2.3 m (*c*) apex velocity=1.0 m/s, apex height= 2.6 m (*d*) apex velocity=1.0 m/s, apex height= 2.9 m

Figure 14: Pogobot experiment



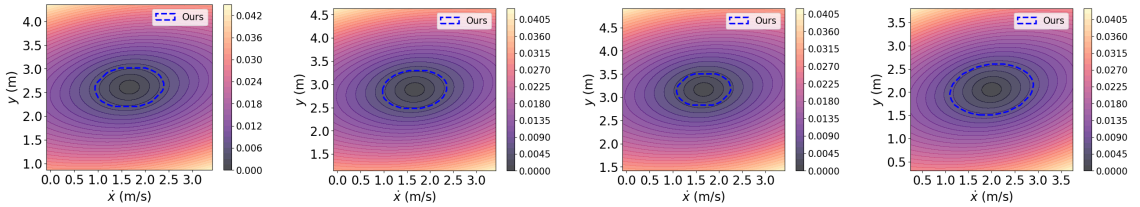
(*a*) apex velocity=1.0 m/s, apex height= 3.2 m (*b*) apex velocity=1.3 m/s, apex height= 2.1 m (*c*) apex velocity=1.3 m/s, apex height= 2.3 m (*d*) apex velocity=1.3 m/s, apex height= 2.6 m

Figure 15: Pogobot experiment



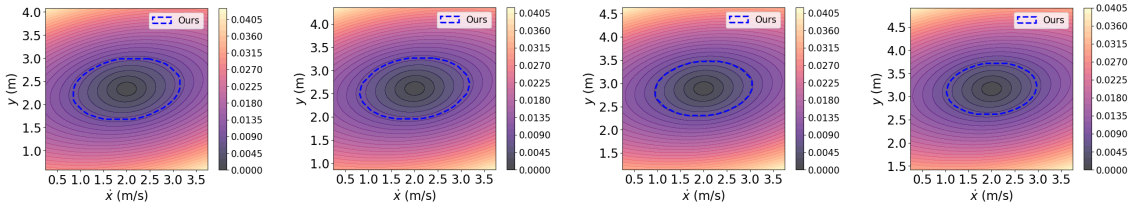
(*a*) apex velocity=1.3 m/s, apex height= 2.9 m (*b*) apex velocity=1.3 m/s, apex height= 3.2 m (*c*) apex velocity=1.7 m/s, apex height= 2.1 m (*d*) apex velocity=1.7 m/s, apex height= 2.3 m

Figure 16: Pogobot experiment



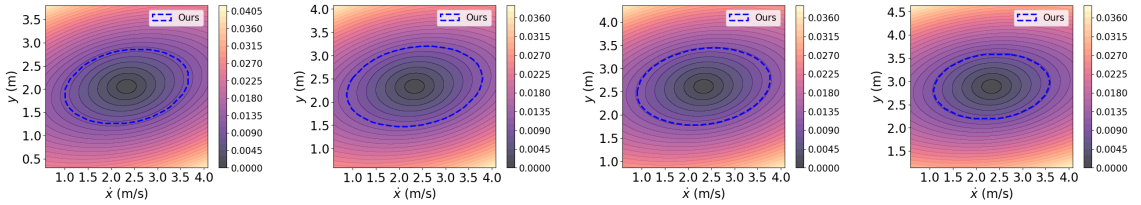
(*a*) apex velocity=1.7 m/s, apex height= 2.6 m (*b*) apex velocity=1.7 m/s, apex height= 2.9 m (*c*) apex velocity=1.7 m/s, apex height= 3.2 m (*d*) apex velocity=2.0 m/s, apex height= 2.1 m

Figure 17: Pogobot experiment



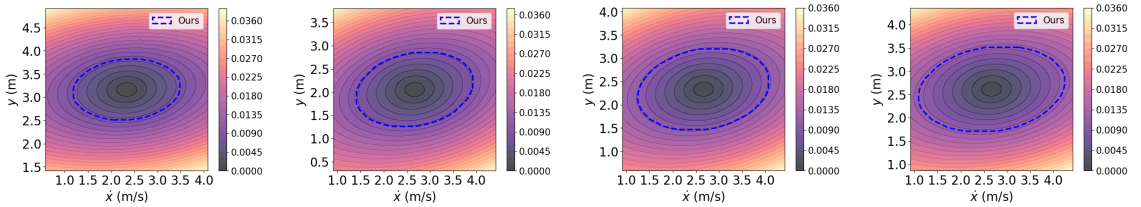
(*a*) apex velocity=2.0 m/s, apex height= 2.3 m (*b*) apex velocity=2.0 m/s, apex height= 2.6 m (*c*) apex velocity=2.0 m/s, apex height= 2.9 m (*d*) apex velocity=2.0 m/s, apex height= 3.2 m

Figure 18: Pogobot experiment



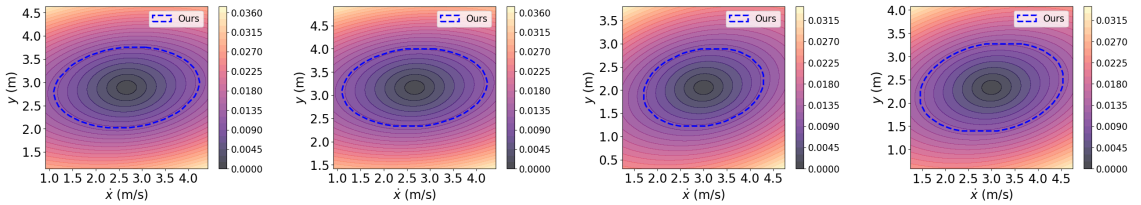
(*a*) apex velocity=2.3 m/s, apex height= 2.1 m (*b*) apex velocity=2.3 m/s, apex height= 2.3 m (*c*) apex velocity=2.3 m/s, apex height= 2.6 m (*d*) apex velocity=2.3 m/s, apex height= 2.9 m

Figure 19: Pogobot experiment



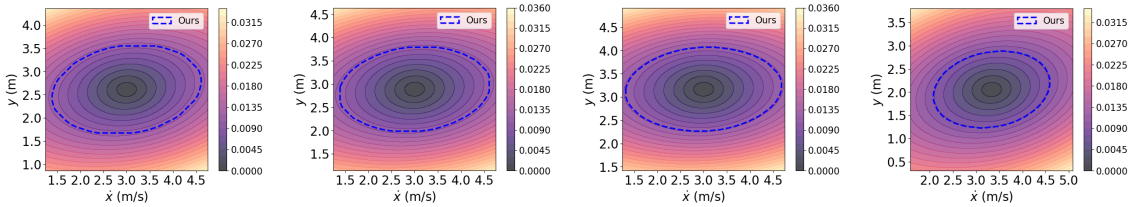
(*a*) apex velocity=2.3 m/s, apex height= 3.2 m (*b*) apex velocity=2.7 m/s, apex height= 2.1 m (*c*) apex velocity=2.7 m/s, apex height= 2.3 m (*d*) apex velocity=2.7 m/s, apex height= 2.6 m

Figure 20: Pogobot experiment



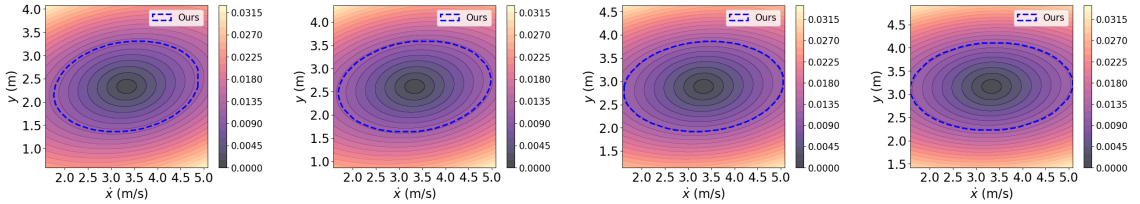
(*a*) apex velocity=2.7 m/s, apex height= 2.9 m (*b*) apex velocity=2.7 m/s, apex height= 3.2 m (*c*) apex velocity=3.0 m/s, apex height= 2.1 m (*d*) apex velocity=3.0 m/s, apex height= 2.3 m

Figure 21: Pogobot experiment



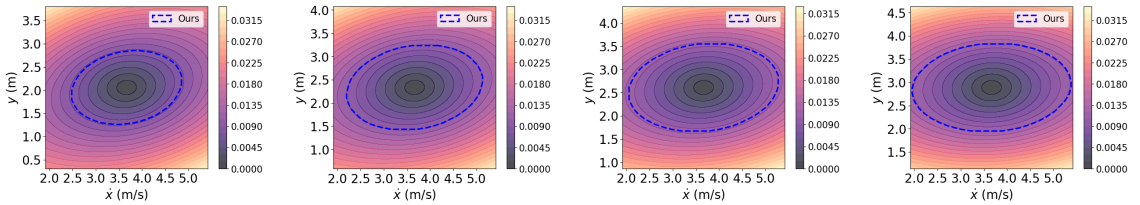
(*a*) apex velocity=3.0 m/s, apex height= 2.6 m (*b*) apex velocity=3.0 m/s, apex height= 2.9 m (*c*) apex velocity=3.0 m/s, apex height= 3.2 m (*d*) apex velocity=3.3 m/s, apex height= 2.1 m

Figure 22: Pogobot experiment



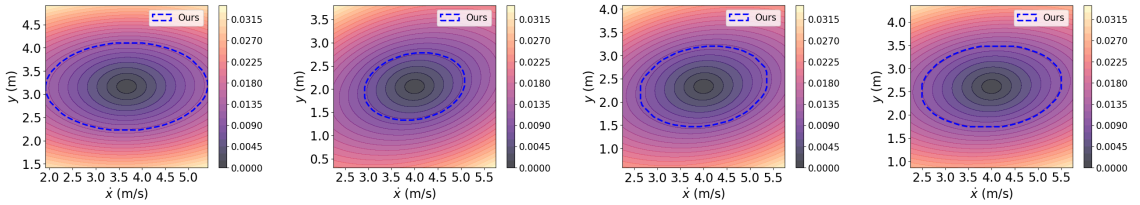
(*a*) apex velocity=3.3 m/s, apex height= 2.3 m (*b*) apex velocity=3.3 m/s, apex height= 2.6 m (*c*) apex velocity=3.3 m/s, apex height= 2.9 m (*d*) apex velocity=3.3 m/s, apex height= 3.2 m

Figure 23: Pogobot experiment



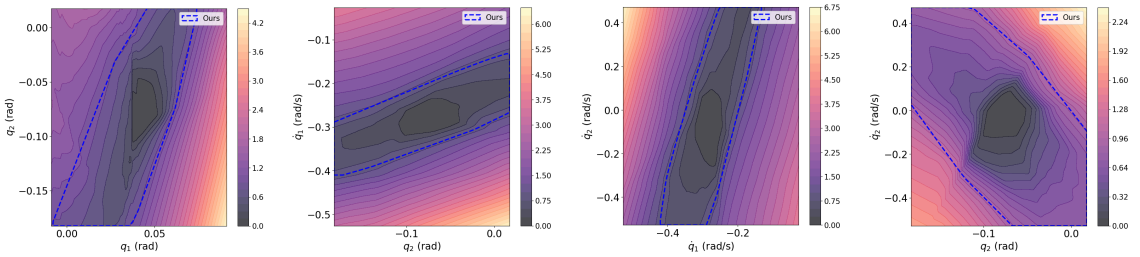
(*a*) apex velocity=3.7 m/s, apex height= 2.1 m (*b*) apex velocity=3.7 m/s, apex height= 2.3 m (*c*) apex velocity=3.7 m/s, apex height= 2.6 m (*d*) apex velocity=3.7 m/s, apex height= 2.9 m

Figure 24: Pogobot experiment



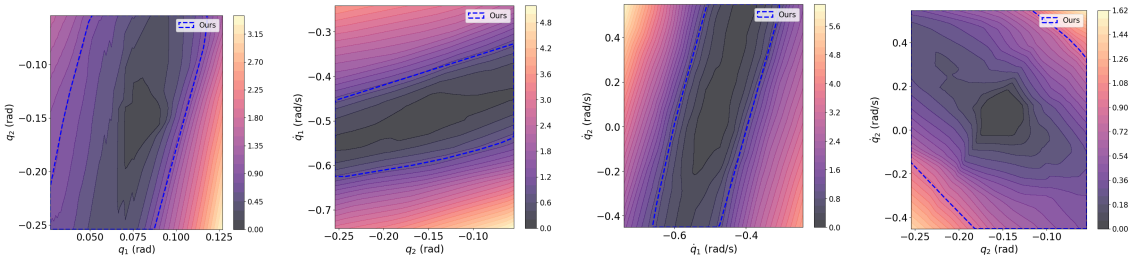
(*a*) apex velocity=3.7 m/s, apex height= 3.2 m (*b*) apex velocity=4.0 m/s, apex height= 2.1 m (*c*) apex velocity=4.0 m/s, apex height= 2.3 m (*d*) apex velocity=4.0 m/s, apex height= 2.6 m

Figure 25: Pogobot experiment



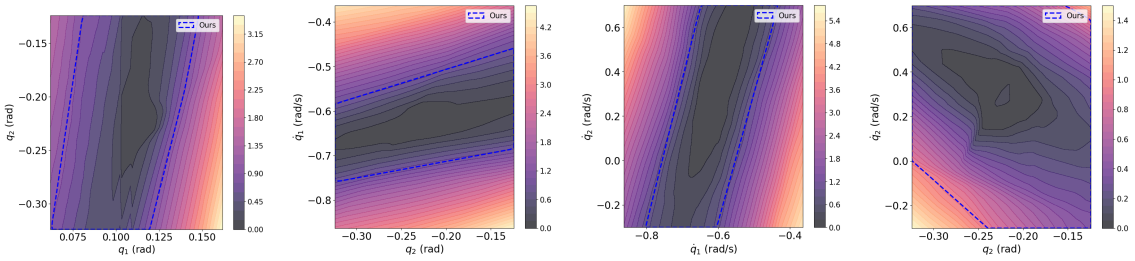
(*a*) x-y plane (*b*) q_2-$\dot{q}_1$ plane (*c*) $\dot{q}_1$-$\dot{q}_2$ plane (*d*) q_2-$\dot{q}_2$ plane

Figure 26: Bipedal walker experiment (reference gait q_1^{ref}=0.05rad)



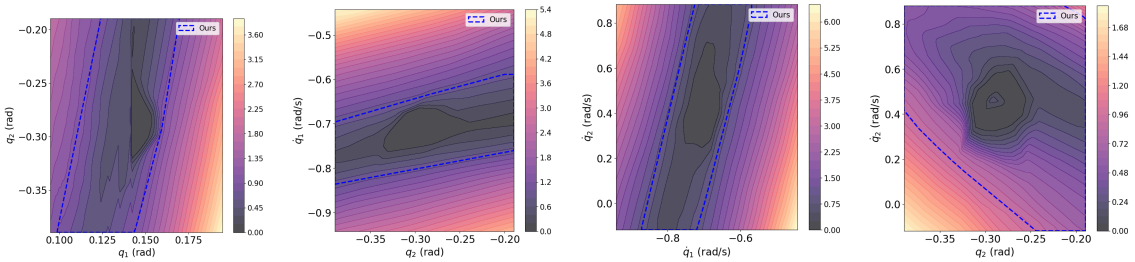
(*a*) x-y plane (*b*) q_2-$\dot{q}_1$ plane (*c*) $\dot{q}_1$-$\dot{q}_2$ plane (*d*) q_2-$\dot{q}_2$ plane

Figure 27: Bipedal walker experiment (reference gait q_1^{ref}=0.08rad)



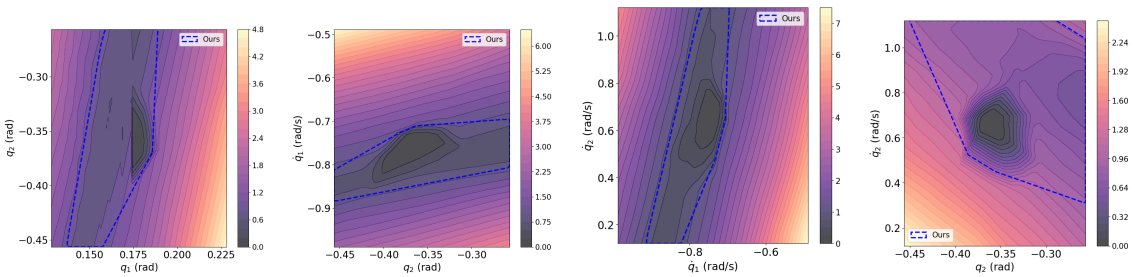
(*a*) x-y plane (*b*) q_2-$\dot{q}_1$ plane (*c*) $\dot{q}_1$-$\dot{q}_2$ plane (*d*) q_2-$\dot{q}_2$ plane

Figure 28: Bipedal walker experiment (reference gait q_1^{ref}=0.10rad)



(*a*) x-y plane (*b*) q_2-$\dot{q}_1$ plane (*c*) $\dot{q}_1$-$\dot{q}_2$ plane (*d*) q_2-$\dot{q}_2$ plane

Figure 29: Bipedal walker experiment (reference gait q_1^{ref}=0.13rad)

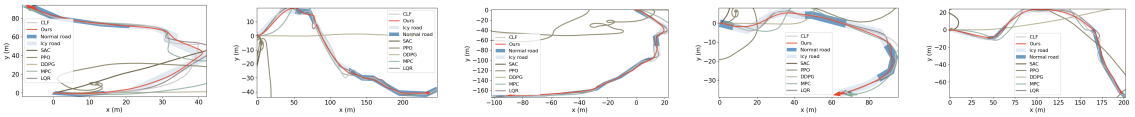


(*a*) x-y plane (*b*) q_2-$\dot{q}_1$ plane (*c*) $\dot{q}_1$-$\dot{q}_2$ plane (*d*) q_2-$\dot{q}_2$ plane

Figure 30: Bipedal walker experiment (reference gait q_1^{ref}=0.18rad)

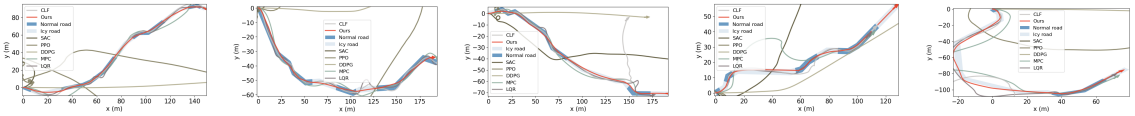
Appendix J. Visualization for the simulations

From Fig. 31 to Fig. 66, we visualize simulation results for all three experiments under different configurations.



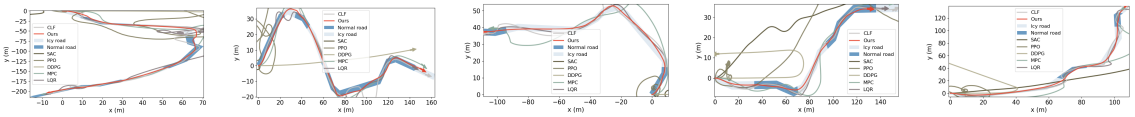
(*a*) Trial 00 (*b*) Trial 01 (*c*) Trial 02 (*d*) Trial 03 (*e*) Trial 04

Figure 31: Car simulation comparisons



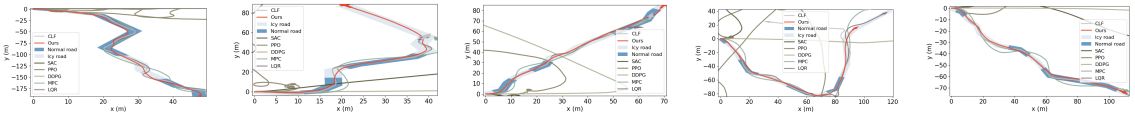
(*a*) Trial 05 (*b*) Trial 06 (*c*) Trial 07 (*d*) Trial 08 (*e*) Trial 09

Figure 32: Car simulation comparisons



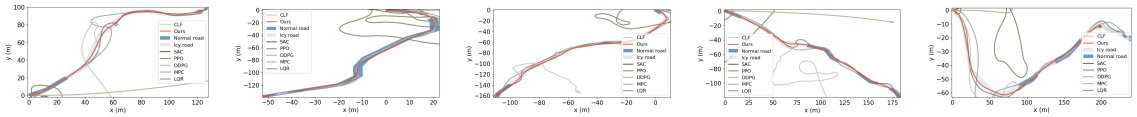
(*a*) Trial 10 (*b*) Trial 11 (*c*) Trial 12 (*d*) Trial 13 (*e*) Trial 14

Figure 33: Car simulation comparisons



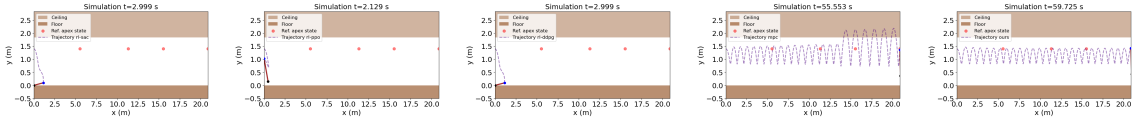
(*a*) Trial 15 (*b*) Trial 16 (*c*) Trial 17 (*d*) Trial 18 (*e*) Trial 19

Figure 34: Car simulation comparisons



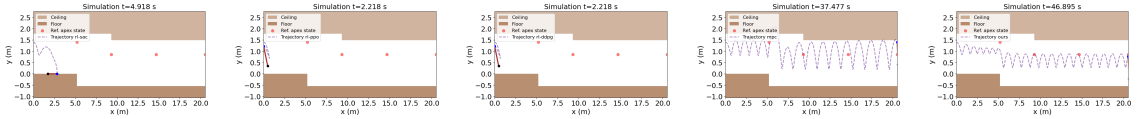
(*a*) Trial 20 (*b*) Trial 21 (*c*) Trial 22 (*d*) Trial 23 (*e*) Trial 24

Figure 35: Car simulation comparisons



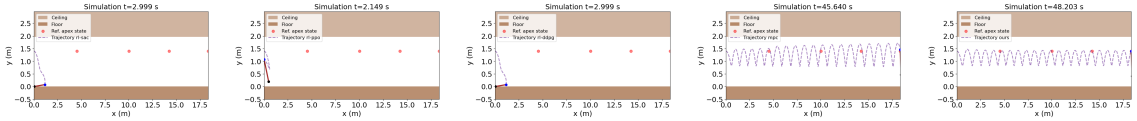
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 36: Pogobot simulation comparisons (trial 04)



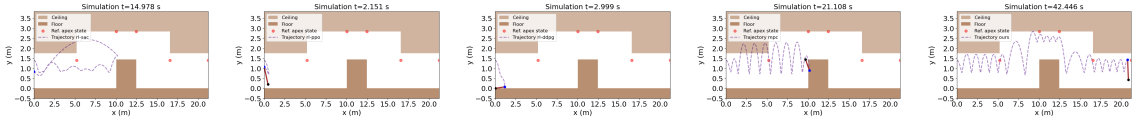
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 37: Pogobot simulation comparisons (trial 05)



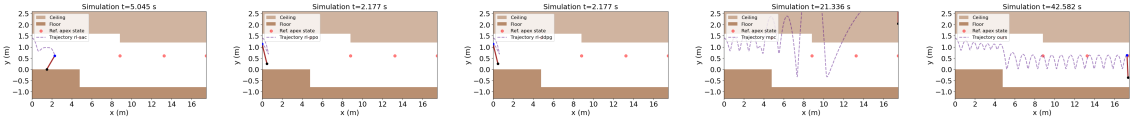
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 38: Pogobot simulation comparisons (trial 06)



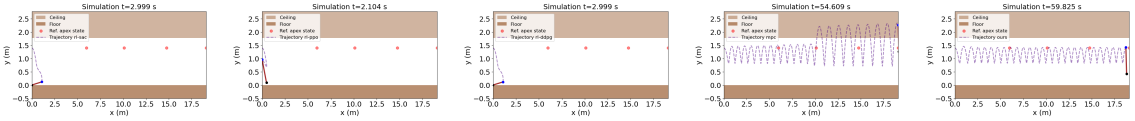
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 39: Pogobot simulation comparisons (trial 07)



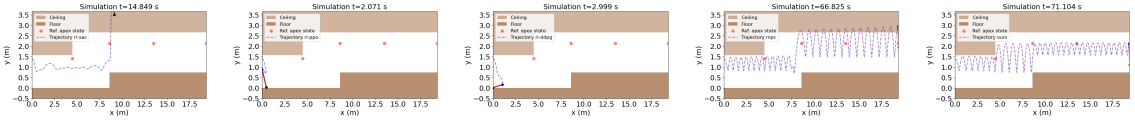
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 40: Pogobot simulation comparisons (trial 08)



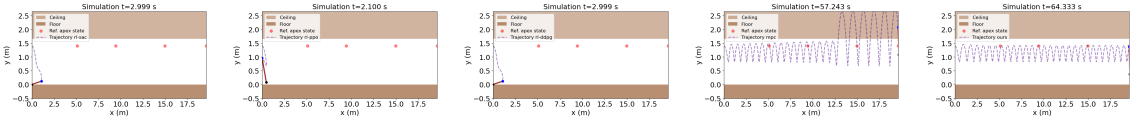
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 41: Pogobot simulation comparisons (trial 09)



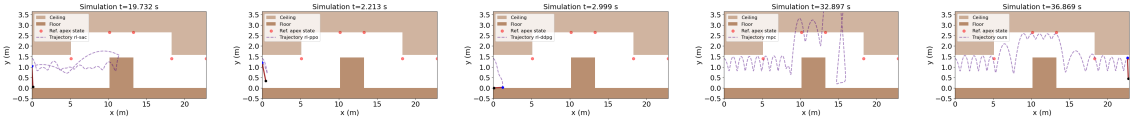
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 42: Pogobot simulation comparisons (trial 10)



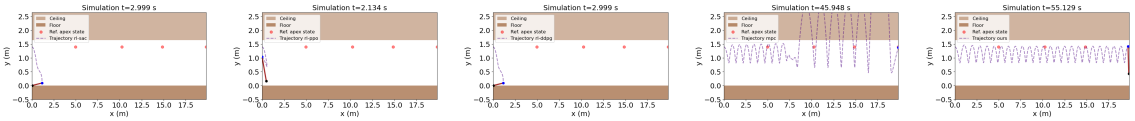
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 43: Pogobot simulation comparisons (trial 11)



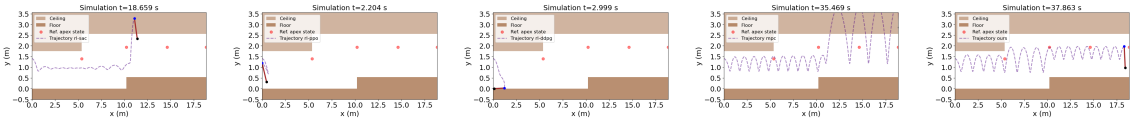
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 44: Pogobot simulation comparisons (trial 12)



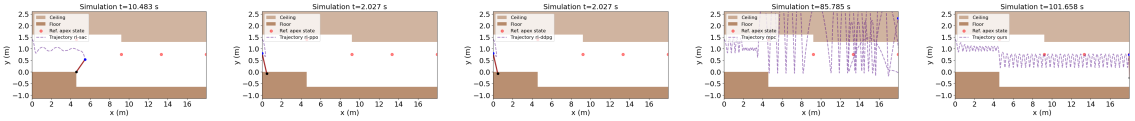
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 45: Pogobot simulation comparisons (trial 13)



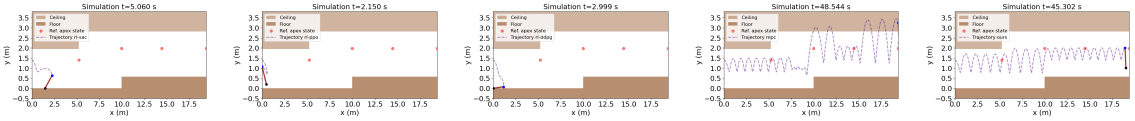
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 46: Pogobot simulation comparisons (trial 14)



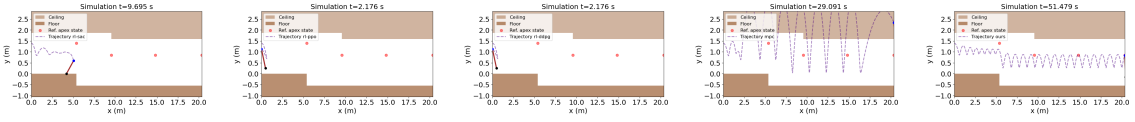
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 47: Pogobot simulation comparisons (trial 15)



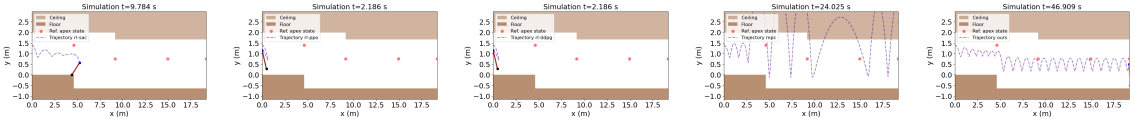
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 48: Pogobot simulation comparisons (trial 16)



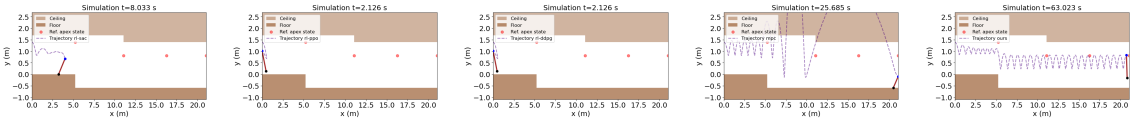
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 49: Pogobot simulation comparisons (trial 17)



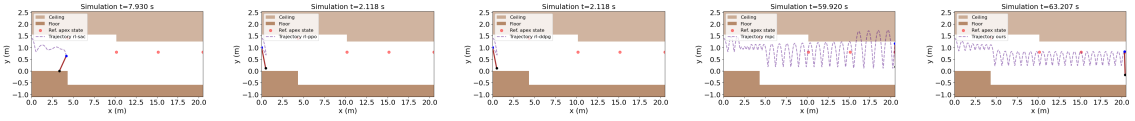
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 50: Pogobot simulation comparisons (trial 18)



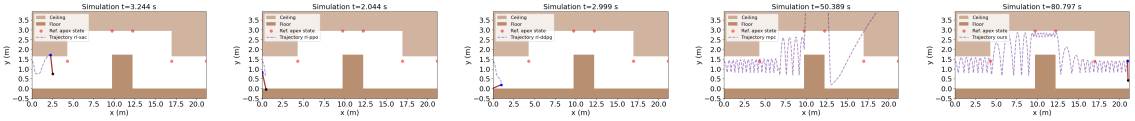
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 51: Pogobot simulation comparisons (trial 19)



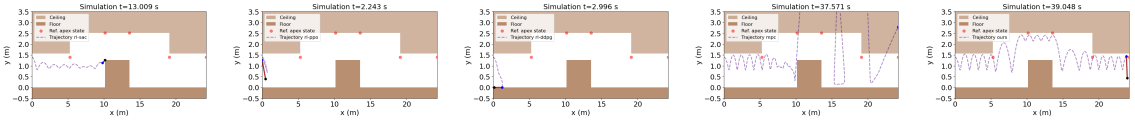
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 52: Pogobot simulation comparisons (trial 20)



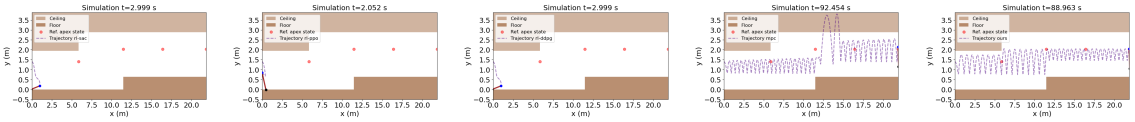
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 53: Pogobot simulation comparisons (trial 21)



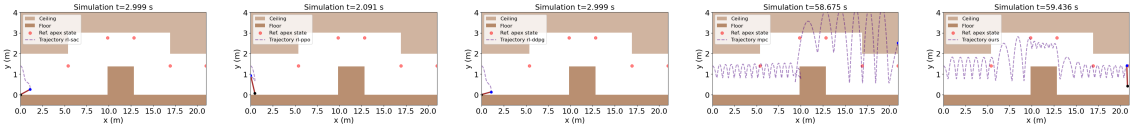
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 54: Pogobot simulation comparisons (trial 22)



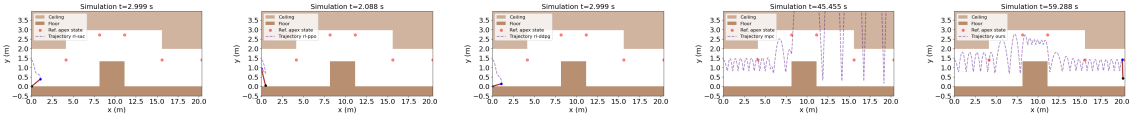
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 55: Pogobot simulation comparisons (trial 23)



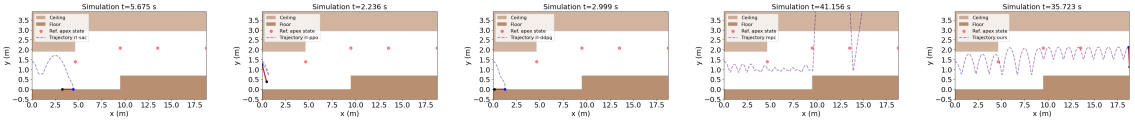
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 56: Pogobot simulation comparisons (trial 24)



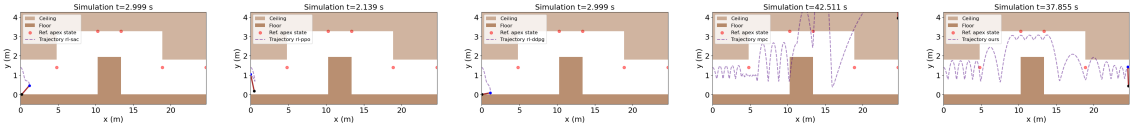
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 57: Pogobot simulation comparisons (trial 25)



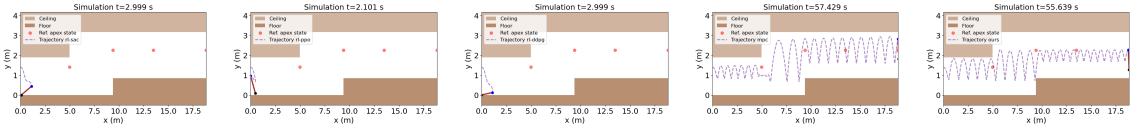
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 58: Pogobot simulation comparisons (trial 26)



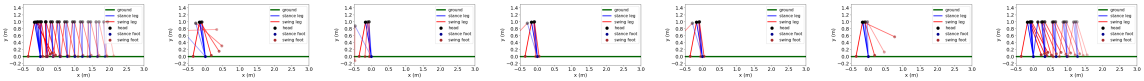
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 59: Pogobot simulation comparisons (trial 27)



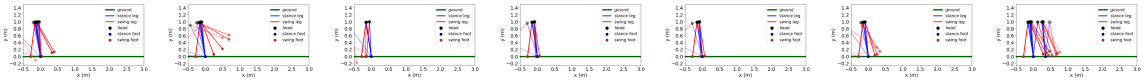
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) Ours

Figure 60: Pogobot simulation comparisons (trial 28)



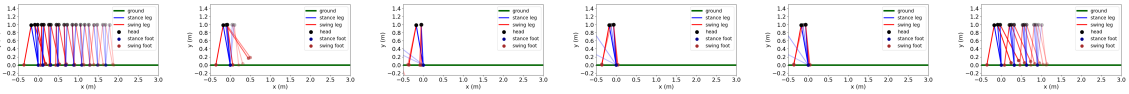
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) QP (*f*) HJB (*g*) Ours

Figure 61: Bipedal walker simulation comparisons (same target angle)



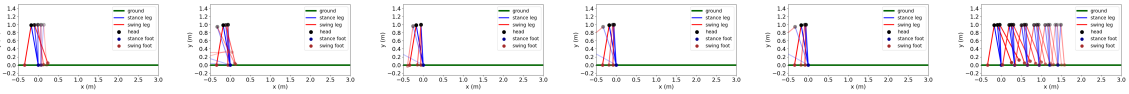
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) QP (*f*) HJB (*g*) Ours

Figure 62: Bipedal walker simulation comparisons (same target angle)



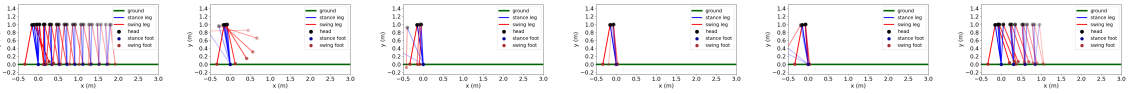
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) QP (*f*) Ours

Figure 63: Bipedal walker simulation comparisons (different target angles)



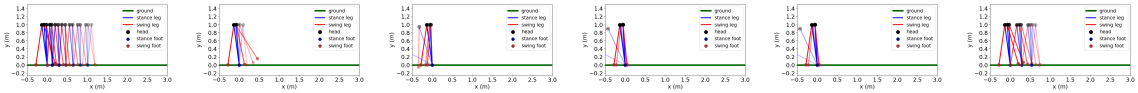
(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) QP (*f*) Ours

Figure 64: Bipedal walker simulation comparisons (different target angles)



(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) QP (*f*) Ours

Figure 65: Bipedal walker simulation comparisons (different target angles)



(*a*) RL-SAC (*b*) RL-PPO (*c*) RL-DDPG (*d*) MPC (*e*) QP (*f*) Ours

Figure 66: Bipedal walker simulation comparisons (different target angles)