Agile But Safe

Learning Collision-Free High-Speed Legged Locomotion

https://agile-but-safe.github.io/

2024.09.01

이정연

Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion

Tairan He* Chong Zhang* Wenli Xiao Guanqi He Changliu Liu Guanya Shi

Carnegie Mellon University ETH zürich

RSS 2024

Outstanding Student Paper Award Finalist (top 3)

Paper Video Summary Code

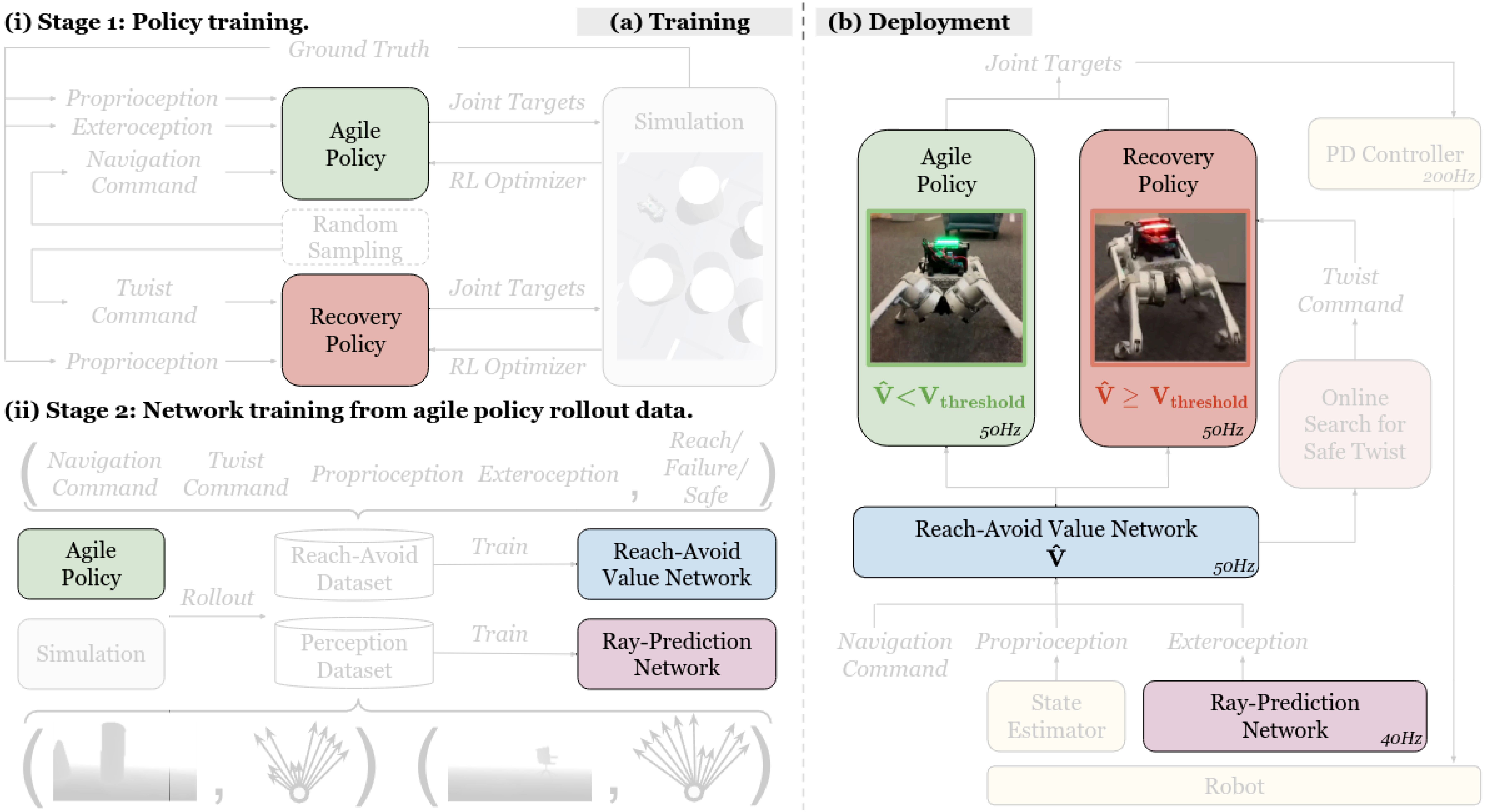
Motivation

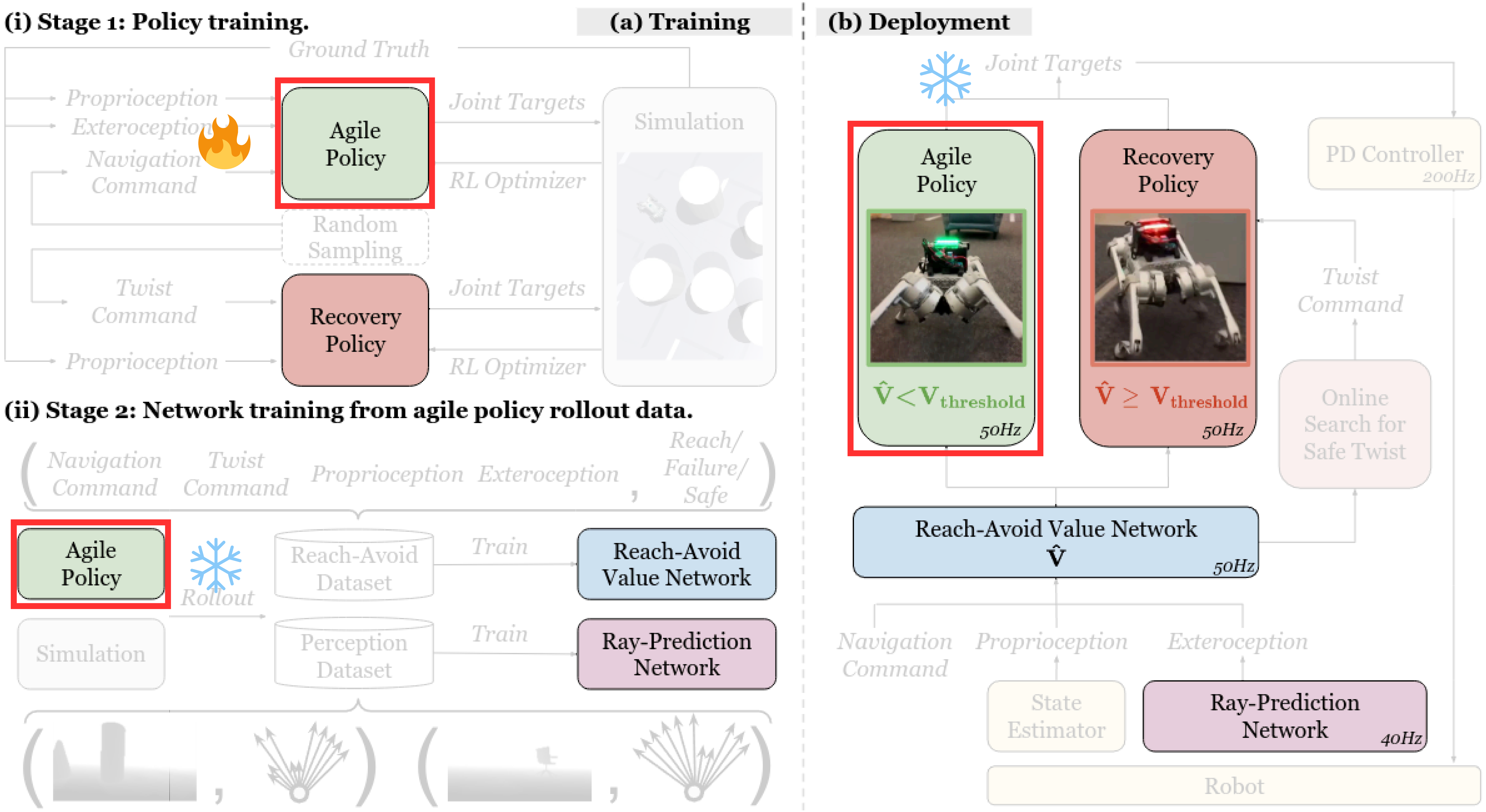
memo

- an agile policy to execute agile motor skills amidst obstacles and a recovery policy to prevent failures, **collaboratively achieving high-speed and collision-free navigation**

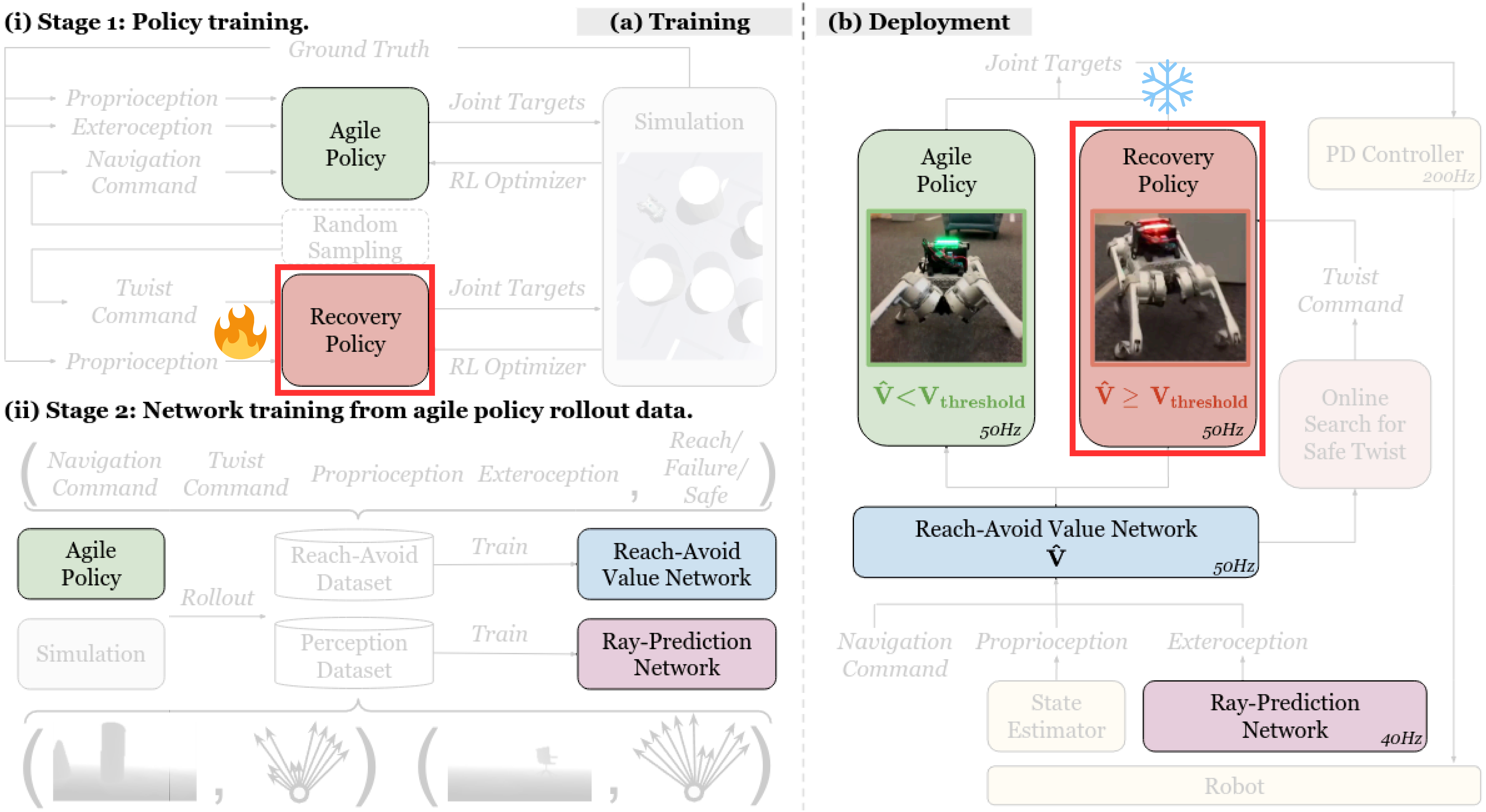
Key Contribution

1. **Agile Policy**: achieve **maximum agility** amidst obstacles
2. **Reach-Avoid Value Network**: predict the **RA values** conditioned on the agile policy as safety indicators
3. **Recovery Policy**: **track desired twist commands** (2D linear velocity & yaw angular velocity) that **lower the RA values**
4. **Ray-Prediction Network**: predict **ray distances** as the policies' exteroceptive inputs **given depth image**

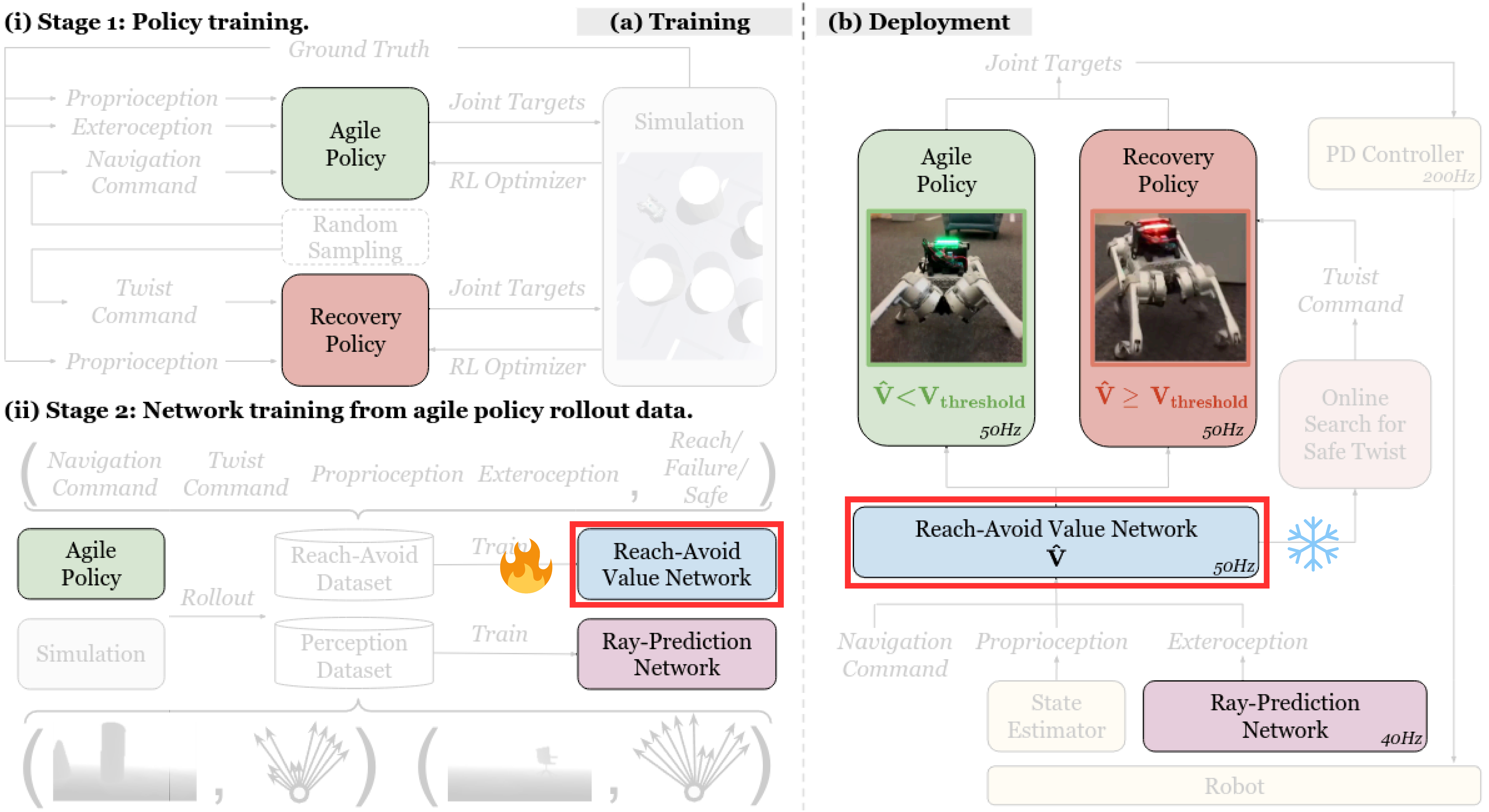

(i) Stage 1: Policy training.
(a) Training
(b) Deployment
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
Simulation
RL Optimizer
Random Sampling
Twist Command
Recovery Policy
Joint Targets
Proprioception
RL Optimizer
(ii) Stage 2: Network training from agile policy rollout data.
Navigation Command
Twist Command
Proprioception
Exteroception
Reach/ Failure/ Safe
Agile Policy
Simulation
Rollout
Reach-Avoid Dataset
Perception Dataset
Train
Train
Reach-Avoid Value Network
Ray-Prediction Network
Joint Targets
Agile Policy
Recovery Policy
PD Controller 200Hz
$\hat{V} < V_{threshold}$
$\hat{V} \geq V_{threshold}$
50Hz
50Hz
Twist Command
Online Search for Safe Twist
Reach-Avoid Value Network $\hat{V}$ 50Hz
Navigation Command
Proprioception
Exteroception
State Estimator
Ray-Prediction Network 40Hz
Robot



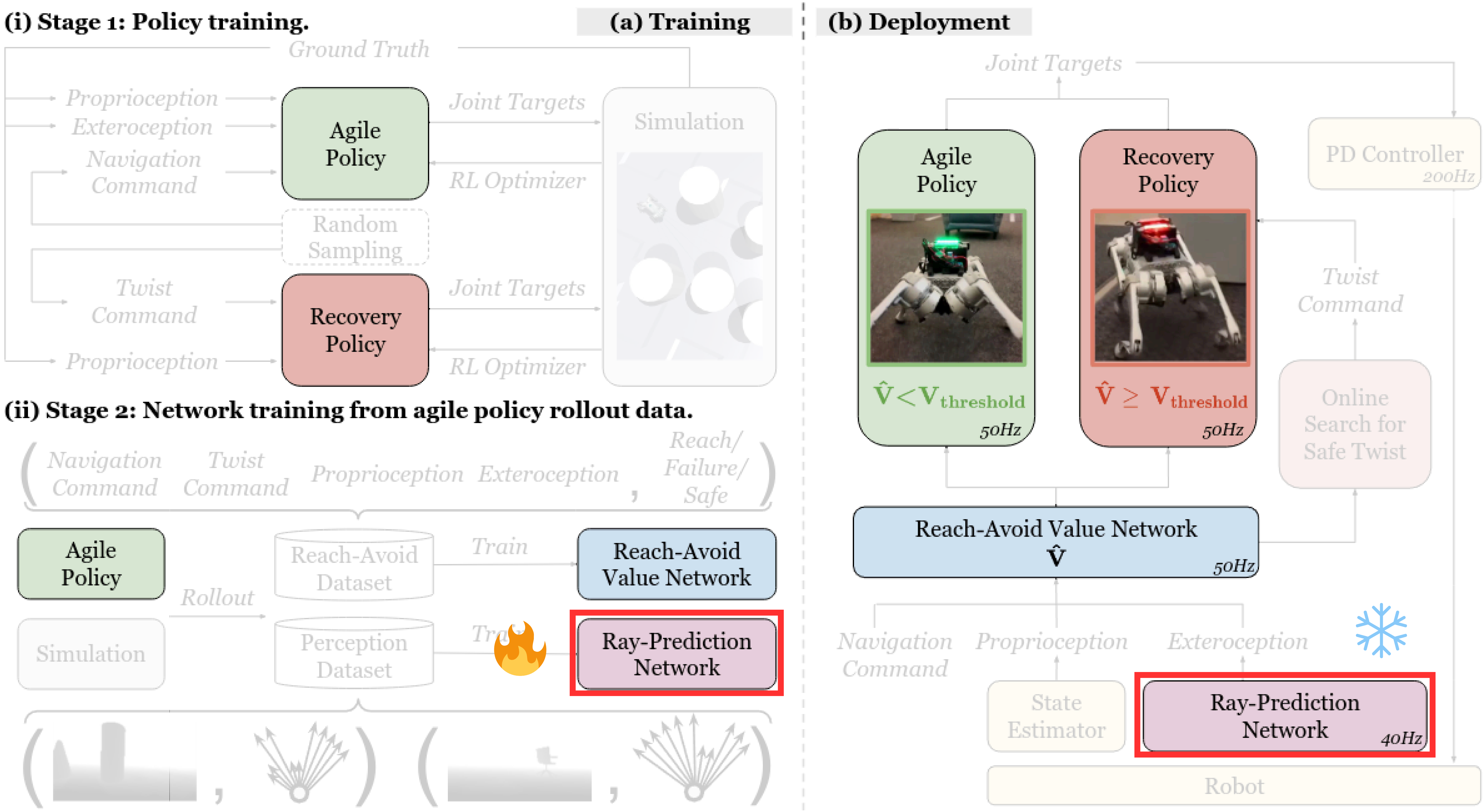
(i) Stage 1: Policy training.
(a) Training
(b) Deployment
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
Simulation
RL Optimizer
Random Sampling
Twist Command
Recovery Policy
Joint Targets
RL Optimizer
Proprioception
(ii) Stage 2: Network training from agile policy rollout data.
Navigation Command
Twist Command
Proprioception
Exteroception
Reach/ Failure/ Safe
Agile Policy
Rollout
Reach-Avoid Dataset
Train
Reach-Avoid Value Network
Simulation
Perception Dataset
Train
Ray-Prediction Network
Joint Targets
Agile Policy
$\hat{V} < V_{\text{threshold}}$
50Hz
Recovery Policy
$\hat{V} \geq V_{\text{threshold}}$
50Hz
PD Controller
200Hz
Twist Command
Online Search for Safe Twist
Reach-Avoid Value Network
$\hat{V}$
50Hz
Navigation Command
Proprioception
Exteroception
State Estimator
Ray-Prediction Network
40Hz
Robot



(i) Stage 1: Policy training.
(a) Training
(b) Deployment
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
Simulation
RL Optimizer
Random Sampling
Twist Command
Recovery Policy
Joint Targets
Proprioception
RL Optimizer
(ii) Stage 2: Network training from agile policy rollout data.
Navigation Command
Twist Command
Proprioception
Exteroception
Reach/ Failure/ Safe
Agile Policy
Reach-Avoid Dataset
Train
Reach-Avoid Value Network
Rollout
Simulation
Perception Dataset
Train
Ray-Prediction Network
Joint Targets
Agile Policy
Recovery Policy
PD Controller
200Hz
$\hat{V} < V_{threshold}$
50Hz
$\hat{V} \geq V_{threshold}$
50Hz
Twist Command
Online Search for Safe Twist
Reach-Avoid Value Network
$\hat{V}$
50Hz
Navigation Command
Proprioception
Exteroception
State Estimator
Ray-Prediction Network
40Hz
Robot



(i) Stage 1: Policy training.
(a) Training
(b) Deployment
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
Simulation
RL Optimizer
Random Sampling
Twist Command
Recovery Policy
Joint Targets
Proprioception
RL Optimizer
(ii) Stage 2: Network training from agile policy rollout data.
Navigation Command
Twist Command
Proprioception
Exteroception
Reach/ Failure/ Safe
Agile Policy
Reach-Avoid Dataset
Train
Reach-Avoid Value Network
Rollout
Simulation
Perception Dataset
Train
Ray-Prediction Network
Joint Targets
Agile Policy
Recovery Policy
PD Controller
200Hz
$\hat{V} < V_{threshold}$
50Hz
$\hat{V} \geq V_{threshold}$
50Hz
Twist Command
Online Search for Safe Twist
Reach-Avoid Value Network
$\hat{V}$
50Hz
Navigation Command
Proprioception
Exteroception
State Estimator
Ray-Prediction Network
40Hz
Robot



(i) Stage 1: Policy training.
(a) Training
(b) Deployment
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
Simulation
RL Optimizer
Random Sampling
Twist Command
Recovery Policy
Joint Targets
Proprioception
RL Optimizer
(ii) Stage 2: Network training from agile policy rollout data.
Navigation Command
Twist Command
Proprioception
Exteroception
Reach/ Failure/ Safe
Agile Policy
Reach-Avoid Dataset
Train
Reach-Avoid Value Network
Rollout
Simulation
Perception Dataset
Train
Ray-Prediction Network
Joint Targets
Agile Policy
Recovery Policy
PD Controller
200Hz
$\hat{V} < V_{threshold}$
50Hz
$\hat{V} \geq V_{threshold}$
50Hz
Twist Command
Online Search for Safe Twist
Reach-Avoid Value Network
$\hat{V}$
50Hz
Navigation Command
Proprioception
Exteroception
State Estimator
Ray-Prediction Network
40Hz
Robot

Safe Reinforcement Learning

memo

- 크게 2가지 부류의 연구가 진행되고 있음
  - **End-to-end**
    - Lagrangian-based method
  - **Hierarchical**
    - structures of underlying dynamics / control-theoretic safety certificates
      - limits the scalability to high-dimensional complex systems
    - learn safety prediction networks (or safety critics)
      - lack interplay between safety critics and backup policies
    - focus on **estimating the reach-avoid values of the agile policy** and **feed the reach avoid values' gradient information back into the system to guide the recovery policy** within a closed loop

Reach-Avoid Problems & Hamilton-Jacobi Analysis

memo

- **Reach-avoid (RA) problems**
  - navigating a system to reach a target while avoiding certain undesirable states
  - leverage contraction properties to derive **a time-discounted reach-avoid Bellman equation**
  - learn a **policy-conditioned** RA value network
- ***HJ reachability analysis**
  - Hamilton-Jacobi partial differential equation, which provides a set of states that the system must stay out of to remain safe
  - HJ 가시성 분석은 Hamilton-Jacobi (HJ) 방정식을 이용하여 시스템의 도달 가능성과 안전성을 분석하는 기법입니다. 하지만 이 분석 기법은 시스템의 차원이 높아질수록 계산 복잡도가 기하급수적으로 증가하는 문제가 있습니다. 이 때문에, 기존의 방법으로는 실시간 응용이나 매우 높은 차원의 시스템에 대한 분석이 어렵습니다.

Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances
https://arxiv.org/abs/1709.07523

Preliminaries

memo

- **Goal-conditioned**
  - goal states: $G \in \Gamma$
  - policy: $\pi : \mathcal{O} \times \Gamma \to \mathcal{A}$
  - reward function: $r : \mathcal{S} \times \mathcal{A} \times \Gamma \to \mathbb{R}$
  - objective:

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|o_t, G), G \sim p_G} \left[\sum_t \gamma_{\mathrm{RL}}^t r(s_t, a_t, G) \right]$$

Preliminaries

memo

- **State Sets**
  - Failure set: unsafe states like collision $\mathcal{F} \subseteq \mathcal{S}$

    failure set

    $\zeta : \mathcal{S} \rightarrow \mathbb{R}$

    $s \in \mathcal{F} \Leftrightarrow \zeta(s) > 0$

  - Target set: = goal $\Theta \subset \mathcal{S}$

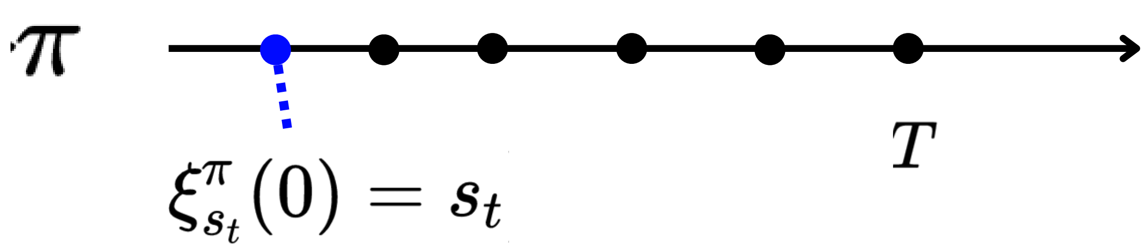
    target set

    $l : \mathcal{S} \rightarrow \mathbb{R}$

    $s \in \Theta \Leftrightarrow l(s) \leq 0$

  - Reach-Avoid set:

    $\xi^{\pi}_{s_t}(\cdot)$ futue trajecotry rollour from state s_t



$$\mathcal{RA}^{\pi}(\Theta; \mathcal{F}) := \{s_t \in \mathcal{S} \mid \xi^{\pi}_{s_t}(T - t) \in \Theta \wedge$$ and
$$\forall t' \in [0, T - t], \xi^{\pi}_{s_t}(t') \notin \mathcal{F}\}$$

Preliminaries

memo

- **Reach-Avoid Value**
  - policy-conditioned reach-avoid values
  - fixed-point RA Bellman equation 만족

$$V^{\pi}_{\mathrm{RA}^*}(s) \leq 0 \Leftrightarrow s \in \mathcal{RA}^{\pi}(\Theta; \mathcal{F})$$

증명은 다른 논문에서 * TODO

$$V^{\pi}_{\mathrm{RA}^*}(s) = \max\Big\{\zeta(s), \min\Big\{l(s), V^{\pi}_{\mathrm{RA}^*}(f(s, \pi(s)))\Big\}\Big\}$$

  - 수렴성 보장을 위해, time-discounted 적용

failure set target set

$$\begin{aligned} V^{\pi}_{\mathrm{RA}}(s) =& \gamma_{\mathrm{RA}} \max\{\zeta(s), \min\{l(s), V^{\pi}_{\mathrm{RA}}(f(s, \pi(s)))\}\} \\ &+ (1 - \gamma_{\mathrm{RA}}) \max\{l(s), \zeta(s)\} \end{aligned}$$

$$V^{\pi}_{\mathrm{RA}}(s) \longrightarrow V^{\pi}_{\mathrm{RA}^*}(s)$$

under -approx.

System Structure

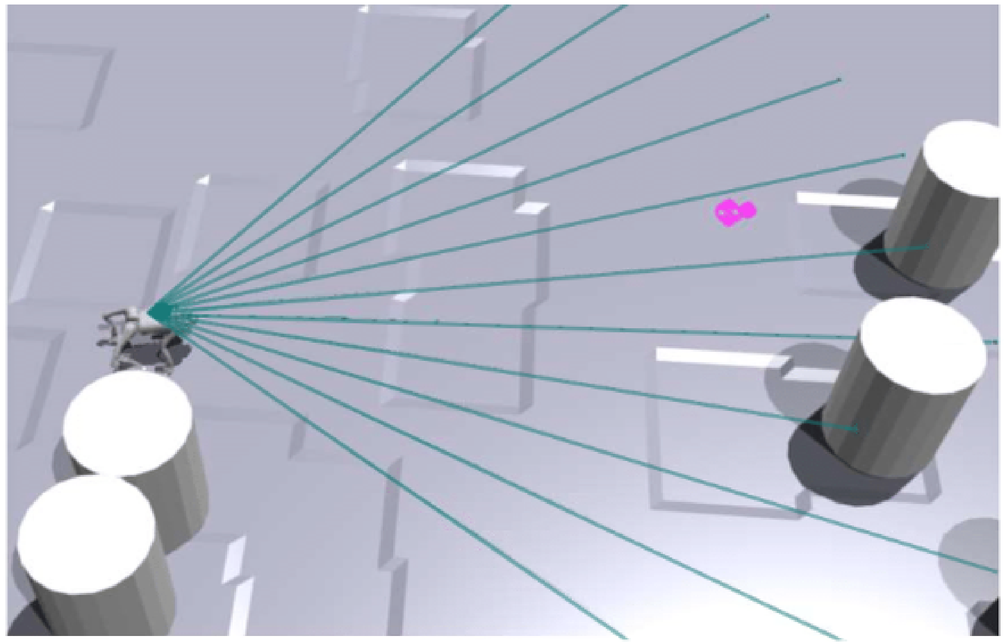
memo

- **Dual policy**
  - **agile policy**
    - 3.1m/s까지 goal command 따라가기
    - target 2D positions and headings
    - in most time
  - **recovery policy**
    - twist command 따라가며 collision avoid
    - 2D linear velocity and yaw rate
    - only risky situation

$$\pi^{\text{Agile}} \leftarrow \hat{V} \geq V_{\text{threshold}}$$

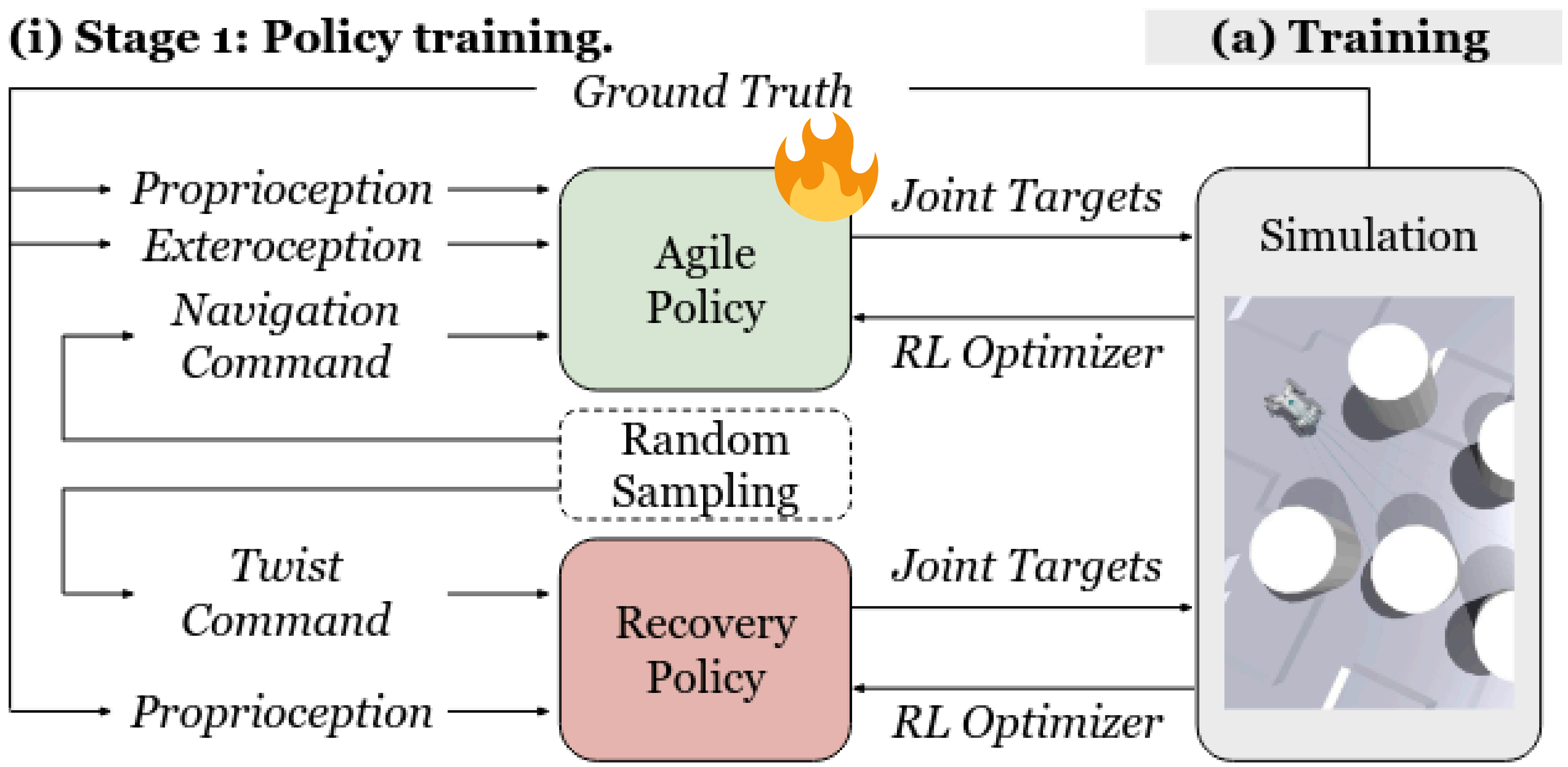
$$\pi^{\text{Recovery}} \leftarrow \hat{V} < V_{\text{threshold}}$$

- **Exteroceptive inputs**
  - low-dimensional representation
  - 11 rays (sparse LiDAR readings)
  - map raw depth img to predicted ray distance
  - agile policy와 RA value network에 observation으로 들어감



Policy Training

Stage1



(i) Stage 1: Policy training.
(a) Training
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
RL Optimizer
Simulation
Random Sampling
Twist Command
Recovery Policy
Joint Targets
RL Optimizer
Proprioception

Agile Policy

memo

- goal-reaching formulation
- train sensorimotor skills that enable the robot to reach specified goals within the episode time w/o collisions

Observation

- foot contacts: $c_{f \in \{1,2,3,4\}}$
- base angular velocities: ω
- **projected gravity:** g
- **goal commands:** G^c
- time left: $T - t$
- joint positions/velocities $q \quad \dot{q}$
- previous actions: a
- exteroception R

o^{Agile}

State Estimators

:IMU-based orientation estimation for g (i.e., roll and pitch) is usually very accurate, and our policy can effectively handle the odometry drift

- change our goal commands even in the run time
- easily overwrite goal commands to achieve instant agile steering

TABLE IX
GOAL COMMANDS FOR INSTANT STEERING

Steering	Goal x (m)	Goal y (m)	Goal Heading (rad)
Forward	5	0	0
Stop	0	0	0
Left Turn	2	1.5	$\frac{\pi}{2}$
Rapid Left Turn	-2	0	3
Right Turn	2	-1.5	$-\frac{\pi}{2}$
Rapid Right Turn	-2	0	-3

Agile Policy

memo

- goal-reaching formulation
- train sensorimotor skills that enable the robot to reach specified goals within the episode time w/o collisions

Action

- 12-d joint targets
- PD controller tracks these joint targets
- fully-connected MLP

$$\tau = K_p(a - q) - K_d\dot{q}$$

Agile Policy

memo

- goal-reaching formulation
- train sensorimotor skills that enable the robot to reach specified goals within the episode time w/o collisions

Reward

- Penalty
- Task G^c
- Regularization

$$r = r_{\text{penalty}} + r_{\text{task}} + r_{\text{regularization}}$$

$$r_{\text{penalty}} = -100 \cdot \mathbf{1}(\text{ undesired collision })$$

$$r_{\text{task}} = 60 \cdot \underline{r_{\text{possoft}}} + 60 \cdot \underline{r_{\text{postight}}} + 30 \cdot \underline{r_{\text{heading}}} - 10 \cdot r_{\text{stand}} + 10 \cdot r_{\text{agile}} - 20 \cdot r_{\text{stall}}$$

$$\begin{aligned} r_{\text{regularization}} = & -2 \cdot v_z^2 - 0.05 \cdot (\omega_x^2 + \omega_y^2) - 20 \cdot (g_x^2 + g_y^2) \\ & - 0.0005 \cdot \|\tau\|_2^2 - 20 \cdot \sum_{i=1}^{12} \text{ReLU}(|\tau_i| - 0.85 \cdot \tau_{i,\,\text{lim}}) \\ & - 0.0005 \cdot \|\dot{q}\|_2^2 - 20 \cdot \sum_{i=1}^{12} \text{ReLU}(|\dot{q}_i| - 0.9 \cdot \dot{q}_{i,\,\text{lim}}) \\ & - 20 \cdot \sum_{i=1}^{12} \text{ReLU}(|q_i| - 0.95 \cdot q_{i,\,\text{lim}}) \\ & - 2 \times 10^{-7} \cdot \|\ddot{q}\|_2^2 - 4 \times 10^{-6} \cdot \|\dot{a}\|_2^2 - 20 \cdot \mathbf{1}(\text{ fly }) \end{aligned}$$

RL-based navigation planners:
free from explicit motion constraints such as target velocities that may limit the agility

$$r_{\text{track (possoftpostight/heading)}} = \frac{1}{1 + \left\| \frac{\text{error}}{\sigma} \right\|^2} \cdot \frac{\mathbf{1}(t > T - T_r)}{T_r}$$

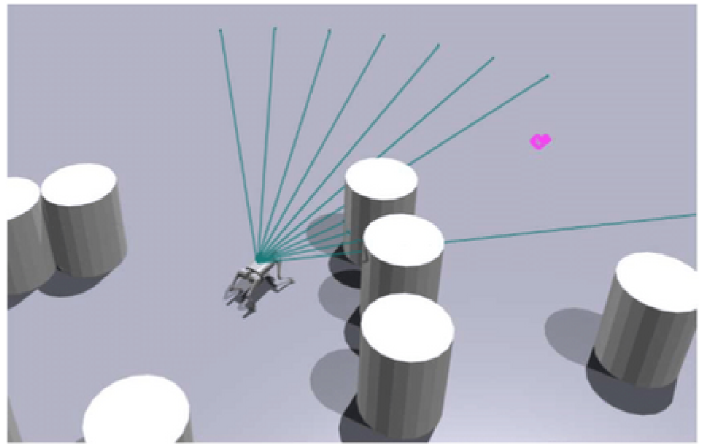


Fig. 3. Example training environments. The **magenta** points indicate the goals, and the **bluegreen** lines indicate the exteroceptive ray observations. Terrains from left to right: flat, low stumbling blocks, and rough.

Agile Policy

memo

- goal-reaching formulation
- train sensorimotor skills that enable the robot to reach specified goals within the episode time w/o collisions

Simulation Training

- Isaac Gym / 1280 environment/PPO
- flat, rough, or low stumbling blocks(height difference from 0 cm to 7 cm)
- cylinders of 40 cm radius / 0~8 obstacles randomly distributed in [11 m× 5 m]
- **two DRs are critical: illusion & ERFI-50**
  - illusion: policy more robust to unseen geometries such as walls: it overwrites the observed ray distances
  - ERFI-50:randomly bias the joint positions to model the motor encoders' offset errors
- Curriculum learning

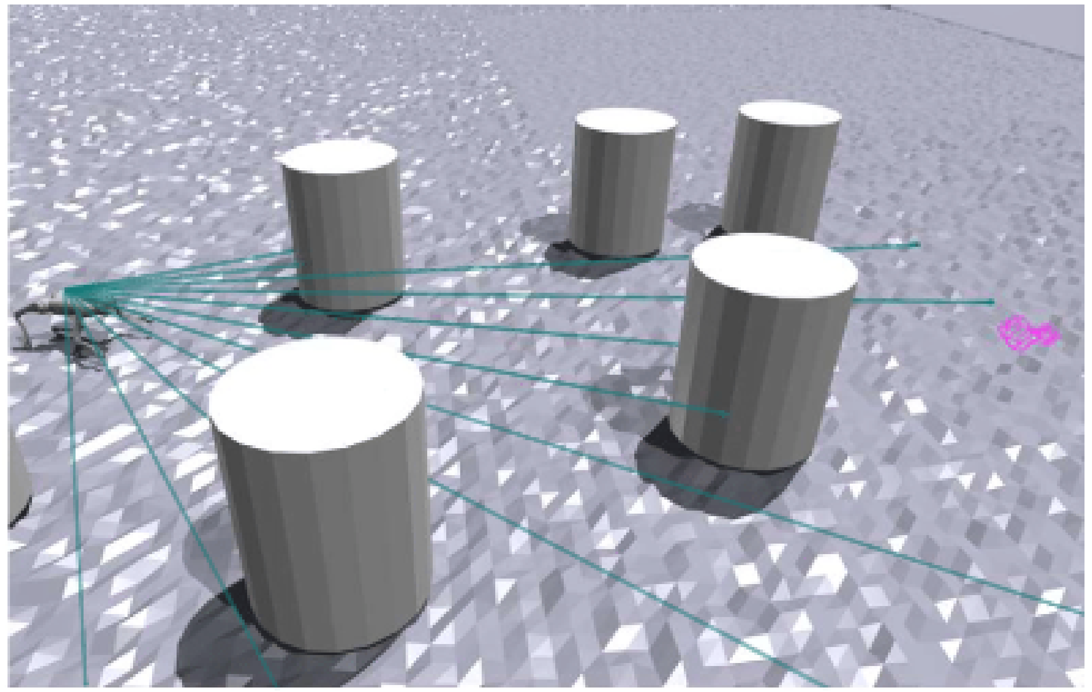


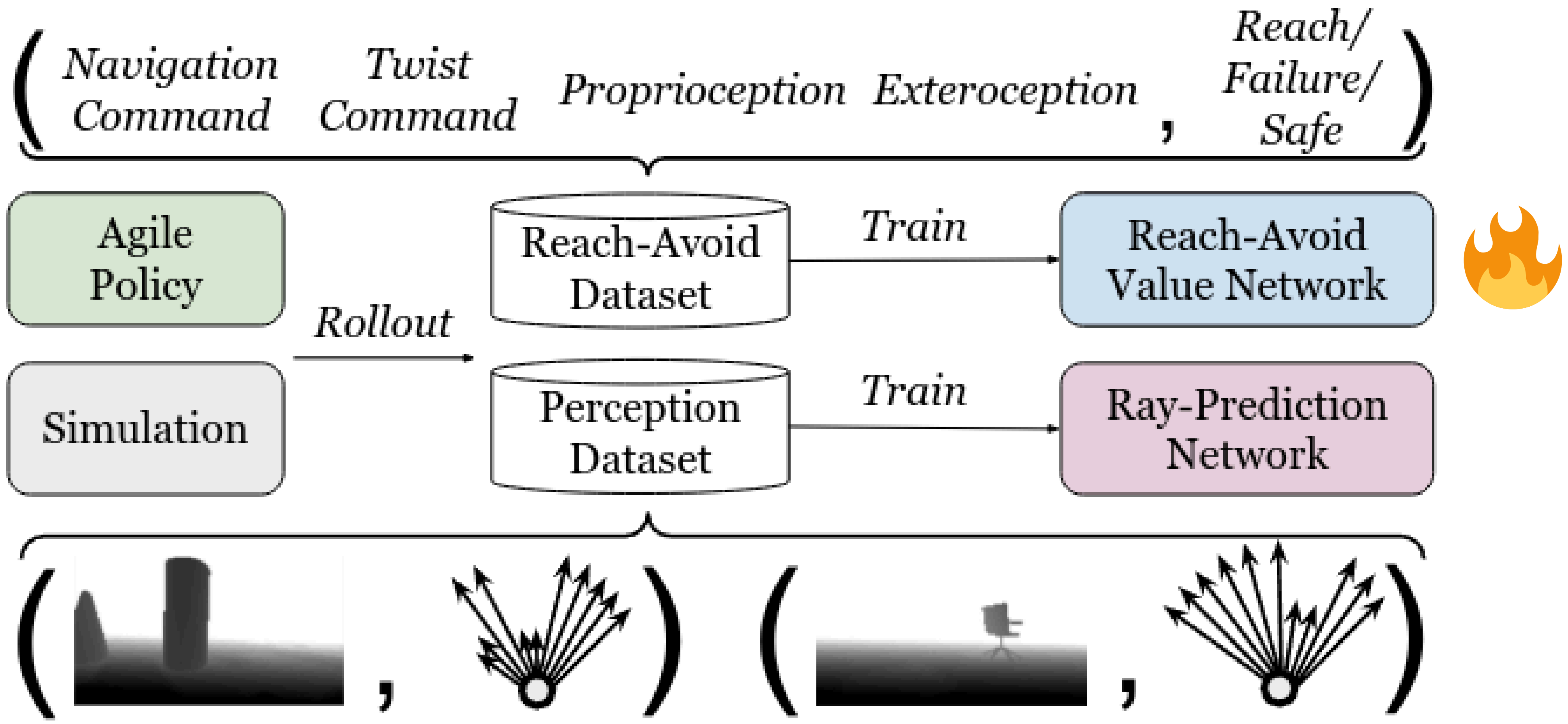
TABLE II
DOMAIN RANDOMIZATION SETTINGS FOR AGILE POLICY TRAINING

Term	Value
Observation	
Illusion	Enabled
Joint position noise	$\mathcal{U}(-0.01, 0.01)$ rad
Joint velocity noise	$\mathcal{U}(-1.5, 1.5)$ rad/s
Angular velocity noise	$\mathcal{U}(-0.2, 0.2)$ rad/s
Projected gravity noise	$\mathcal{U}(-0.05, 0.05)$
log(ray distance) noise	$\mathcal{U}(-0.2, 0.2)$
Dynamics	
ERFI-50 [8]	0.78 N m× difficulty level
Friction factor	$\mathcal{U}(0.4, 1.1)$
Added base mass	$\mathcal{U}(-1.5, 1.5)$ kg
Joint position biases	$\mathcal{U}(-0.08, 0.08)$ rad
Episode	
Episode length	$\mathcal{U}(7.0, 9.0)$ s
Initial robot position	$x = 0, y = 0$
Initial robot yaw	$\mathcal{U}(-\pi, \pi)$ rad
Initial robot twist	$\mathcal{U}(-0.5, 0.5)$ m/s or rad/s
Goal Position	$x_{\text{goal}} \sim \mathcal{U}(1.5, 7.5)$ m $y_{\text{goal}} \sim \mathcal{U}(-2.0, 2.0)$ m
Goal Heading	$\arctan 2(y_{\text{goal}}, x_{\text{goal}}) + \mathcal{U}(-0.3, 0.3)$ rad

Network Training

Stage2

(ii) Stage 2: Network training from agile policy rollout data.



RA Learning

memo

- To safeguard the robot, we propose to use RA values to predict the failures, and then a recovery policy can save the robot based on the RA values.
- Not learn the global RA values, but make it **policy-conditioned**

RA value

- use a reduced set of observations as the inputs of the RA value function
- train an RA value network

exteroception

$$o^{\mathrm{RA}} = \left[[v; \omega]; G^{c}_{x,y}; R \right]$$

base twists

the goal (x, y) position in the robot frame

network

$$V_{\mathrm{RA}}^{\pi^{\mathrm{Agile}}}(s) \approx \hat{V}\left(o^{\mathrm{RA}}\right)$$

loss

$$L = \frac{1}{T} \sum_{t=1}^{T} \left(\hat{V}\left(o_t^{\mathrm{RA}}\right) - \hat{V}^{\mathrm{target}} \right)^2$$

$$\hat{V}^{\mathrm{target}} = \gamma_{\mathrm{RA}} \max\left\{ \zeta(s_t), \min\left\{ l(s_t), \hat{V}^{\mathrm{old}}\left(o_{t+1}^{\mathrm{RA}}\right) \right\} \right\} + (1 - \gamma_{\mathrm{RA}}) \max\{ l(s_t), \zeta(s_t) \}$$

RA Learning

memo

- To safeguard the robot, we propose to use RA values to predict the failures, and then a recovery policy can save the robot based on the RA values.
- Not learn the global RA values, but make it **policy-conditioned**

Implementation

- l(s) and ζ(s) should be Lipschitz continuous for theoretical guarantees
  - define:

$$\begin{aligned}\hat{V}^{\text{target}} =& \gamma_{\text{RA}} \max\left\{\zeta(s_t), \min\left\{l(s_t), \hat{V}^{\text{old}}\left(o_{t+1}^{\text{RA}}\right)\right\}\right\} \\ &+ (1-\gamma_{\text{RA}}) \max\{l(s_t), \zeta(s_t)\}\end{aligned}$$

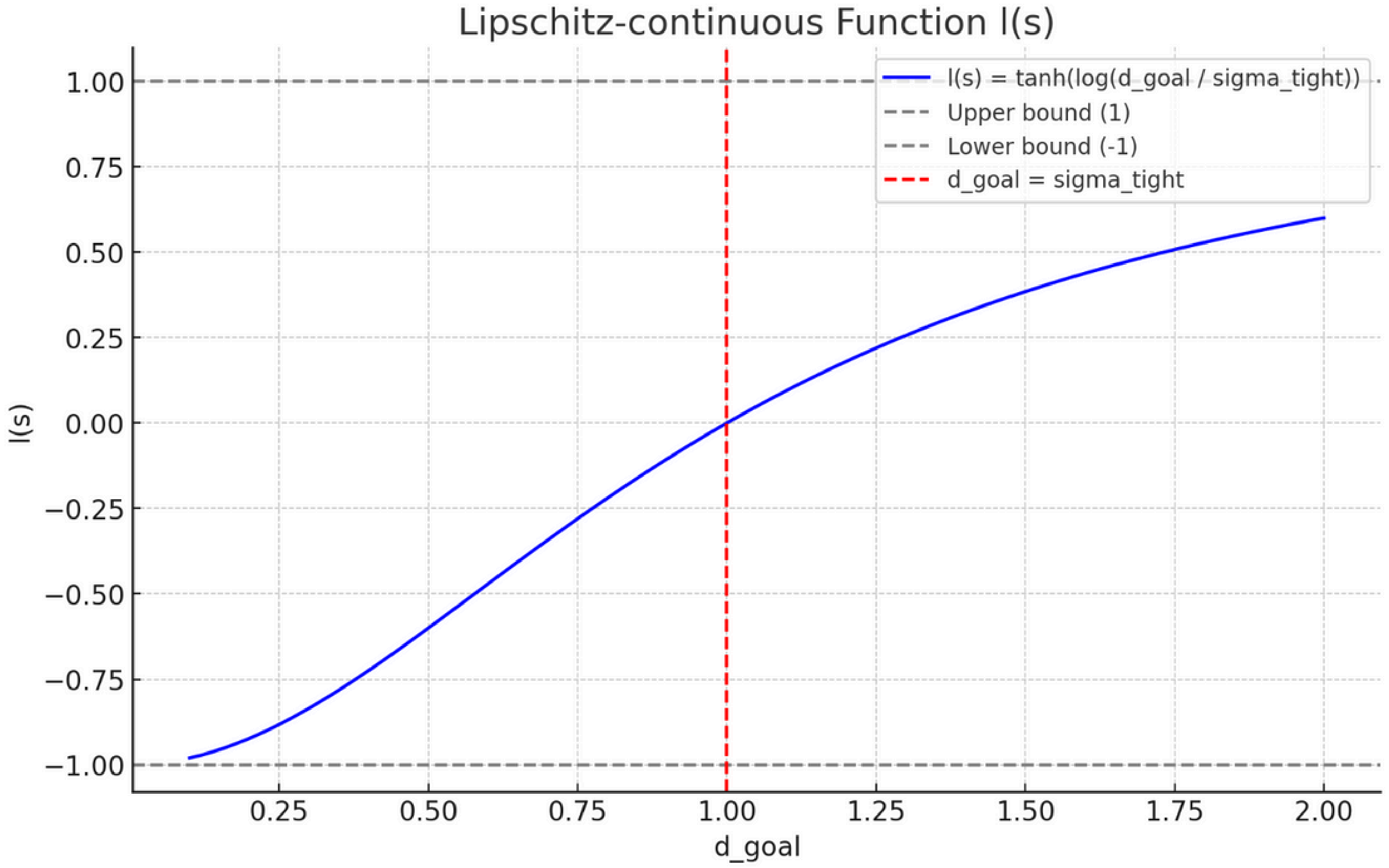
target set $$l(s) = \tanh \log \frac{d_{\text{goal}}}{\sigma_{\text{tight}}}$$ bounding it with $(-1, 1)$, and setting $d_{\text{goal}} \leq \sigma_{\text{tight}}$ as "reach"

failure set $$\zeta(s) = 2 * \mathbf{1}(\text{ undesired collision }) - 1$$

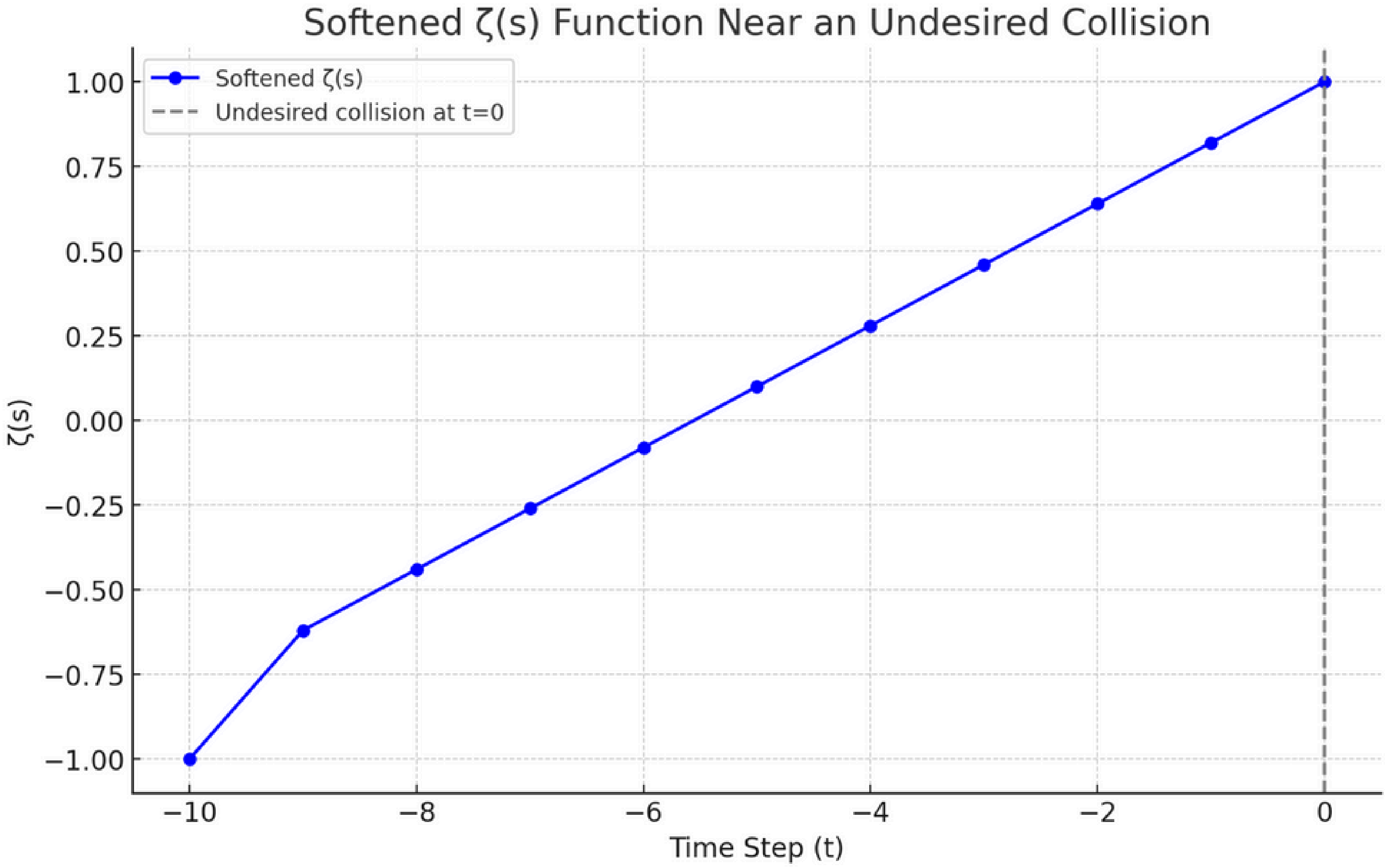
soften the function in a hindsight way

ζ values for the last 10 timesteps are relabelled to be $-0.8, -0.6, \ldots, 0.8, 1.0$

$$l(s) = \tanh \log \frac{d_{\text{goal}}}{\sigma_{\text{tight}}}$$



$$\zeta(s) = \mathbf{2} * \mathbf{1}(\text{ undesired collision }) - 1$$



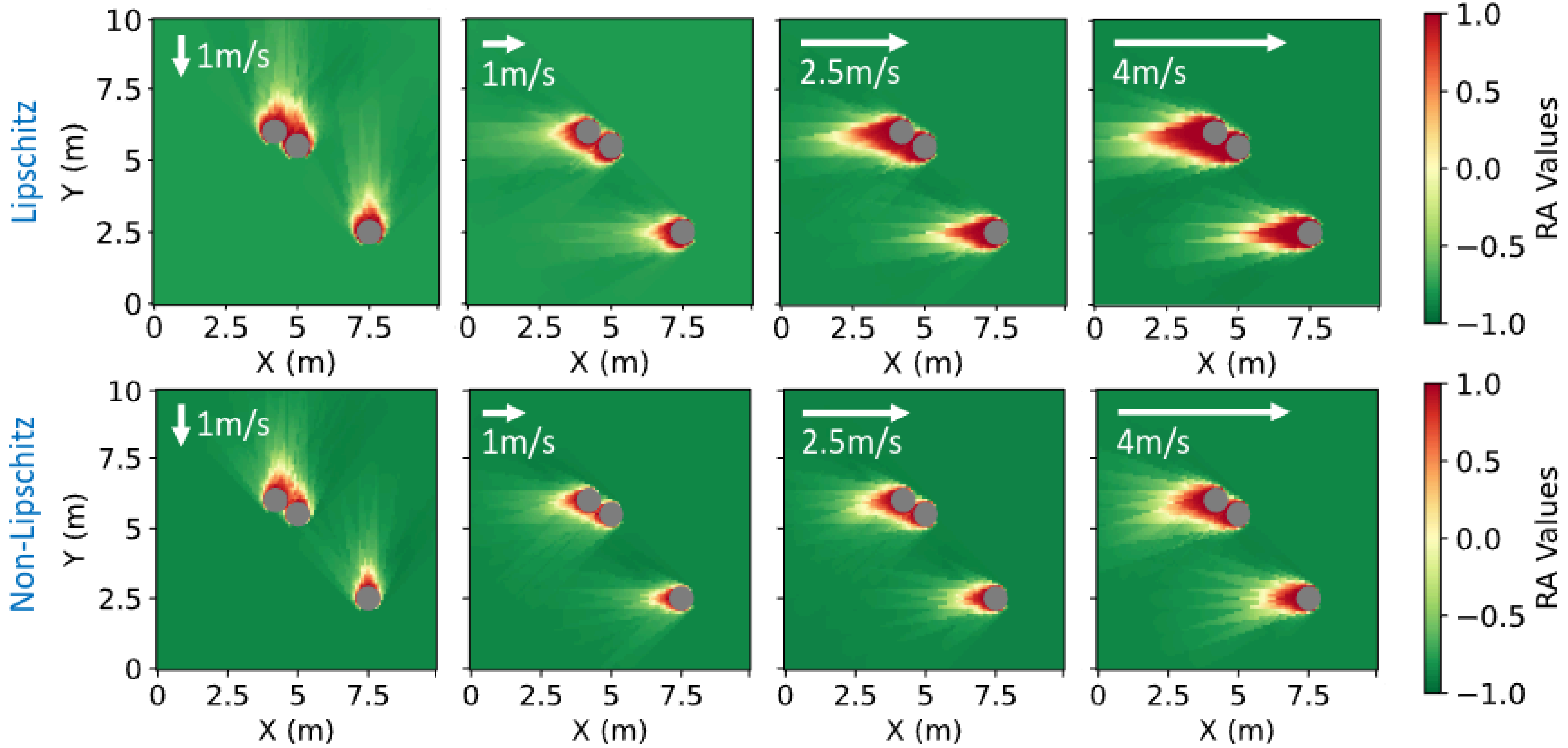


Fig. 4. Visualization of $\hat{V}$ with different linear velocities and 2D positions relative to the 3 fixed obstacles. The angular velocities are set to zero, and the relative goal commands are set to 5 m ahead of the robot. The grey circles represent the obstacles, and the colors represent the values of $\hat{V}$ at corresponding 2D positions. The first row presents the RA values trained with the softened failure function ζ, while the second row uses the raw one in Equation (19). Without softening ζ to approach the Lipschitz continuity, the value estimation fails to indicate collisions on the sides of obstacles and has local minima in front of the obstacles, compromising safety.

RA Learning

memo

- To safeguard the robot, we propose to use RA values to predict the failures, and then a recovery policy can save the robot based on the RA values.
- Not learn the global RA values, but make it **policy-conditioned**

For Recovery

- Robot decides **the optimal twist** to avoid collisions using the RA value function, and employs the recovery policy to track these twist command
- recovery policy is triggered as a backup shielding policy if and only if $\hat{V}\left(o^{\mathrm{RA}}\right) \geq V_{\text{threshold}}$
  -0.05
- recovery policy가 targetting하는 twist cmd

$$tw^c = \left[v_x^c, v_y^c, 0, 0, 0, \omega_z^c\right]$$

approximate distance to the goal after tracking the twist command
for a small amount of time δt = 0.05 s

$$tw^c = \arg\min d_{\text{goal}}^{\text{future}} \text{ s.t. } \hat{V}\left(\left[tw^c; G_{x,y}^c; R\right]\right) < V_{\text{threshold}}$$

$$\delta x = v_x^c \delta t - 0.5 v_y^c \omega_z^c \delta t^2$$
$$\delta y = v_y^c \delta t + 0.5 v_x^c \omega_z^c \delta t^2$$

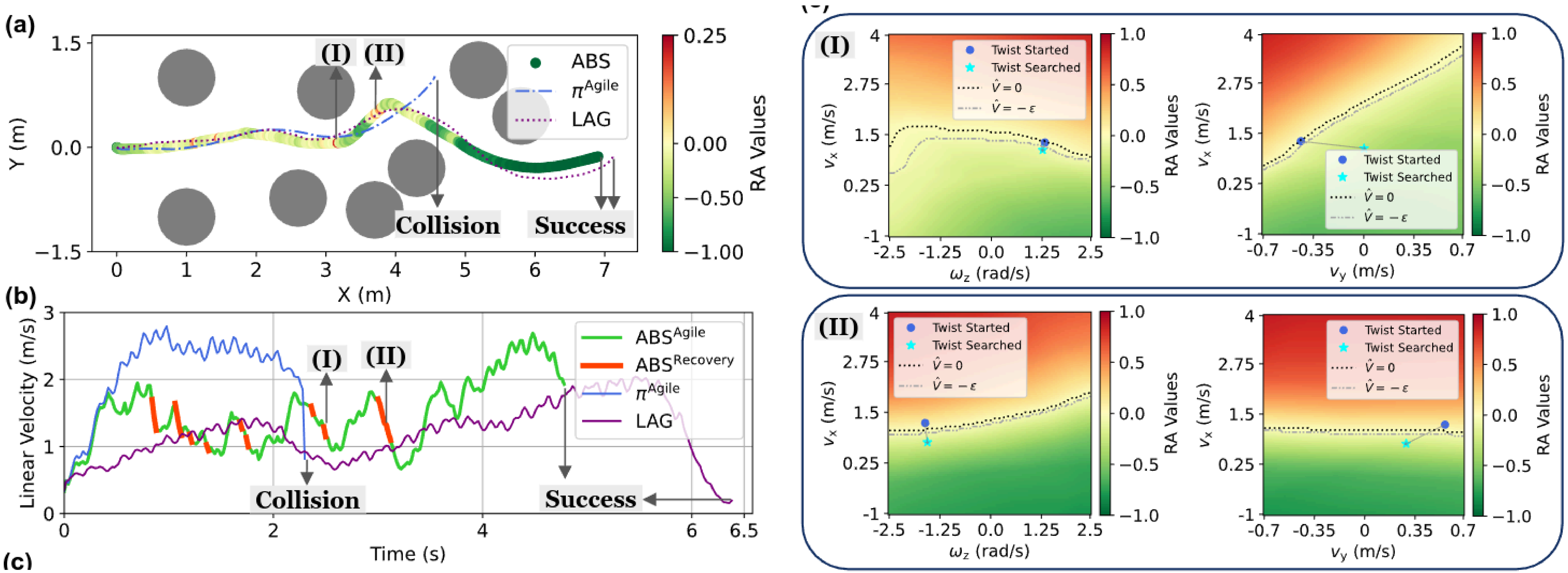
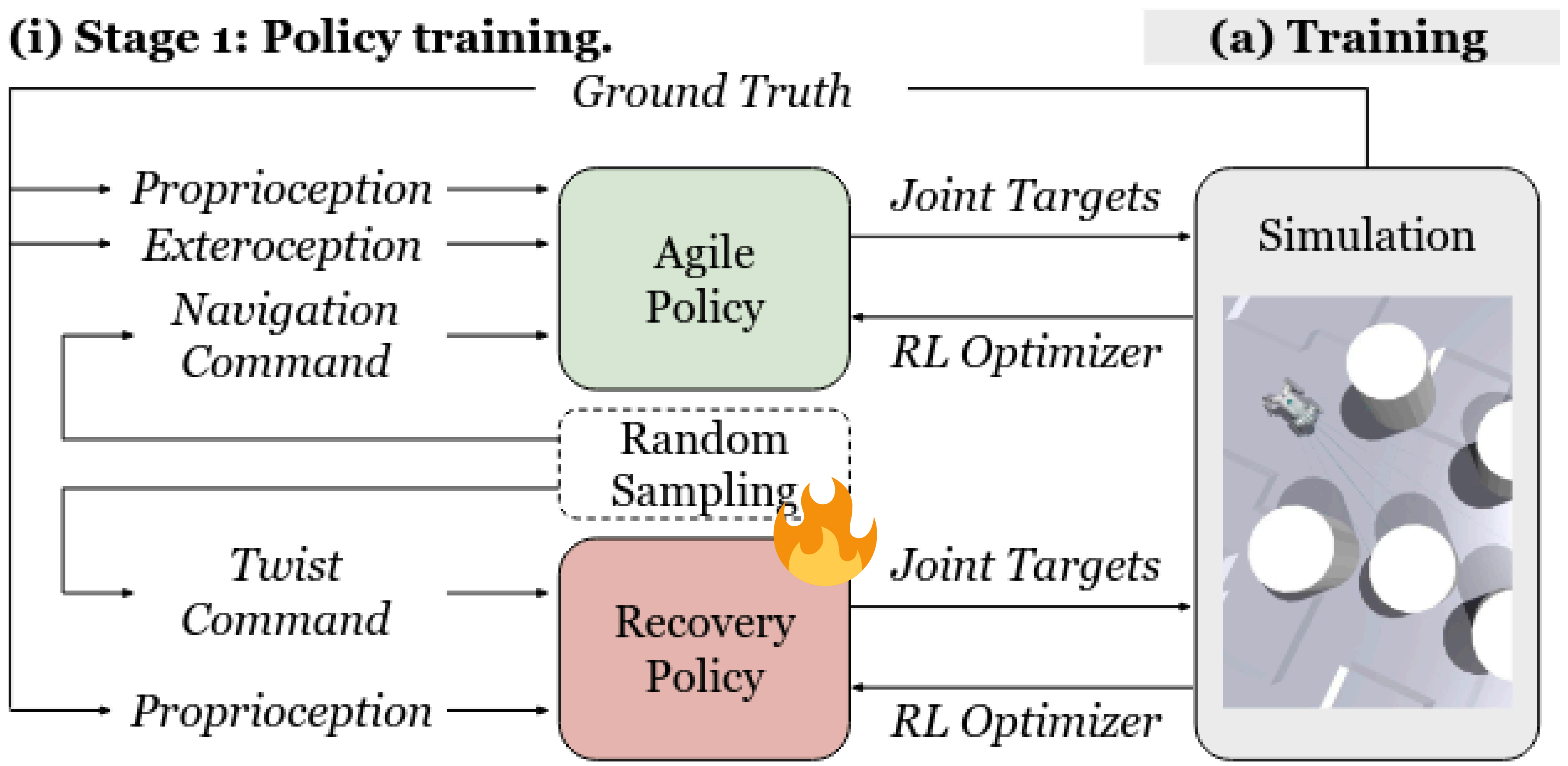


Fig. 8. An example case in simulation where π^{Agile} fails to reach the goal. a) Trajectories of ABS and other baselines, with RA values visualized for ABS. b) The velocity-time curves showing that ABS is much faster than the LAG baseline. c) Illustrations of the RA value landscape when the recovery policy is triggered at (I) and (II), projected in the $v_x - \omega_z$ plane and the $v_x - v_y$ plane. We show the initial twist before search (*i.e.*, the current twist of the robot base) and the searched commands based on Equation (21).


(i) Stage 1: Policy training.
(a) Training
Ground Truth
Proprioception
Exteroception
Navigation Command
Agile Policy
Joint Targets
RL Optimizer
Simulation
Random Sampling
Twist Command
Recovery Policy
Joint Targets
RL Optimizer
Proprioception

Recovery Policy

memo

- make the robot track a given twist command as fast as possible

Observation

$$\left.\begin{array}{ll}
\bullet\ \text{foot contacts:} & \boldsymbol{c}_{f\in\{1,2,3,4\}} \\
\bullet\ \text{base angular velocities:} & \boldsymbol{\omega} \\
\bullet\ \text{projected gravity:} & \boldsymbol{g} \\
\bullet\ \text{twist commands(only non-zero)} & \boldsymbol{tw}^c = \left[v_x^c, v_y^c, 0, 0, 0, \omega_z^c\right] \\
\bullet\ \text{joint positions/velocities:} & \boldsymbol{q}\ \ \dot{\boldsymbol{q}} \\
\bullet\ \text{previous actions:} & \boldsymbol{a}
\end{array}\right\} \boldsymbol{o}^{\mathrm{Rec}}$$

Action

- 12-d joint targets
- PD controller tracks these joint targets
- fully-connected MLP

Recovery Policy

memo

- make the robot track a given twist command as fast as possible

Reward

- Penalty
- Task
- Regularization

$$r = r_{\text{penalty}} + \underline{r_{\text{task}}} + r_{\text{regularization}}$$

$$r_{\text{task}} = 10 \cdot r_{\text{linvel}} - 0.5 \cdot r_{\text{angvel}} + 5 \cdot r_{\text{alive}} - 0.1 \cdot r_{\text{posture}}$$

$$r_{\text{linvel}} = \exp\left[-\frac{(v_x - v_x^c)^2 + \left(v_y - v_y^c\right)^2}{\sigma_{\text{linvel}}^2}\right]$$

$$r_{\text{angvel}} = \|\omega_z - \omega_z^c\|_2^2$$

$$r_{\text{alive}} = 1 \cdot \mathbf{1}(\text{ alive })$$

$$r_{\text{posture}} = \|q - \bar{q}_{\text{ree}}\|_1$$

Recovery Policy

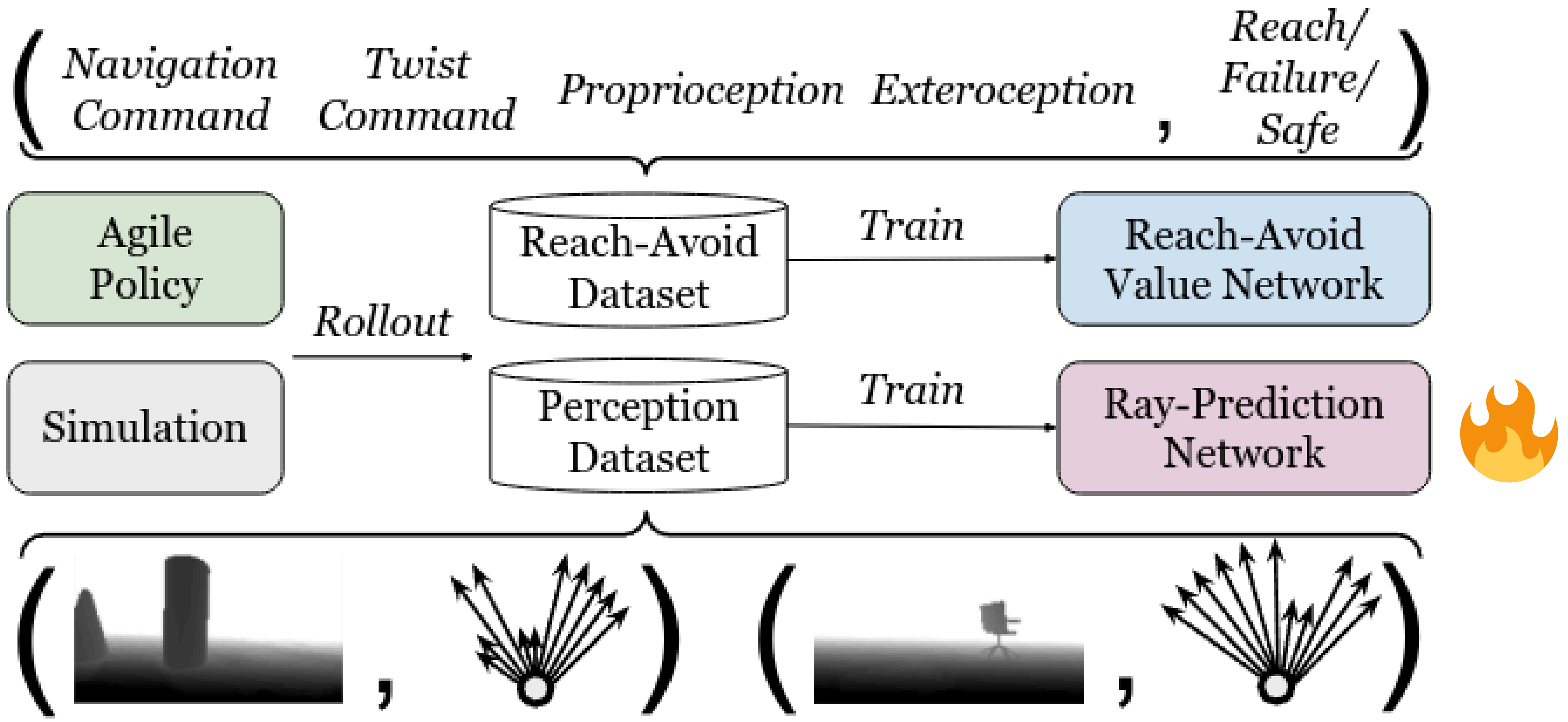
memo

- make the robot track a given twist command as fast as possible

Simulation Training

- Similar to Agile policy setting
- episode length is changed to 2 s
- some DR ranges are modified - these changes better accommodate the states that can trigger the recovery policy during the agile running.

(ii) Stage 2: Network training from agile policy rollout data.



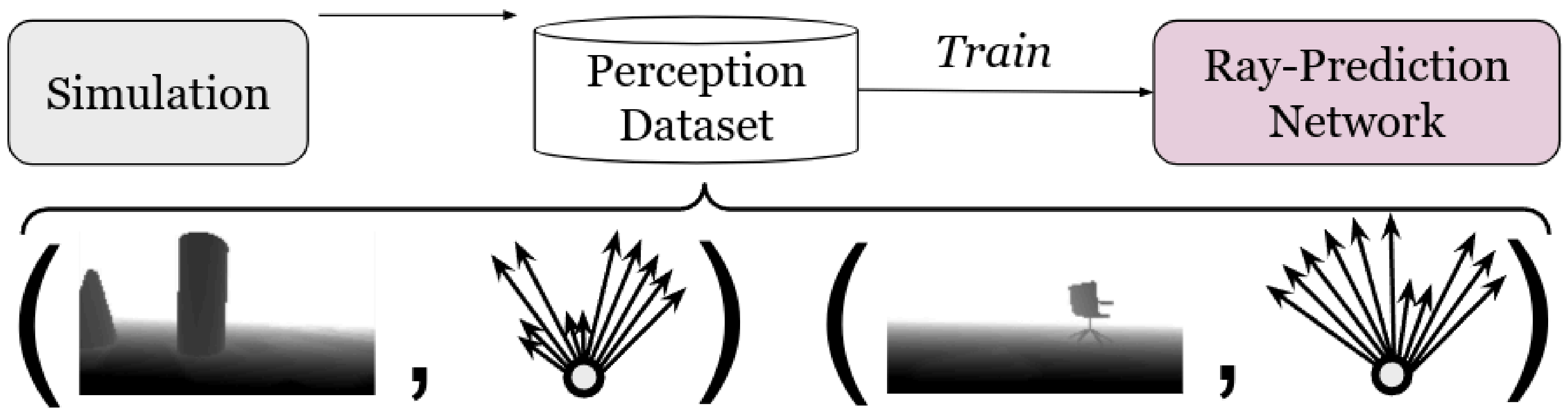
Perception

memo

- both the agile policy and the RA value network use the **exteroceptive 11-d ray distances** as part of the observations, with access to their ground truth values during training

Ray-prediction network

- collect a dataset of pairs of depth images and ray distances
- Data Augmentation for Sim-to-Real
- To make the network focus more on close obstacles, take the logarithm of depth values as the NN inputs
- ResNet-18 with pretrained weights



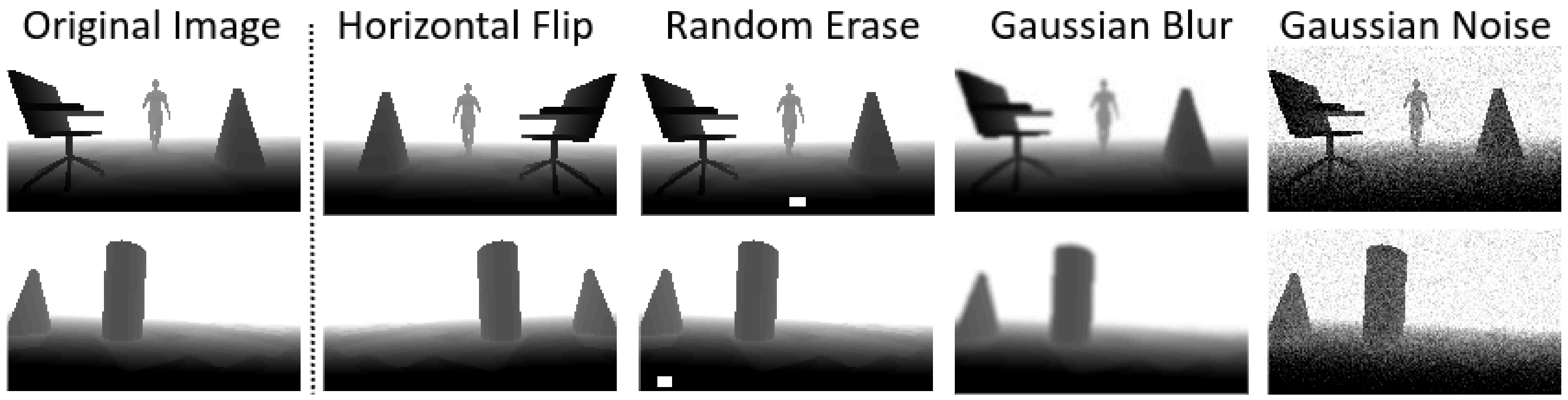


Fig. 6. Illustration of four kinds of image augmentation used for depth-based ray-prediction training.

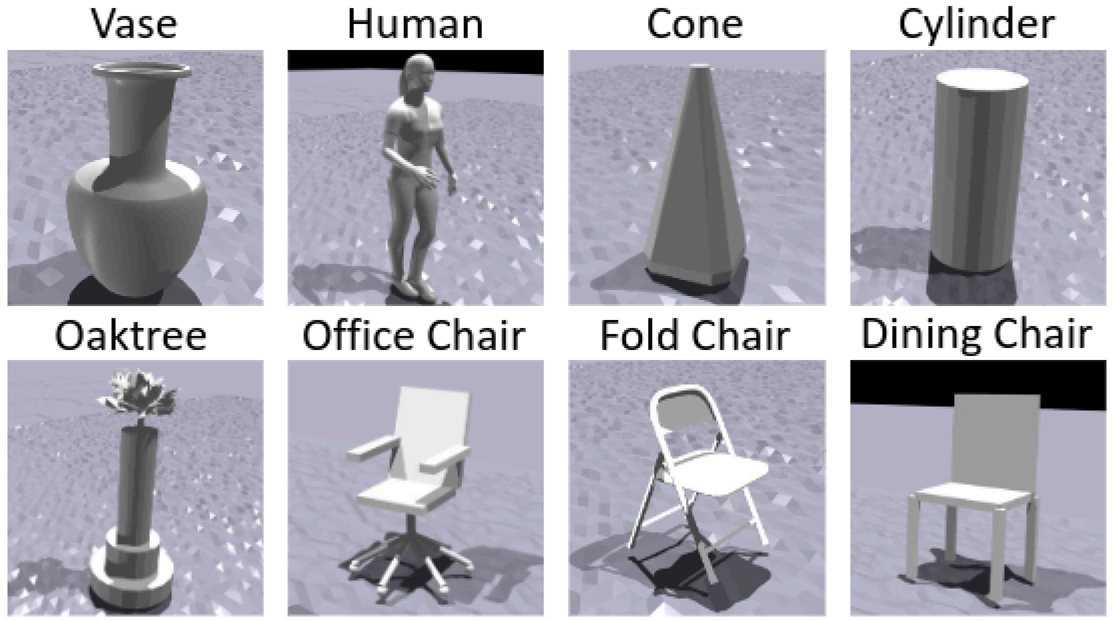
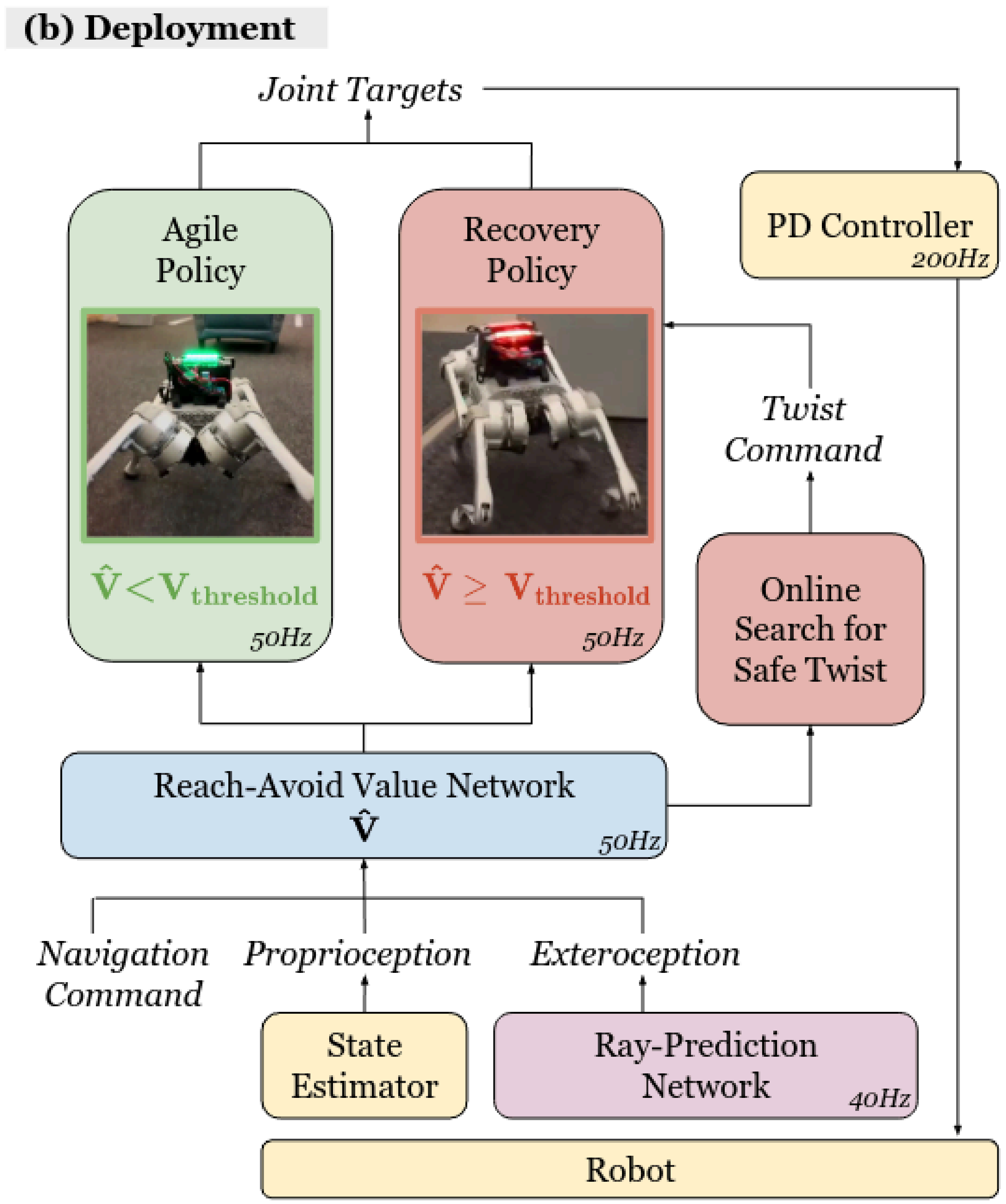


Fig. 5. Various obstacles used for ray-prediction data collection.

Deployment

Real world


(b) Deployment
Joint Targets
Agile Policy
$\hat{V} < V_{threshold}$
50Hz
Recovery Policy
$\hat{V} \geq V_{threshold}$
50Hz
PD Controller
200Hz
Twist Command
Online Search for Safe Twist
Reach-Avoid Value Network
$\hat{V}$
50Hz
Navigation Command
Proprioception
Exteroception
State Estimator
Ray-Prediction Network
40Hz
Robot

Experiments

Baselines

memo

1. **ABS** system, with both the agile policy and the recovery policy
2. Our agile policy π **Agile only**
3. **LAG**: we use PPO-Lagrangian to train end-to-end safe RL policies with the agile policy's formulation

- **Simulation**
  - 3 variants for each setting:
    - an aggressive one (**"-a"**) doubling the agile reward term r_{agile}
    - a nominal one (**"-n"**)
    - a conservative one (**"-c"**) halving the r_{agile}
  - distribute eight obstacles within a 5.5 m× 4 m rectangle (during training it was 11 m× 5 m)

BENCHMARKED COMPARISON IN SIMULATION

	Success Rate (%)	Collision Rate (%)	Timeout Rate (%)	$\bar{v}_{\text{peak}}$ of Success (m/s)	$\bar{v}$ of Success (m/s)
ABS-a	78.9±1.4	4.4±0.5	16.7±1.9	3.74±0.02	2.15±0.04
ABS-n	79.1±4.4	5.7±2.9	15.2±2.1	3.48±0.06	2.08±0.01
ABS-c	**85.8±5.6**	**2.9±0.7**	11.3±5.1	2.98±0.12	1.87±0.03
π^{Agile}-a	73.3±4.3	26.1±4.4	**0.6±0.1**	**3.83±0.03**	**2.55±0.03**
π^{Agile}-n	77.3±4.2	21.7±3.9	1.0±0.4	3.55±0.04	2.39±0.04
π^{Agile}-c	**83.2±1.7**	15.5±2.0	1.3±0.6	3.04±0.13	2.04±0.08
LAG-a	**82.5±6.0**	10.9±2.6	6.6±4.5	2.70±0.13	1.69±0.09
LAG-n	77.4±11.5	9.1±1.8	13.5±13.0	2.45±0.07	1.41±0.03
LAG-c	49.1±8.4	7.4±2.7	43.5±11.1	2.45±0.10	1.12±0.08

*Bold values: the mean falls within the range of top1's mean ± top1's std.

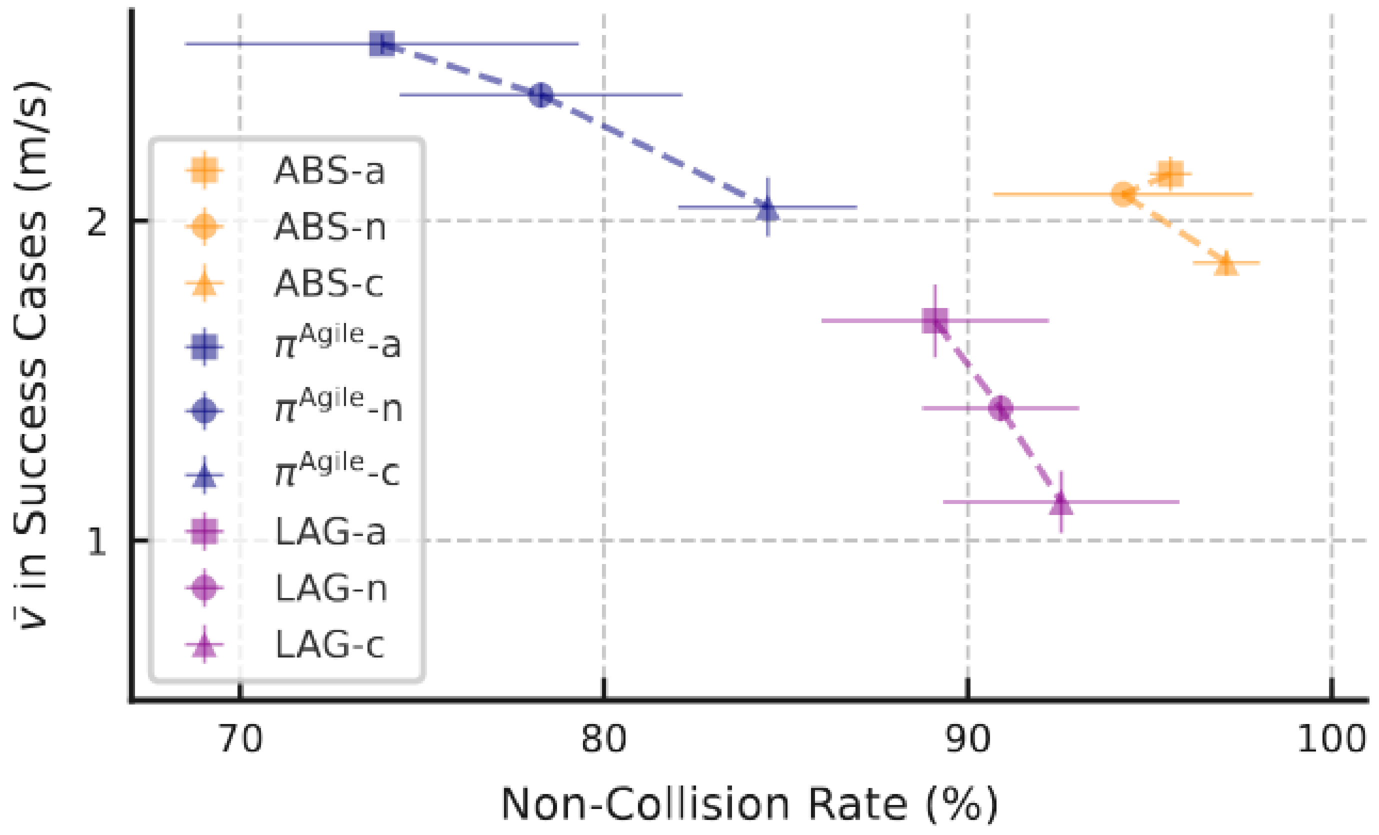


Fig. 7. Illustration of agility-safety trade-off in benchmarked comparison. Agility is quantified by the average speed achieved in success cases while safety is represented by the non-collision rate. Points indicate the mean values, and error bars indicate the std values.

Baselines

memo

1. **ABS** system, with both the agile policy and the recovery policy
2. Our agile policy π **Agile only**
3. **LAG**: we use PPO-Lagrangian to train end-to-end safe RL policies with the agile policy's formulation

- **Simulation**
  - Example Case
    - starting from (0, 0) needs to run through 8 obstacles to reach the goal (7, 0)
    - first go through an open space, followed by two tight spaces, and then another open space
    - ABS runs fast in the open spaces, and **slows down in the tight spaces for safety thanks to the shielding of RA values and the recovery policy**

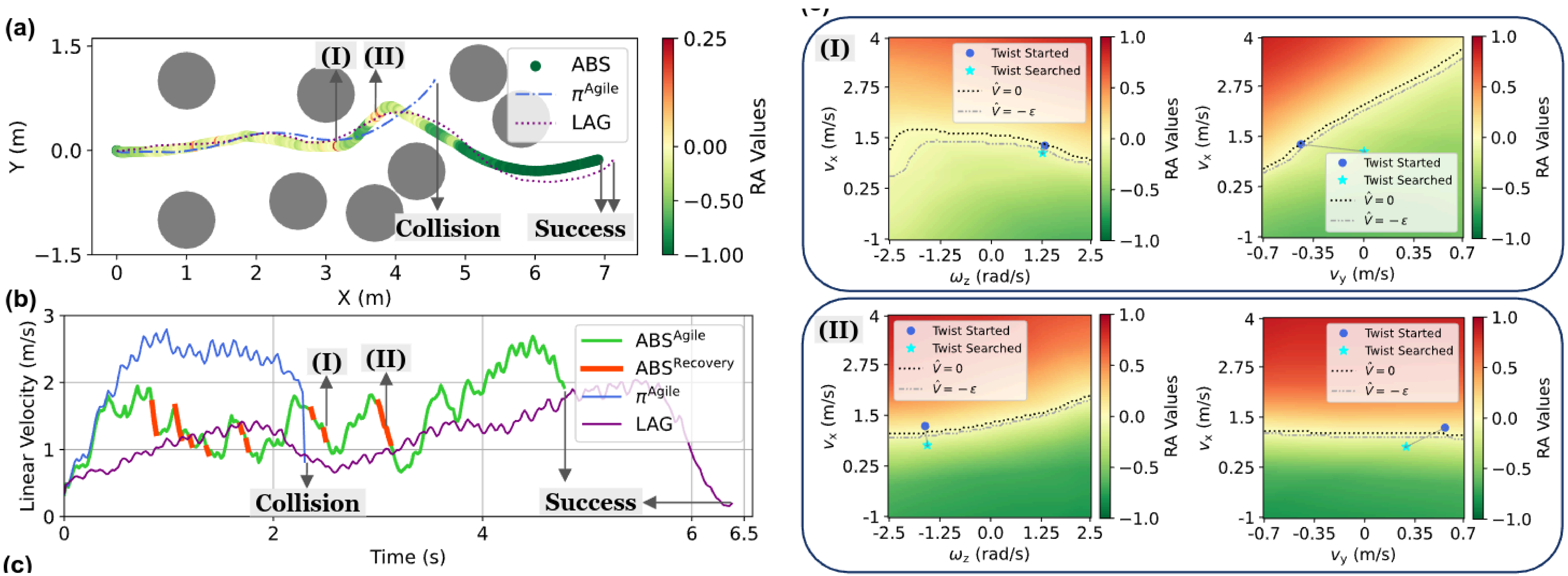


Fig. 8. An example case in simulation where π^{Agile} fails to reach the goal. a) Trajectories of ABS and other baselines, with RA values visualized for ABS. b) The velocity-time curves showing that ABS is much faster than the LAG baseline. c) Illustrations of the RA value landscape when the recovery policy is triggered at (I) and (II), projected in the $v_x - \omega_z$ plane and the $v_x - v_y$ plane. We show the initial twist before search (*i.e.*, the current twist of the robot base) and the searched commands based on Equation (21).

Baselines

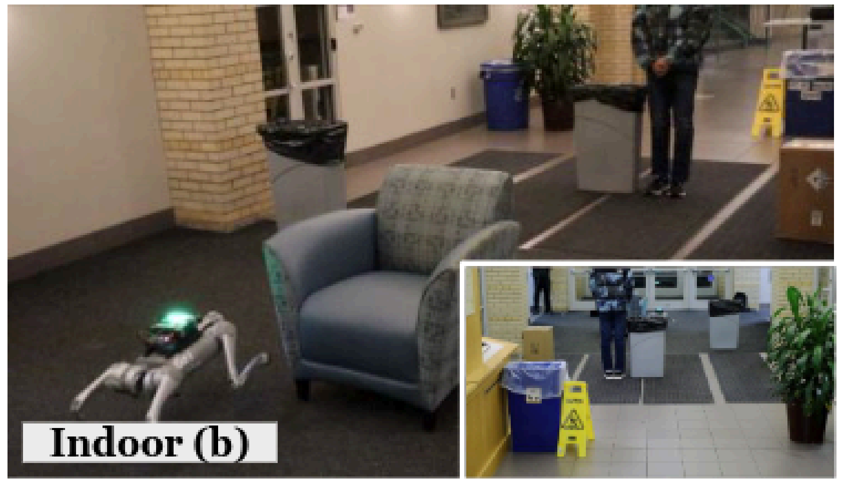
memo

1. **ABS** system, with both the agile policy and the recovery policy
2. Our agile policy π **Agile only**
3. **LAG**: we use PPO-Lagrangian to train end-to-end safe RL policies with the agile policy's formulation

- **Real-World**
  - HW setup
    - Go1, Orin NX, Zed Mini Stereo Camera
  - two indoor and one outdoor testbeds


Indoor (a)

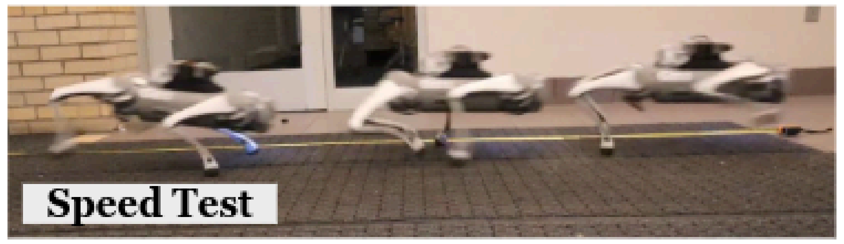
	Success	Collision	Time Cost
ABS	9/10	1/10	5.91 s
ABS (only π^{Agile})	7/10	3/10	5.06 s
LAG	8/10	2/10	6.80 s


Indoor (b)

	Success	Collision	Time Cost
ABS	10/10	0/10	4.75 s
ABS (only π^{Agile})	7/10	3/10	3.74 s
LAG	9/10	1/10	6.13 s


Outdoor

	Success	Collision	Time Cost
ABS	10/10	0/10	4.46 s
ABS (only π^{Agile})	9/10	1/10	4.15 s
LAG	9/10	1/10	6.05 s


Speed Test

	ABS	LAG
Peak Speed	3.1 m/s	2.1 m/s

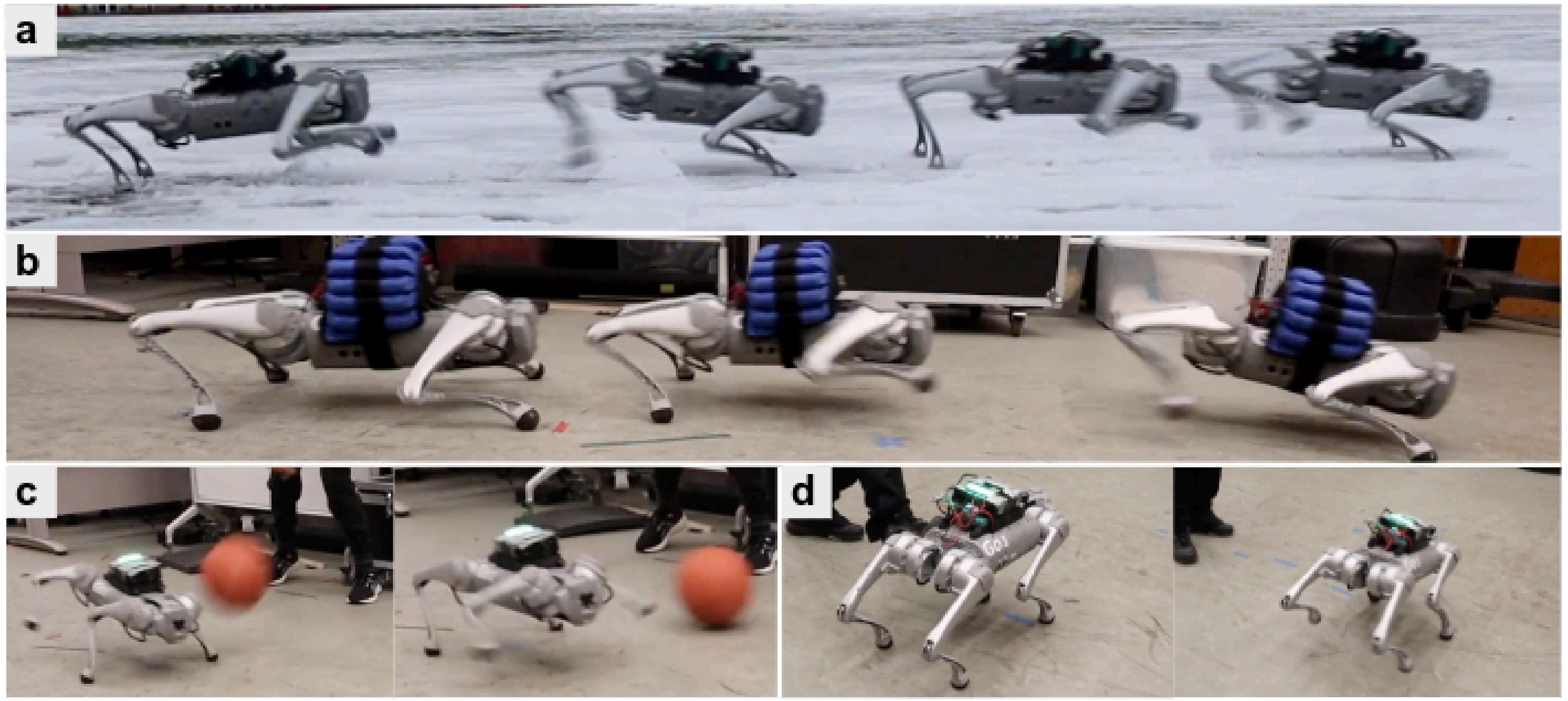


Fig. 10. Robustness Tests of our ABS system, a) in snowy terrain,s b) bearing a 12-kg payload, c) against a ball hit when running, and d) withstand a kick when standing at the goal.

EXTENSIVE STUDIES AND ANALYSES

Maximizing Agility

memo

- **Goal-Reaching v.s. Velocity-Tracking**
  - 대부분 velocity tracking 방식을 취함
  - goal reaching is a better choice because it does **not decouple locomotion and navigation** for collision avoidance and can fully unleash the agility that is learned

TABLE IV
GOAL-REACHING POLICY V.S. VELOCITY-TRACKING POLICY

Term	Our π^{Agile}	Rapid [48]
Gait patterm	**gallop**	near trot
Max #. uncontrollable DoFs	**1**	3
Peak vel. in simulation	4.0 m/s	**4.1 m/s**
Peak torque in simulation	**23.5 Nm**	35.5 Nm
Peak joint vel. in simulation	**22.0 rad/s**	30.0 rad/s
Peak vel. in real world	**3.1 m/s**	2.5 m/s
Collision avoidance	**as trained**	need high-level commands
Fully unleashed agility	**as trained**	non-trivial for high level
Changing vel. for steering	**in distribution**	out of distribution
Curriculum learning	**straightforward**	carefully designed

Maximizing Agility

memo

- **Effects of illusion and ERFI-50 randomization**
  - Two key components we add in domain randomization
  - **Without the illusion,** the robot will sometimes tremble near a wall which it has never seen in simulation
  - **Without ERFI-50,** the robot will hit the ground with its head during running due to the sim-to-real gap in motor dynamics

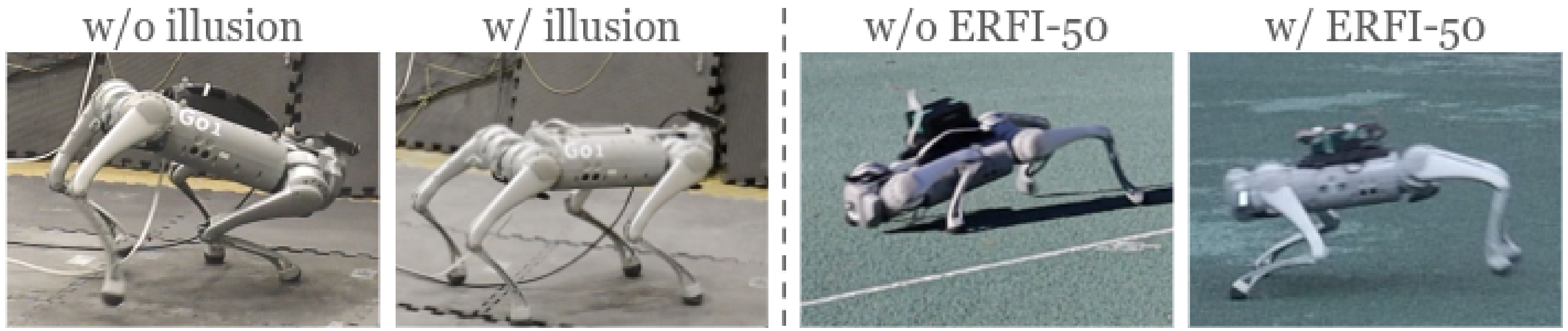


Fig. 11. Effects of illusion and ERFI-50 randomization. The robot will tremble near a wall without illusion randomization and will hit the ground during running without ERFI-50 randomization.

Extensive Studies on RA Values

memo

- **Selecting safety threshold**
  - For safety shielding, we choose Vthreshold = −0.05
  - Scanning Vthreshold from −0.001 to −0.1 brings no significant change in the overall performance
  - the collision rate slightly decreases as expected whereas the success rate also slightly decreases
- **Soft Lipschitz continuity for the failure indicator**
  - soften the discrete collision indicator to approach the Lipschitz continuity
  - significantly enhances the safety of our system while slightly increasing the conservativeness

TABLE V
EFFECTS OF DIFFERENT $V_{\text{THRESHOLD}}$ ON ABS

$V_{\text{threshold}}$	-0.001	-0.01	-0.05	-0.1
Success Rate (%)	78.0±2.1	78.1±3.4	79.1±4.4	75.8± 2.0
Collision Rate (%)	5.0±0.6	5.8±1.9	5.7±2.9	4.3±0.6
$\bar{v}_{\text{peak}}$ of Success (m/s)	3.42±0.06	3.46±0.08	3.48±0.06	3.42±0.05
$\bar{v}$ of Success (m/s)	2.08±0.02	2.08±0.01	2.08±0.01	2.05± 0.03

0.1 is considered big given that
^V is bounded between −1 and1

TABLE VI
EFFECTS OF SOFTENED FAILURE INDICATOR ON ABS

	ABS w/ softened ζ	ABS w/o softened ζ	π^{Agile}
Success Rate (%)	79.1±4.4	**81.7±1.3**	77.3±4.2
Collision Rate (%)	**5.7±2.9**	14.7±1.5	21.7±3.9
$\bar{v}_{\text{peak}}$ of Success (m/s)	3.48±0.06	3.45±0.06	**3.55±0.04**
$\bar{v}$ of Success (m/s)	2.08±0.01	2.27±0.03	**2.39±0.04**

Enhancing Perception Training

memo

- **Several factors**
  - network architecture
  - **pretrained weights**
  - **data augmentation**
- For real-time high-speed locomotion, we opt for **ResNet-18**, balancing accuracy and responsiveness in dynamic environments

TABLE VIII
PERFORMANCE METRICS FOR DIFFERENT NETWORK ARCHITECTURES AND TRAINING APPROACHES

Architecture	Test Set MSE	Inference Time (ms)
EfficientNet-B0*	3.627×10^{-2}	19
MobileNet-V2*	3.387×10^{-2}	15
ResNet-34	3.081×10^{-2}	14
ResNet-18	3.238×10^{-2}	9
ResNet-18 (w/o pretraining)	3.526×10^{-2}	9
ResNet-18 (w/o augmentation)	3.393×10^{-2}	9

* We use the PyTorch-ONNX pipeline where the implementations of these network architectures may be not fully optimized.

Failure Cases and Limitations

memo

- when **the obstacles are too dense** and form a local minimum
- The RA values are **learned with static obstacles**, and can only generalize to quasi-static environments
  - predict the motions of the obstacles in the future
- limit the robot behaviors to **only 2D locomotion** and constrain the motions to have no flying phase
  - For 3D terrains such as stairs and gaps
- **implicit system identification** techniques
- **vision system** needs further improvement
  - Indoor (a) testbed, the only collision of ABS is due to the "undetected" objects by the ray-prediction network <u>as the corridor is quite dim.</u>

END