DODGE-IT: Learning Collision-Aware VLA Models for Robotic Manipulation

Jiawei Feng[1*] **Chuzhao Huang**[1,*] **Xinli Xu**[1] **Ying-Cong Chen**[1,2†]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]The Hong Kong University of Science and Technology
jfeng644@connect.hkust-gz.edu.cn, chuzhaohuang@hkust-gz.edu.cn
yingcongchen@ust.hk

Abstract

Vision-Language-Action (VLA) models are increasingly used as general-purpose policies for robotic manipulation, but completing the instructed task requires more than reaching the target state. A policy must also avoid nearby objects throughout the rollout, including approach, interaction, transport, and handover. A common approach is to invoke conventional collision-avoidance mechanisms, such as motion planners or safety filters, which provide effective geometric safeguards. However, the resulting system may struggle to produce actions that are simultaneously safe and task-progressing. When collision avoidance and manipulation progress require misaligned actions, the system may avoid the obstacle while slowing, disrupting, or even failing the task. To address this, we introduce DODGE-IT, a collision-aware learning framework that leverages collision signals as policy-improvement feedback to strengthen the VLA policy's collision-aware manipulation capability. DODGE-IT first designs Task-Relevant Assessment of Collision Events (TRACE) that detects and identifies task-relevant obstacles from RGB-D observations and turns executions into collision outcomes. These outcomes are then incorporated together with task success signals into representative reinforcement learning frameworks for VLA. We validate DODGE-IT on SafeLIBERO and real-robot manipulation tasks, showing improved obstacle avoidance while preserving task completion across simulation and real-robot settings.

1 Introduction

Consider a robot assistant controlled by a goal-driven Vision-Language-Action (VLA) policy, operating on a cluttered desk. After taking a sip, a user casually places a cup beside the laptop, then asks, "Could you hand me the red book?" The instruction is simple, and the target is visible. Yet after grasping the book, the assistant may return it along a direct path that brushes the cup. This is a collision-awareness failure: the policy makes task progress, but not in a way that respects the surrounding workspace. As VLA models become increasingly capable general-purpose robot policies [Brohan et al., 2023, Kim et al., 2024, Black et al., 2024], collision-aware execution becomes a necessary manipulation capability.

A common route for collision avoidance is to place a geometric safeguard around a task policy. Motion planners and learned motion policies can generate collision-free motions for manipulation [Yamada et al., 2021, Fishman et al., 2023]. Safety controllers and constraint-based action filters can instead constrain or project learned actions during execution [Chow et al., 2020, Hu et al., 2025]. These safeguards are effective and remain an important part of robotic systems. However, external

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

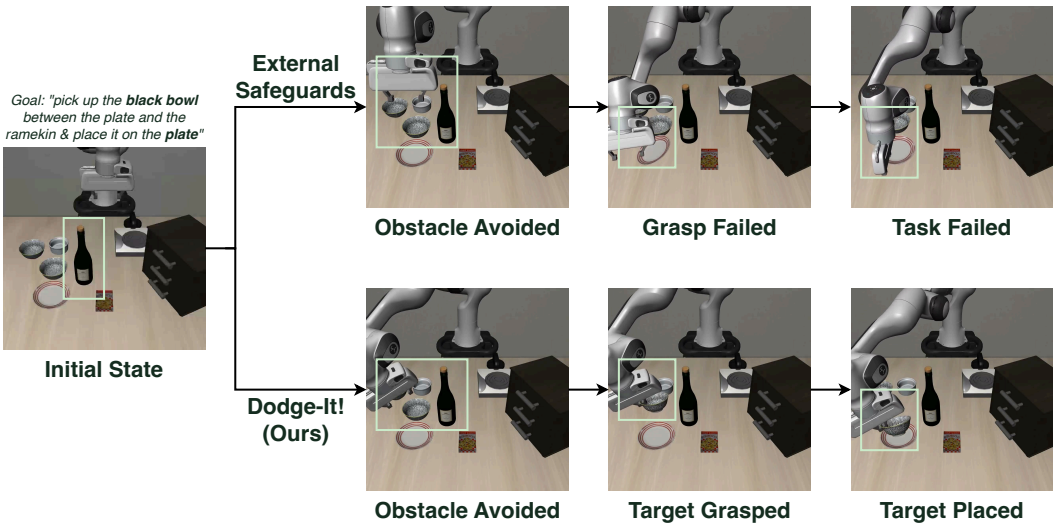


Figure 1: Example comparison between external safeguards and DODGE-IT. From the same initial state and language goal, the external-safeguard rollout avoids the obstacle but leaves the gripper offset from the target bowl during the approach, and the subsequent grasp misses. DODGE-IT avoids the obstacle while keeping the gripper aligned with the bowl, then completes the grasp-and-place sequence.

safeguards can make the current action safer without necessarily making it task-progressing as the task behavior and collision avoidance are optimized as separate modules. When the action changes required for collision avoidance and manipulation progress are misaligned, the system may avoid the obstacle while slowing, disrupting, or even failing the task. As illustrated in Figure 1, the safeguard avoids the obstacle but interferes with the approach to the target, a black bowl, leading to a failed grasp and task failure.

Unlike approaches that directly enforce collision avoidance through external safeguards, DODGE-IT aims to leverage collision signals as policy-improvement feedback to strengthen the VLA policy's collision-aware manipulation capability. To obtain such feedback, we design Task-Relevant Assessment of Collision Events (TRACE) to detect and identify task-relevant obstacles from RGB-D observations and assess whether a rollout causes collision events. DODGE-IT then incorporates TRACE-derived collision outcomes together with task success signals into representative reinforcement learning frameworks for VLA, such as PPO and RECAP [Lu et al., 2025, Physical Intelligence et al., 2025].

We validate DODGE-IT in both simulation and real-robot settings. In simulation, we follow the standard SafeLIBERO protocol [Hu et al., 2025] and evaluate across four safety-critical manipulation suites. In real-robot experiments, we further evaluate DODGE-IT over multiple manipulation tasks and obstacle configurations. Experiments show that DODGE-IT improves obstacle avoidance while preserving task completion across simulated and real-robot manipulation.

We make the following contributions:

- We design TRACE, a task-relevant collision-event assessment module that detects and identifies task-relevant obstacles from RGB-D observations and turns executions into collision outcomes.
- We propose DODGE-IT, which incorporates TRACE-derived collision outcomes and task success signals into reinforcement-learning optimization of VLA policies, instantiated with PPO in resettable simulation and RECAP in real-robot experiments.
- We validate DODGE-IT on SafeLIBERO and real-robot manipulation tasks, showing improved obstacle avoidance while preserving task completion across simulation and real-robot settings.

2 Related Work

VLA policy improvement beyond imitation. Large VLA policies connect visual observations, language instructions, and robot actions in a single policy interface [Brohan et al., 2023, Kim et al., 2024, Black et al., 2024]. Subsequent work studies how such policies can be adapted or improved beyond their initial imitation-trained behavior. OpenVLA-OFT analyzes design choices for efficient fine-tuning [Kim et al., 2025], while VLA-RL, RIPT-VLA, ConRFT, RobustVLA, and RECAP/$\pi^*_{0.6}$ improve VLA policies through rollout, sparse success rewards, reinforced fine-tuning, robustness-aware regularization, or advantage-conditioned real-robot experience [Lu et al., 2025, Tan et al., 2025, Chen et al., 2025, Zhang et al., 2025b, Physical Intelligence et al., 2025]. This line establishes that VLA policies can be improved from feedback after pretraining. DODGE-IT shares this policy-improvement perspective, but focuses on collision-aware manipulation, which requires completing the instructed task while avoiding nearby obstacles throughout execution.

External safeguards, motion planning, and guidance. Another family of methods improves safety or task execution by adding mechanisms around a policy. VLSA/AEGIS introduces SafeLIBERO and applies a plug-and-play safety constraint layer to VLA actions [Hu et al., 2025]. OmniGuide steers generalist policies at inference time using differentiable guidance fields from external models [Song et al., 2026]. Outside the VLA setting, neural motion policies such as MπNets, Avoid Everything, and Deep Reactive Policy learn reactive collision-free motion generation from depth or point-cloud observations [Fishman et al., 2023, 2025, Yang et al., 2025], while planner-augmented RL methods such as MoPA-RL and its follow-up distillation work use motion planning to support obstructed manipulation [Yamada et al., 2021, Liu et al., 2022]. These methods are valuable because they provide geometric safeguards, guidance, or collision-free motion behavior around a task policy. Their design also highlights the central distinction of DODGE-IT: external safeguards and guidance act as separate mechanisms around the task policy, whereas DODGE-IT uses collision feedback to improve the policy's own task-collision trade-off.

Safety and feasibility feedback for policy learning. Safety and feasibility can also enter robot learning as constraints, auxiliary supervision, or recovery feedback. SafeVLA studies VLA safety alignment with constrained learning across navigation and manipulation [Zhang et al., 2025a]. Explicit Physical Feasibility adds geometry-grounded feasibility supervision to VLA learning [Wei et al., 2026], while Safe-Night VLA uses thermal perception and runtime filtering for safety-critical manipulation [Yu et al., 2026]. Outside VLA, Recovery RL and safe policy optimization study how constraint information and recovery mechanisms can guide policy learning under safety requirements [Thananjeyan et al., 2021, Chow et al., 2020]. Compared to these approaches above, DODGE-IT studies how VLA policies can learn to avoid nearby obstacles while completing manipulation tasks, using task-relevant collision feedback from robot executions.

3 DODGE-IT: Learning Collision-Aware VLA for Robotic Manipulation

DODGE-IT trains VLA policies to complete manipulation tasks while avoiding nearby obstacles. As shown in Figure 2, TRACE identifies task-relevant obstacles and assesses collision events from RGB-D observations of robot executions. The resulting collision outcomes are incorporated together with task success signals into policy improvement, instantiated with PPO in simulation and RECAP-style learning in real-robot experiments.

3.1 Problem Setup

We consider language-conditioned manipulation in scenes containing objects that should not be contacted. At timestep t, the robot observes $s_t = (o_t, q_t)$, where o_t contains visual observations and q_t is the proprioceptive state, and a VLA policy $\pi_\theta(a_t \mid s_t, l)$ outputs a continuous robot action under language instruction l. Executing the policy produces a trajectory $\tau = (s_0, a_0, \ldots, s_T)$.

We optimize the policy for task completion while incorporating collision feedback to encourage obstacle avoidance. The task signal $y_{\text{task}}(\tau)$ indicates whether the trajectory completes the language instruction. Collision feedback can be represented at the step level as $c_t \in \{0, 1\}$, indicating whether the execution contacts or disturbs task-relevant obstacles at timestep t. We set the trajectory-level

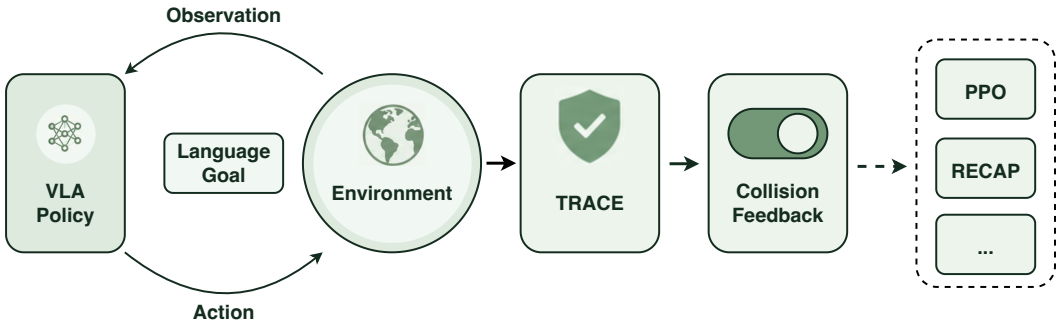


Figure 2: Overview of DODGE-IT. A VLA policy interacts with the environment under a language goal, while Task-Relevant Assessment of Collision Events (TRACE) converts rollout observations into collision outcomes. DODGE-IT combines these outcomes with task success signals and uses them in reinforcement-learning optimization of the VLA policy, instantiated with PPO [Schulman et al., 2017] in resettable simulation and RECAP [Physical Intelligence et al., 2025] in real-robot experiments.

collision outcome to $y_{\text{col}}(\tau) = \max_t c_t$, where a value of 1 indicates that at least one collision event occurs during the trajectory.

3.2 TRACE: Task-Relevant Assessment of Collision Events

To learn obstacle avoidance during manipulation, DODGE-IT needs collision feedback from robot executions. We design TRACE to obtain this feedback by identifying task-relevant obstacles and assessing collision events from RGB-D observations. As shown in Figure 3, the system first adapts the obstacle detector to the deployment scene. At runtime, TRACE consists of two modules: an obstacle detection module that selects task-relevant obstacles from visual candidates, and a collision assessment module that turns multiview RGB-D observations into collision outcomes.

Scene adaptation for obstacle detection. For a deployment scene, the system uses VLM point annotations and SAM mask prediction to generate object masks, which are then used to fine-tune a YOLO detector for the scene. This stage gives the runtime detector reliable candidate masks without requiring dense manual mask annotation.

Obstacle detection module. At runtime, the YOLO detector proposes candidate object masks from multiview RGB observations. Since not every visible object is an obstacle for a given instruction, TRACE uses the language goal to select task-relevant obstacles from these candidates. This produces an obstacle mask M_t for the objects whose contact should be penalized during the current task.

Collision assessment module. Given multiview RGB-D observations, TRACE back-projects the selected obstacle mask into 3D to obtain an obstacle point cloud,

$$P_t^{\text{obs}} = \pi^{-1}(M_t, D_t), \tag{1}$$

and fuses multiview observations to obtain robot and obstacle point clouds. A collision event is detected using two complementary cues. First, if a tabletop obstacle should remain static, unintended displacement provides direct evidence of physical contact. We estimate the median inter-frame obstacle displacement

$$\Delta_t^{\text{obs}} = \text{median}_{p \in P_t^{\text{obs}}} \left\| p - \mathcal{F}(p, P_{t-1}^{\text{obs}}) \right\|_2, \tag{2}$$

where $\mathcal{F}$ maps a current obstacle point to its correspondence in the previous frame. Second, TRACE computes the robot-obstacle proximity

$$d_t^{\text{prox}} = \min_{p \in P_t^{\text{obs}},\, r \in P_t^{\text{rob}}} \|p - r\|_2. \tag{3}$$

The per-step collision indicator is

$$c_t = \mathbf{1}\left[\Delta_t^{\text{obs}} > \tau_{\text{disp}} \ \vee \ d_t^{\text{prox}} < \tau_{\text{prox}}\right], \tag{4}$$

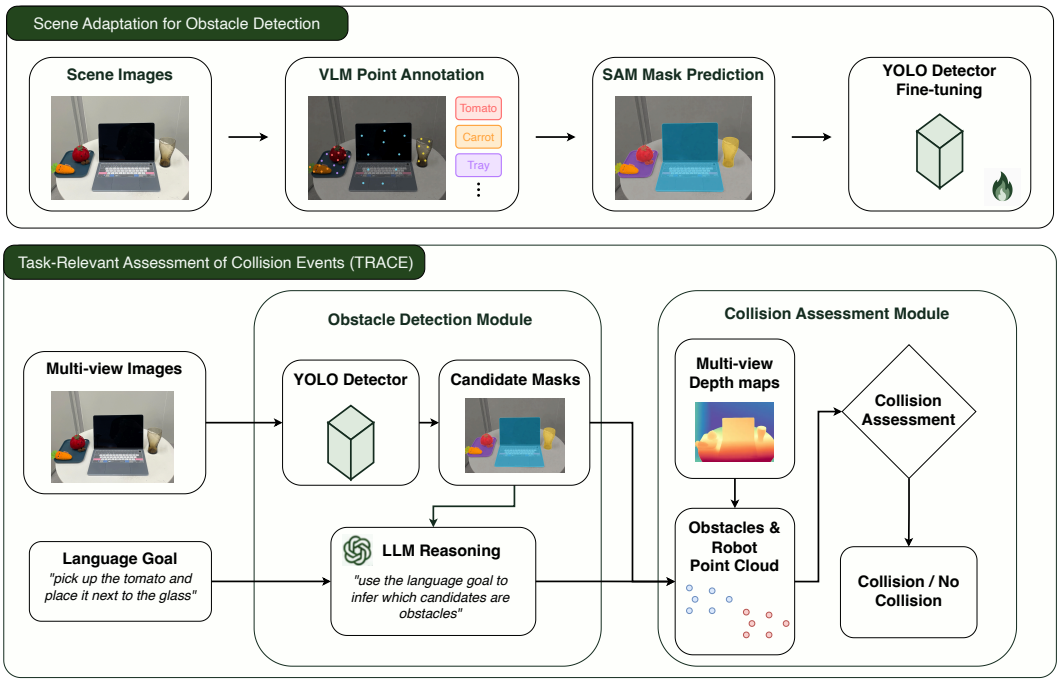


Figure 3: Overview of TRACE. The scene-adaptation stage uses VLM point annotations and SAM mask prediction to fine-tune the YOLO detector for the deployment scene. At runtime, TRACE consists of an obstacle detection module and a collision assessment module. The obstacle detection module uses YOLO candidate masks and language-conditioned LLM reasoning to identify task-relevant obstacles, while the collision assessment module fuses multiview RGB-D observations into robot and obstacle point clouds and outputs collision outcomes from obstacle displacement and robot-obstacle proximity.

and the trajectory-level collision outcome is $y_{\text{col}}(\tau) = \mathbf{1}[\max_t c_t = 1]$. The displacement term captures actual unintended obstacle motion, while the proximity term detects near-contact events that may be visually occluded or not yet produce measurable displacement.

3.3 Collision-Aware Reinforcement Learning Objective

DODGE-IT incorporates TRACE-derived collision outcomes together with task success signals into reinforcement-learning optimization of the VLA policy. Let r_t^{task} denote the task reward and c_t the collision indicator from TRACE. In simulation, we use a shaped reward

$$r_t = r_t^{\text{task}} - \lambda c_t, \tag{5}$$

where λ controls the penalty for collision events. A policy-gradient optimizer can then update π_θ using advantages estimated from this reward. In real-robot experiments, task success outcomes and collision outcomes provide trajectory-level feedback for scoring or conditioning collected executions during offline policy improvement. Thus, task and collision feedback is shared across regimes, while the optimizer depends on the data-access setting.

3.4 Simulation Instantiation: PPO

In resettable simulation, DODGE-IT uses online rollouts to optimize the VLA with PPO. Starting from a $\pi_{0.5}$ base policy, we first perform supervised fine-tuning on the original LIBERO demonstrations to obtain a task-capable initialization. The policy is then deployed in SafeLIBERO tasks, where each rollout provides RGB-D observations, TRACE-derived collision indicators, and task success rewards. PPO updates the VLA policy using the reward in Eq. (5), encouraging obstacle avoidance without removing the task-success objective.

3.5 Real-Robot Instantiation: RECAP

In real-robot experiments, DODGE-IT uses the same TRACE-derived collision outcomes but updates the VLA through RECAP-style offline or advantage-conditioned learning. Starting from a $\pi_{0.5}$ policy supervised-fine-tuned with real-robot demonstrations, we collect executions in obstacle settings. For each trajectory, we obtain a task success outcome and use TRACE to determine the collision outcome. These outcomes provide task and collision feedback for advantage estimation and advantage-conditioned policy improvement.

4 Experiments

4.1 Setup

We evaluate DODGE-IT on SafeLIBERO, a safety-critical manipulation benchmark introduced by VLSA [Hu et al., 2025]. We follow the standard VLSA protocol and compare against OpenVLA-OFT, $\pi_{0.5}$, and VLSA. We report three metrics: collision avoidance rate (CAR, higher is better), task success rate (TSR, higher is better), and execution time steps (ETS, lower is better). CAR measures whether the robot avoids collision, TSR measures whether it completes the instruction, and ETS captures execution efficiency.

For the main real-robot evaluation, we use four manipulation tasks: Tomato Transfer, Cup Emptying, Tomato & Carrot Transfer, and Paper Cup Nesting. Each task is evaluated with three obstacle categories: a differently colored paper cup, a mannequin hand, and a cardboard box. For each method, we perform 20 evaluation trials for each of the 12 task-obstacle combinations with randomized object placement, totaling 240 trials per method. We report TSR and CAR to measure task completion and collision avoidance, respectively. Detailed task definitions, obstacle specifications, and real-robot configurations are provided in Appendix A.1.

4.2 Implementation Details

Simulation training. Starting from $\pi_{0.5}$ supervised-fine-tuned on the original LIBERO demonstrations, DODGE-IT performs PPO training in SafeLIBERO. Task success provides a reward of $+1$, and the first collision detected by TRACE in each trajectory incurs a penalty of -2. We train for 1,000 steps using Adam on four H800 GPUs. The policy and value-network learning rates start at 5.0×10^{-6} and 1.0×10^{-4}, respectively, and follow cosine schedules with a minimum learning-rate ratio of 0.1. Adam uses $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1.0 \times 10^{-8}$, and a weight decay of 0.01. The gradient clipping threshold is 1.0.

Real-robot training. We initialize the real-robot VLA policy by fine-tuning $\pi_{0.5}$ on 400 obstacle-free demonstrations, with 100 demonstrations per task. We then deploy this SFT policy to collect 60 rollouts for each of the 12 task-obstacle combinations, yielding 720 trajectories.

For RECAP-style training [Physical Intelligence et al., 2025], we use a value model with SigLIP-2 (so400m-patch14-224) and Gemma-3 (270M). A top-30% advantage threshold defines the positive condition. We apply condition dropout to 10% of the positively conditioned training samples for classifier-free guidance (CFG). The VLA policy is trained for 30,000 steps using Adam on four H800 GPUs, with a peak learning rate of 1.0×10^{-5}, a 500-step warmup, and cosine decay to 0.0. Adam uses $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 1.0 \times 10^{-8}$, and a weight decay of 1.0×10^{-10}.

4.3 Main SafeLIBERO Results

Table 1 reports results on the four SafeLIBERO suites. DODGE-IT achieves higher CAR and TSR and lower ETS than the reported baselines on every suite.

Across all four suites, DODGE-IT improves average CAR from 78.91% to 83.83% relative to VLSA and average TSR from 64.19% to 87.60%. ETS decreases from 267.63 to 184.60 steps. Section 4.4 separates the contributions of task-reward training and collision feedback. Appendix A.3 reports episode-level 95% confidence intervals for the DODGE-IT results in Table 1.

Table 1: SafeLIBERO results on reported suites under the VLSA protocol. Results are presented as CAR↑ / TSR↑ / ETS↓. DODGE-IT results are averaged over three evaluation seeds with 1,200 episodes per suite. **Bold** indicates the best method per column.

Methods	Spatial	Goal	Object	Long
	CAR / TSR / ETS	CAR / TSR / ETS	CAR / TSR / ETS	CAR / TSR / ETS
OpenVLA-OFT	12.75 / 35.75 / 238.1	25.00 / 22.50 / 255.6	11.25 / 28.50 / 245.3	5.50 / 3.50 / 541.5
$\pi_{0.5}$	15.25 / 59.75 / 201.7	23.75 / 54.25 / 210.3	14.50 / 48.75 / 215.1	12.75 / 35.75 / 478.0
VLSA	75.50 / 73.25 / 188.2	81.50 / 75.25 / 179.6	79.00 / 64.50 / 222.6	79.63 / 43.75 / 480.1
DODGE-IT	**83.60 / 85.80 / 152.5**	**87.10 / 88.40 / 132.0**	**82.80 / 88.80 / 166.8**	**81.80 / 87.40 / 287.1**

4.4 Ablation Study

Contribution and sensitivity of the collision penalty. We study the contribution of collision feedback and the effect of penalty strength by varying $\lambda \in \{0, 1, 2\}$ under the same PPO training pipeline. Table 2 summarizes the results alongside the SFT initialization, averaged across the four SafeLIBERO suites. The $\lambda = 0$ setting disables the collision penalty, while $\lambda = 2$ is used in our main experiments.

Table 2: Contribution and sensitivity of the collision penalty under the same PPO training pipeline. Results are averaged across the four SafeLIBERO suites; the $\lambda = 2$ row uses the three-seed means from Table 1. **Bold** indicates the best result per metric.

Setting	CAR (%)↑	TSR (%)↑	ETS↓
$\pi_{0.5}$ SFT	16.56	49.63	276.28
Task-only PPO ($\lambda = 0$)	49.80	86.10	**181.00**
DODGE-IT ($\lambda = 1$)	81.25	**88.56**	183.20
DODGE-IT ($\lambda = 2$)	**83.83**	87.60	184.60

Task-only PPO reaches 86.10% TSR, accounting for nearly all of the TSR improvement over SFT. With $\lambda = 2$, CAR rises from 49.80% to 83.83% (+34.03 percentage points), while TSR increases from 86.10% to 87.60% and ETS increases from 181.00 to 184.60 steps. Thus, collision feedback substantially improves obstacle avoidance while preserving task performance, with a small execution-efficiency cost relative to task-only PPO. Increasing λ from 1 to 2 raises CAR by 2.58 percentage points, while TSR decreases by 0.96 points and ETS increases by 1.40 steps, reflecting the trade-off between obstacle avoidance and task performance.

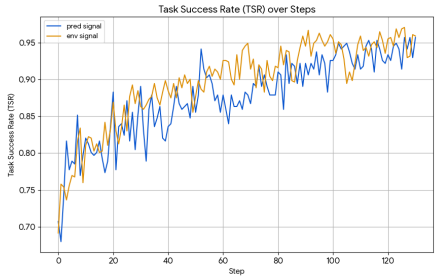
Collision-feedback fidelity. We evaluate TRACE-derived collision outcomes against simulator collision ground truth on 1,664 trajectories from the four SafeLIBERO suites: 460 collision-positive and 1,204 collision-negative trajectories. At the thresholds used in the main SafeLIBERO experiments, $\tau_{\text{disp}} = 0.005$ and $\tau_{\text{prox}} = 0.005$, TRACE achieves precision of 0.910 and recall of 0.874.

The proximity cue recovers 40 collision-positive trajectories missed by displacement alone. Table 3 shows that adding proximity increases recall from 0.787 to 0.874, while precision changes from 0.919 to 0.910.

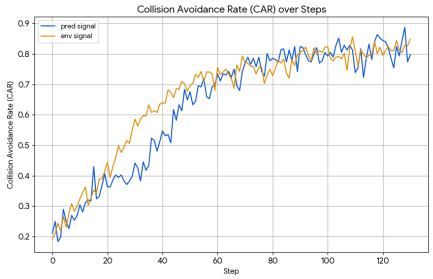
Table 3: Collision-assessment cues evaluated against simulator collision ground truth on the same 1,664 trajectories.

Collision-assessment cues	Precision	Recall
Displacement only	0.919	0.787
Displacement + proximity	0.910	0.874

Simulation also allows a matched comparison of policies trained with TRACE-derived or simulator-provided collision outcomes. Both settings retain the same task success feedback; only the source of collision outcomes is changed. Figure 4 shows close TSR and CAR, supporting the use of TRACE-derived outcomes alongside task success signals in the joint learning objective. Appendix A.2 reports real-robot collision assessment, threshold sensitivity, and component-level evaluations of TRACE.



TSR: simulator vs. TRACE feedback



CAR: simulator vs. TRACE feedback

Figure 4: **Ablation study on collision-feedback fidelity.** We compare the performance of DODGE-IT trained with simulator-provided or TRACE-derived collision outcomes. Both settings use the same task success feedback and achieve close TSR and CAR.

4.5 Combining Policy Learning with External Safeguards

We apply the VLSA external safeguard at inference time to the trained DODGE-IT policy and evaluate this combination on the same four SafeLIBERO suites, with 400 episodes per suite across the two difficulty levels. The safeguard uses simulator-provided obstacle identities. This tests whether an external safeguard provides additional collision avoidance after collision-aware policy learning.

Table 4: Combining DODGE-IT with the VLSA external safeguard in SafeLIBERO. DODGE-IT uses the three-seed means from Table 1; DODGE-IT + VLSA uses 400 episodes per suite. Entries are TSR (%) / CAR (%). Bold marks the best result for each metric.

Method	Spatial	Goal	Object	Long	Mean
DODGE-IT	**85.8** / 83.6	**88.4** / 87.1	**88.8** / 82.8	**87.4** / 81.8	**87.6** / 83.8
DODGE-IT + VLSA	74.2 / **93.3**	78.2 / **93.3**	81.3 / **88.8**	59.0 / **89.8**	73.2 / **91.3**

Table 4 shows that adding the safeguard raises average CAR by 7.5 percentage points, from 83.8% to 91.3%, while reducing average TSR by 14.4 points, from 87.6% to 73.2%. This trade-off occurs on every suite; Long has the largest TSR decrease, from 87.4% to 59.0%. External safeguarding can therefore further increase collision avoidance after policy learning, while its action corrections can still interfere with task completion. The task-only PPO control in Section 4.4 separately shows that collision feedback improves CAR within policy learning while preserving TSR.

4.6 Real-Robot Evaluation

We compare DODGE-IT with three real-robot baselines. **SFT** is the $\pi_{0.5}$ policy fine-tuned on the 400 obstacle-free demonstrations. **SFT + VLSA** applies the VLSA external safeguard to this SFT policy at inference time. **Collision-Free BC** starts from the same SFT policy as DODGE-IT and uses the same collected rollouts, performing behavior cloning only on successful trajectories for which TRACE detects no collision. The BC baseline tests whether cloning successful, collision-free executions is sufficient to match DODGE-IT's advantage-conditioned policy improvement.

Table 5: Real-robot results across four tasks. Entries are TSR (%) / CAR (%), aggregated over three obstacle categories with 20 trials per category. Mean averages the four tasks. **Bold** indicates the best result per metric and column.

Method	Tomato Transfer	Cup Emptying	Tomato & Carrot Transfer	Paper Cup Nesting	Mean
SFT	71.7 / 20.0	55.0 / 15.0	40.0 / 8.3	23.3 / 10.0	47.5 / 13.3
SFT + VLSA	65.0 / 66.7	55.0 / 63.3	35.0 / 56.7	18.3 / 46.7	43.3 / 58.3
Collision-Free BC	78.3 / 73.3	71.7 / 68.3	48.3 / 58.3	30.0 / 46.7	57.1 / 61.7
DODGE-IT	**86.7 / 81.7**	**76.7 / 75.0**	**55.0 / 71.7**	**40.0 / 56.7**	**64.6 / 71.3**

Table 5 shows that DODGE-IT achieves the highest TSR and CAR on all four tasks, with averages of 64.6% and 71.3%, respectively. Compared with Collision-Free BC, DODGE-IT improves average TSR by 7.5 percentage points and CAR by 9.6 percentage points. With the same SFT initialization and collected rollouts, these gains support the benefit of advantage-conditioned policy improvement over cloning only successful, collision-free trajectories.

Applying VLSA to the SFT policy increases average CAR from 13.3% to 58.3%, while average TSR decreases from 47.5% to 43.3%. DODGE-IT achieves both higher task success and collision avoidance than SFT + VLSA, exceeding it by 21.3 TSR points and 13.0 CAR points on average.

We further evaluate generalization to two unseen obstacles: a real human hand and a paper cup with a different color and texture. Neither appears in the training rollouts. We evaluate SFT, SFT + VLSA, and DODGE-IT without retraining across all four tasks, with 20 trials per task-obstacle combination, yielding 160 trials per method.

Table 6: Real-robot generalization to unseen obstacles: a real human hand and a paper cup with a different color and texture. Entries are TSR (%) / CAR (%). Each obstacle column aggregates 80 trials across four tasks; Mean averages the two obstacles. **Bold** indicates the best result per metric and column.

Method	Human hand	Paper cup	Mean
SFT	47.5 / 13.8	48.8 / 13.8	48.1 / 13.8
SFT + VLSA	38.8 / 41.3	43.8 / 51.3	41.3 / 46.3
DODGE-IT	**63.8 / 62.5**	**63.8 / 65.0**	**63.8 / 63.8**

Table 6 shows that DODGE-IT achieves the highest TSR and CAR on both unseen obstacles. Its average TSR and CAR are both 63.8%, compared with 48.1% / 13.8% for SFT and 41.3% / 46.3% for SFT + VLSA. Compared with SFT + VLSA, DODGE-IT improves average TSR by 22.5 percentage points and CAR by 17.5 percentage points.

5 Discussion and Limitations

Why different optimizers for simulation and real-robot learning? The optimizer follows the data-access setting. Resettable simulation supports repeated online interaction, allowing PPO to update the policy from fresh rollouts. Real-robot training reuses collected executions through RECAP-style advantage estimation and advantage-conditioned policy improvement. Both instantiations use task success outcomes and TRACE-derived collision outcomes as feedback.

Limitations. Our evaluation focuses on collision avoidance in manipulation and relies on RGB-D obstacle perception. Failures in detection, depth, or point-cloud construction can affect the collision outcomes used for policy improvement, and the displacement and proximity thresholds introduce a precision–recall trade-off (Appendix A.2). Scene adaptation is also required; the measured setup cost for a new suite with 400 scenes is approximately 13.4 minutes on one GPU. Our real-robot evaluation covers four tasks and includes unseen obstacles, while continuously moving obstacles remain to be studied. DODGE-IT improves empirical collision avoidance while preserving task completion, but does not provide formal safety guarantees.

6 Conclusion

We presented DODGE-IT, a collision-aware learning framework for improving VLA manipulation policies from task and collision feedback. DODGE-IT uses TRACE to convert RGB-D-derived obstacle observations and execution outcomes into collision signals, then combines these signals with task success in representative reinforcement-learning frameworks. Across SafeLIBERO and real-robot experiments, DODGE-IT improves obstacle avoidance while preserving task completion. These findings support collision-aware policy improvement as a practical complement to external collision-avoidance safeguards for robotic manipulation.

Acknowledgments and Disclosure of Funding

We sincerely thank the ENVISION Lab for providing computational resources and robotic hardware for this research. We sincerely thank Shangjin Xie for valuable discussions.

References

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, et al. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of Machine Learning Research*, volume 229, 2023. URL https://robotics-transformer2.github.io.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A reinforced fine-tuning method for VLA models via consistency policy. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.019. URL https://cccedric.github.io/conrft/.

Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Dueñez-Guzman, and Mohammad Ghavamzadeh. Safe policy learning for continuous control. In *Conference on Robot Learning*, 2020. URL https://corlconf.github.io/corl2020/paper_171/.

Adam Fishman, Adithyavairavan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter Fox. Motion policy networks. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 967–977. PMLR, 2023. URL https://proceedings.mlr.press/v205/fishman23a.html.

Adam Fishman, Aaron Walsman, Mohak Bhardwaj, Wentao Yuan, Balakumar Sundaralingam, Byron Boots, and Dieter Fox. Avoid everything: Model-free collision avoidance with expert-guided fine-tuning. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 1925–1948. PMLR, 2025. URL https://proceedings.mlr.press/v270/fishman25a.html.

Songqiao Hu, Zeyi Liu, Shuang Liu, Jun Cen, Zihan Meng, and Xiao He. VLSA: Vision-language-action models with plug-and-play safety constraint layer, 2025. URL https://arxiv.org/abs/2512.11891.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

I-Chun Arthur Liu, Shagun Uppal, Gaurav S. Sukhatme, Joseph J. Lim, Peter Englert, and Youngwoon Lee. Distilling motion planner augmented policies into visual control policies for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 641–650. PMLR, 2022. URL https://proceedings.mlr.press/v164/liu22b.html.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. VLA-RL: Towards masterful and general robotic manipulation with scalable reinforcement learning, 2025. URL https://arxiv.org/abs/2505.18719.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, et al. $\pi^*_{0.6}$: a VLA that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Yunzhou Song, Long Le, Yong-Hyun Park, Jie Wang, Junyao Shi, Lingjie Liu, Jiatao Gu, Eric Eaton, Dinesh Jayaraman, and Kostas Daniilidis. OmniGuide: Universal guidance fields for enhancing generalist robot policies, 2026. URL https://arxiv.org/abs/2603.10052.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models, 2025. URL https://arxiv.org/abs/2505.17016.

Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery RL: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3): 4915–4922, 2021. doi: 10.1109/LRA.2021.3070252.

Yubai Wei, Chen Wu, and Hashem Haghbayan. Can explicit physical feasibility benefit VLA learning? an empirical study, 2026. URL https://arxiv.org/abs/2604.17896.

Jun Yamada, Youngwoon Lee, Gautam Salhotra, Karl Pertsch, Max Pflueger, Gaurav Sukhatme, Joseph Lim, and Peter Englert. Motion planner augmented reinforcement learning for robot manipulation in obstructed environments. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 589–603. PMLR, 2021. URL https://proceedings.mlr.press/v155/yamada21a.html.

Jiahui Yang, Jason Jingzhou Liu, Yulong Li, Youssef Khaky, Kenneth Shaw, and Deepak Pathak. Deep reactive policy: Learning reactive manipulator motion planning for dynamic environments. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 5056–5071. PMLR, 2025. URL https://proceedings.mlr.press/v305/yang25d.html.

Dian Yu, Qingchuan Zhou, Bingkun Huang, Majid Khadiv, and Zewen Yang. Safe-night VLA: Seeing the unseen via thermal-perceptive vision-language-action models for safety-critical manipulation, 2026. URL https://arxiv.org/abs/2603.05754.

Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Josef Dai, Yuanpei Chen, and Yaodong Yang. SafeVLA: Towards safety alignment of vision-language-action model via constrained learning. In *Advances in Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=dt940loCBT.

Hongyin Zhang, Shuo Zhang, Junxi Jin, Qixin Zeng, Runze Li, and Donglin Wang. RobustVLA: Robustness-aware reinforcement post-training for vision-language-action models, 2025b. URL https://arxiv.org/abs/2511.01331.

A Appendix

A.1 Additional Real-Robot Experimental Details

In this section, we provide extended details regarding our real-robot evaluation. The real-robot setup utilizes an **AgileX Robotics Piper**, a lightweight 6-degree-of-freedom (6-DOF) robotic arm, equipped with an Intel RealSense depth camera. We additionally use an agent camera for VLA and 2 cameras for multiview detection. We evaluate DODGE-IT across the following four tasks and three obstacle categories. Representative obstacles are shown in Figure 5.



Figure 5: **Obstacles used in the real-robot experiments.**

Task Definitions:

- **Task 1 (Tomato Transfer):** "Pick the tomato then place it onto the tray." This task requires the robot to navigate a narrow passage between the starting position and the target tray, testing the policy's ability to maintain clearance while moving towards a goal.
- **Task 2 (Cup Emptying):** "Pick up the mug containing the eggplant and pour the eggplant onto the tray." This task involves more complex rotational movements and a higher-reaching trajectory, which increases the likelihood of potential collisions throughout the interaction and transport phases.
- **Task 3 (Tomato & Carrot Transfer):** "Pick the tomato and place it onto the tray, then pick the carrot and place it onto the tray." This longer-horizon task requires two grasp-and-place operations in sequence.
- **Task 4 (Paper Cup Nesting):** "Pick up the green cup and place it inside the red cup." This task requires precise alignment of the grasped cup with the receiving cup during placement.

Obstacle Configurations:

- **Obstacle 1 (Differently Colored Paper Cup):** A differently colored, light, deformable paper cup representing common household items. Its low mass makes it sensitive to even slight physical displacements, serving as a strict test for our visual-displacement-based collision detector.
- **Obstacle 2 (Mannequin Hand):** An articulated obstacle with high-texture and complex geometry. This obstacle tests the robustness of the RGB-D perception pipeline and the policy's ability to learn collision-aware behavior around more complex geometry.
- **Obstacle 3 (Cardboard Box):** A bulkier cardboard box measuring $20 \times 14 \times 4$ cm.

A.2 Additional Evaluation of TRACE

We evaluate scene adaptation and runtime TRACE separately. Scene adaptation covers VLM point annotation, SAM mask generation, and detector fine-tuning. Runtime evaluation covers task-relevant obstacle selection and collision assessment.

A.2.1 Scene Adaptation

We evaluate VLM point annotation on 400 scenes per SafeLIBERO suite, using simulator ground truth for scoring. Qwen3-VL-Plus achieves precision of 0.906 and recall of 0.864. For correctly placed point prompts, SAM2.1-b achieves an average mask IoU of 0.882 across the four suites. We fine-tune YOLO11x-seg for 20 epochs on 320 scenes per suite and evaluate it on 80 held-out scenes with ground-truth labels. Its segmentation mAP50, averaged across the four suites, is 0.905.

Point-annotation errors include 4.3% hallucinated points and 5.1% misplaced points. The lowest class-level recall occurs for visually ambiguous or occluded objects such as `wooden_box`. SAM errors are dominated by the composite `flat_stove` fixture, for which the best of eight tested point-prompt strategies achieves an IoU of 0.495.

A.2.2 Task-Relevant Obstacle Selection

We evaluate task-relevant obstacle selection on 400 scenes per suite, for a total of 1,600 scenes. Using Qwen2.5-7B-Instruct, this step achieves precision of 0.921 and recall of 0.907. The predicted obstacle set exactly matches the ground-truth set in 75.4% of scenes. Before obstacle selection, 3.4% of ground-truth task-relevant objects are already absent from the YOLO detector candidates.

A.2.3 Real-Robot Collision Assessment

We evaluate collision assessment on real-robot rollouts across the four manipulation tasks described in Appendix A.1, using the paper-cup obstacle from the main evaluation. We manually annotate collision onset in these rollouts and use these annotations as ground truth.

Against these human annotations, TRACE achieves precision of 0.760 and recall of 0.950.

A.2.4 Threshold Sensitivity

Using the 1,664 simulation trajectories evaluated in Section 4.4, we vary one threshold at a time while holding the other at 0.005. Table 7 reports both sweeps. Decreasing the displacement threshold increases recall while reducing precision; increasing the proximity threshold produces the same trade-off. Both sweeps exhibit gradual precision–recall trade-offs over the tested ranges.

Table 7: Threshold sensitivity of TRACE. Top: displacement-threshold sweep with $\tau_{\text{prox}} = 0.005$. Bottom: proximity-threshold sweep with $\tau_{\text{disp}} = 0.005$. Bold marks the main-experiment setting and its results.

τ_{disp}	0.020	0.010	**0.005**	0.002	0.001
Precision	0.911	0.911	**0.910**	0.870	0.862
Recall	0.598	0.735	**0.874**	0.946	0.976

τ_{prox}	0	0.001	0.002	**0.005**	0.010
Precision	0.919	0.918	0.918	**0.910**	0.887
Recall	0.787	0.833	0.854	**0.874**	0.891

A.2.5 Scene-Adaptation Cost and Runtime

Scene adaptation for a new suite with 400 scenes takes approximately 13.4 minutes on one GPU: 4.0 minutes for VLM point annotation, 3.4 minutes for SAM mask generation and export, and 6.0 minutes for 20-epoch detector training. No manual boxes, point annotations, or mask validation are required for this adaptation procedure.

At runtime, task-relevant obstacle selection using Qwen2.5-7B-Instruct takes 0.15 seconds per scene. For the remaining computation, TRACE takes a median of 62 ms per environment step in a single environment. The reported component timings include 32.4 ms for YOLO segmentation, 11.7 ms for point-cloud fusion, and 4.7 ms for collision assessment. With 32 parallel training environments, GPU batching reduces the average TRACE time to approximately 26 ms per environment-step. Table 8 summarizes these timings.

Table 8: Runtime of TRACE. Obstacle selection is measured per scene; the remaining timings are measured per environment step.

Operation	Time
Task-relevant obstacle selection	0.15 s/scene
YOLO segmentation	32.4 ms/step
Point-cloud fusion	11.7 ms/step
Collision assessment	4.7 ms/step
Single-environment total (median)	62 ms/step
32-environment batching (mean)	26 ms/environment-step

A.3 Additional SafeLIBERO Evaluation

We repeat evaluation of the fixed DODGE-IT policy under three evaluation seeds. Each suite contains 1,200 episodes: three seeds $\times$ two difficulty levels $\times$ 200 episodes. Table 9 reports episode-level 95% confidence intervals: Wilson intervals for the binary TSR and CAR outcomes, and bootstrap intervals for mean ETS using 10,000 resamples. These intervals quantify evaluation variability for the fixed policy weights.

Table 9: DODGE-IT under three evaluation seeds, with 1,200 episodes per suite. TSR and CAR are percentages; brackets give episode-level 95% confidence intervals.

Suite	TSR [95% CI]	CAR [95% CI]	ETS [95% CI]
Spatial	85.8 [83.7, 87.6]	83.6 [81.4, 85.6]	152.5 [149.0, 156.3]
Goal	88.4 [86.5, 90.1]	87.1 [85.1, 88.9]	132.0 [127.9, 136.0]
Object	88.8 [86.9, 90.5]	82.8 [80.5, 84.8]	166.8 [163.7, 169.9]
Long	87.4 [85.4, 89.2]	81.8 [79.6, 83.9]	287.1 [281.1, 293.4]

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the scope of the paper, the TRACE module, the reinforcement-learning instantiations, and the simulation and real-robot evaluation settings. The claims are limited to improving collision-aware manipulation and preserving task completion, which are evaluated in the Experiments section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The Limitations paragraph in Section 5 discusses dependence on RGB-D perception, threshold trade-offs, scene-adaptation cost, continuously moving obstacles, and the absence of formal safety guarantees. Section 4.4 and Appendix A.2 quantify TRACE accuracy, threshold sensitivity, and adaptation cost.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: The paper does not present theoretical results. The equations define the feedback and learning objective used in the method rather than formal theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The Method and Experiments sections describe the collision-feedback construction, metrics, benchmark protocol, baselines, real-robot evaluation setup, reward design, optimizers, data collection, and training hyperparameters. The appendix provides additional real-robot, TRACE, and repeated-evaluation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No].

   Justification: The paper does not provide open access to code or data. Reproducibility information is provided through the method description, experimental protocol, implementation details, and appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The Experiments section specifies the benchmark, baselines, metrics, evaluation protocol, and simulation and real-robot training details, including reward design, optimizers, trajectory counts, and hyperparameters. The appendix provides additional real-robot, task, obstacle, TRACE, and repeated-evaluation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

Justification: Appendix A.3 reports episode-level 95% confidence intervals for DODGE-IT evaluated under three seeds with fixed policy weights: Wilson intervals for TSR/CAR and bootstrap intervals for mean ETS using 10,000 resamples. The evaluation protocol and number of real-robot trials are reported in the Experiments section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No].

   Justification: Section 4.2 reports the GPU type and number used for the main experiments, and Appendix A.2.5 reports TRACE adaptation cost and runtime. Memory, runtime for each policy-training experiment, and total compute cost are not fully specified.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The research studies robot manipulation and collision avoidance using simulation and controlled real-robot experiments, and we are not aware of any aspect that violates the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No].

    Justification: The current draft focuses on the technical method and empirical evaluation and does not include a dedicated discussion of both positive and negative societal impacts. The main potential impact is more collision-aware robot manipulation, while incorrect deployment without validation could still cause physical damage.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release high-risk pretrained models, image generators, scraped datasets, or other assets that require misuse-specific release safeguards in the current submission.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No].

    Justification: The paper cites the existing models, benchmarks, and methods used in the study, including VLA models and SafeLIBERO. The current draft does not yet explicitly list the licenses or terms of use for all existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The current submission does not release new datasets, code packages, or model checkpoints as standalone assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or human-subject experiments. The real-robot experiments evaluate robot policy behavior in controlled manipulation tasks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or human-subject experiments, so IRB approval or equivalent human-subject review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: The method uses an LLM as part of TRACE to reason over language goals and candidate object masks for task-relevant obstacle identification, as described in the Method section and Figure 3.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.