PAC: RISK-AWARE PLANNER-AUGMENTED CONTROL FOR SAFE MANIPULATION IN DYNAMIC ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic obstacles create a fundamental challenge for robotic manipulation: the controller must react rapidly to imminent collisions without losing progress toward the task goal. Motion planners provide structured and goal-consistent references, but their solutions can quickly become outdated when the environment changes during execution. In contrast, learned reactive policies can respond to local disturbances, yet may produce unstable or excessively conservative actions without global guidance. We introduce Planner-Augmented Control (PAC), a hybrid framework that continuously combines planner-generated references with learned reactive corrections. A differentiable, risk-aware gate adjusts their relative contributions online, favoring nominal trajectory tracking in low-risk states and stronger corrective behavior as collision risk increases. PAC further incorporates a lightweight model-predictive safety shield that evaluates short-horizon motion and modifies potentially unsafe commands before execution. The framework is trained through a two-stage procedure consisting of behavior-cloning initialization and danger-aware optimization with explicit gate supervision. We evaluate PAC across manipulation scenarios involving static obstacles, moving obstacles, sudden obstacle appearance, and goal-blocking interactions. Experimental results show that PAC preserves reliable goal-directed behavior in static scenes while improving collision avoidance and task completion under dynamic disturbances compared with classical planning and purely learning-based baselines. These results demonstrate that risk-aware planner–policy fusion provides an effective approach to robust manipulation in changing environments.

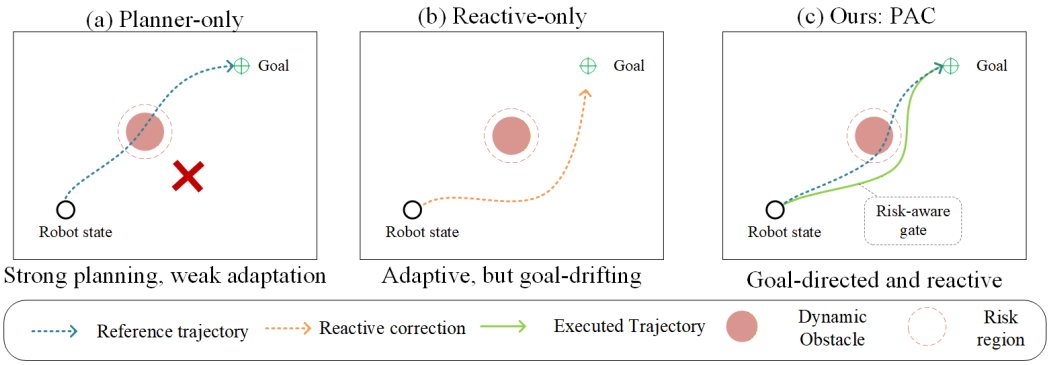


Figure 1: Motivation for PAC. Planner-only execution is goal-directed but brittle to dynamic obstacles, whereas reactive-only execution adapts locally but may drift from the goal. PAC combines both capabilities through continuous planner–policy integration.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Robotic manipulation in dynamic environments needs both globally consistent goal-directed progress and rapid adaptation to unexpected obstacle motion. Figure 1 illustrates that optimization-based planners provide goal-directed reference trajectories but may fail when moving obstacles violates static-scene assumptions. Reactive policies adapt quickly to local changes, but without global guidance, may take inefficient evasive actions or drift away from the intended goal. Achieving both capabilities within a unified execution framework therefore remains a fundamental challenge.

Optimization-based methods such as CHOMP (Ratliff et al., 2009), TrajOpt (Schulman et al., 2014), sampling-based optimal planning (Karaman & Frazzoli, 2011), and cuRobo (Sundaralingam et al., 2023) efficiently generate feasible, collision-free trajectories for known environments. However, recovering from unexpected obstacle motion through replanning can introduce computational latency and discontinuous motion updates (Fox et al., 2002). In contrast, learning-based controllers operate in closed loop and respond rapidly to local observations (Ross et al., 2011; Schulman et al., 2017; Chi et al., 2025), but many do not explicitly preserve long-horizon, goal-directed trajectory structure (Kumar et al., 2019). This exposes a fundamental trade-off in dynamic manipulation. Planner guidance ensures goal-directed motion but may become unsafe under environmental changes, while reactive control adapts locally at the risk of deviating from the goal.

We propose **Planner-Augmented Control (PAC)**, a hybrid execution framework that integrates planner guidance with learned reactive correction, as shown in Figure 2. At each control step, a global planner provides a reference joint increment, while a lightweight policy predicts a local corrective joint update and a continuous gate from the current robot state, planner reference, and dynamic-obstacle observations. The gate favors planner-guided motion in nominal states and gradually increases reactive correction as the obstacle configuration becomes more hazardous. Unlike hard controller switching, this continuous arbitration preserves task progress while reducing abrupt motion transitions. Continuous gating improves adaptability but cannot guarantee that every learned action is safe under sudden or out-of-distribution disturbances. PAC therefore includes a **risk-triggered MPC (Model Predictive Control) safety shield**. The shield evaluates short-horizon candidate actions only when residual collision risk is detected, preserving low-latency nominal execution while providing additional correction in safety-critical states.

Learning proper corrective actions and gate values needs anticipating danger rather than reacting only when collision is imminent. We therefore introduce **prospective danger-aware supervision**, which derives gate targets from near-future collision risk over a short prediction horizon. We train PAC using a **two-stage training curriculum**. Stage 1, **planner-imitation pretraining**, uses cuRobo reference actions to establish a stable, feasible, and goal-directed motion prior. Stage 2, **dynamic-aware reactive fine-tuning**, warm-starts from the Stage-1 checkpoint and jointly learns local corrective actions and continuous gating in dynamic-obstacle scenes. Thus, Stage 1 learns how to move toward the goal, while Stage 2 learns when and how to preserve or deviate from that motion. The MPC safety shield is applied only during inference and is not an additional training stage.

We evaluate PAC on six static and dynamic manipulation families. Our results show improved robustness to dynamic obstacle motion while maintaining reliable performance in static environments. Ablations further validate the benefits of planner-reference conditioning, continuous gating, two-stage training, and the inference-time safety shield.

Our contributions are fourfold:

(1) We introduce **Planner-Augmented Control (PAC)**, which continuously integrates goal-directed planner references with learned local corrective actions through a differentiable gate.

(2) We develop a **two-stage training curriculum with prospective danger-aware supervision**, combining planner-imitation pretraining with dynamic-aware learning of corrective actions and continuous gating.

(3) We introduce a **risk-triggered MPC safety shield** that selectively provides predictive action correction during safety-critical execution.

(4) We conduct a structured evaluation across six static and dynamic manipulation families, including component ablations and matched comparisons between continuous gating and hard switching.

2 Related Work

Optimization-Based Motion Planning is a core paradigm in robotic manipulation. Methods such as CHOMP, STOMP, and TrajOpt generate smooth, collision-free trajectories via gradient optimization (Ratliff et al., 2009; Kalakrishnan et al., 2011; Schulman et al., 2014). Recent advances improve scalability and real-time performance, including cuRobo's GPU-parallel optimization (Sundaralingam et al., 2023) and differentiable collision checking with Gaussian processes (Schulman et al., 2014; Dong et al., 2016). Sampling-based planners (RRT*, BIT*) improve asymptotic optimality via informed graph search (Karaman & Frazzoli, 2011; Gammell et al., 2015). However, these trajectory-level methods assume static environments; dynamic obstacles need replanning, causing latency and disrupting execution. Thus, they provide strong global structure but limited adaptability to dynamic disturbances.

Transformer and Diffusion-Based Motion Policies. Recent neural motion generation advances improve closed-loop control. Diffusion Policy (Chi et al., 2025) models actions via iterative denoising, reaching strong stability. Transformer-based methods (ACT (Zhao et al., 2023) and extensions (Zhang, 2025; Li et al., 2025)) enhance temporal modeling and multimodal visuomotor fusion. Large-scale pretrained policies show scalability. DRP (Yang et al., 2025) gives reactive goal proposals for dynamic settings, while vision-language-action models (RT-2 (Zitkovich et al., 2023), PaLM-E (Driess et al., 2023), Octo (Octo Model Team et al., 2024)) show foundation model potential in robotics. However, they are fully neural, lacking explicit global trajectory structure. Thus, they enable fast local adaptation but may yield globally inconsistent behavior under dynamic perturbations or distribution shifts.

Planner-Learning Integration and Residual Policy Learning. Hybrid methods bridge planning and reactivity by combining model-based planning with learned policies. Residual learning adds corrections to nominal trajectories. Model Predictive Path Integral (MPPI) control and related MPC methods enable predictive avoidance but need repeated online optimization and accurate environment modeling (Williams et al., 2017b). Others combine planning and learning via interpolated policy gradients and motion policy networks (Gu et al., 2017; Fishman et al., 2023). RMPflow (Cheng et al., 2018) composes task-space objectives smoothly, but mixture-of-experts enables soft controller arbitration (Jacobs et al., 1991). Safety layers (shielding) enforce constraints during execution (Alshiekh et al., 2018). However, many hybrids rely on discrete controller switching or repeated optimization, causing instability or discontinuities. Continuous integration of a global reference with learned corrections is underexplored. In this work, our PAC models planner augmentation as a differentiable joint-space gating mechanism.

Dynamic Obstacle Avoidance and Simulation Platforms. Dynamic obstacle avoidance uses model predictive control (MPC), trajectory deformation, and reinforcement learning (RL) (Williams et al., 2017a; Gu et al., 2017). MPC offers prediction but is computationally costly due to repeated optimization. RL adapts to unmodeled dynamics but lacks guarantees of consistent goal-directed behavior. Simulation platforms accelerate research: Isaac Gym (Makoviychuk et al., 2021) enables large-scale GPU simulation, and ManiSkill2 (Gu et al., 2023) provides standardized manipulation benchmarks. Unlike replanning-only or fully neural methods, PAC conditions reactive correction on a planner reference increment. In stable settings, execution follows the global plan; under dynamic risk, a learned gate continuously modulates deviation, enabling both global consistency and local adaptability.

3 Our Proposed Method

3.1 Overview

We propose PAC (**P**lanner-**A**ugmented reactive **C**ontrol; **PAC**), a framework for dynamic obstacle avoidance in robotic manipulation. Existing methods often struggle to achieve both global task consistency and fast local adaptation. Optimization-based planners provide goal-directed motion, but they are vulnerable to unexpected obstacle motion during execution. Reactive policies can respond quickly to local changes, yet they lack the long-horizon guidance needed to reliably reach the goal. Our key idea is to combine planner guidance with learned reactive control in a unified execution framework. Figure 2 illustrates that PAC consists of three main components. A global planner first generates a reference trajectory that provides task-level motion structure. A learned reactive policy

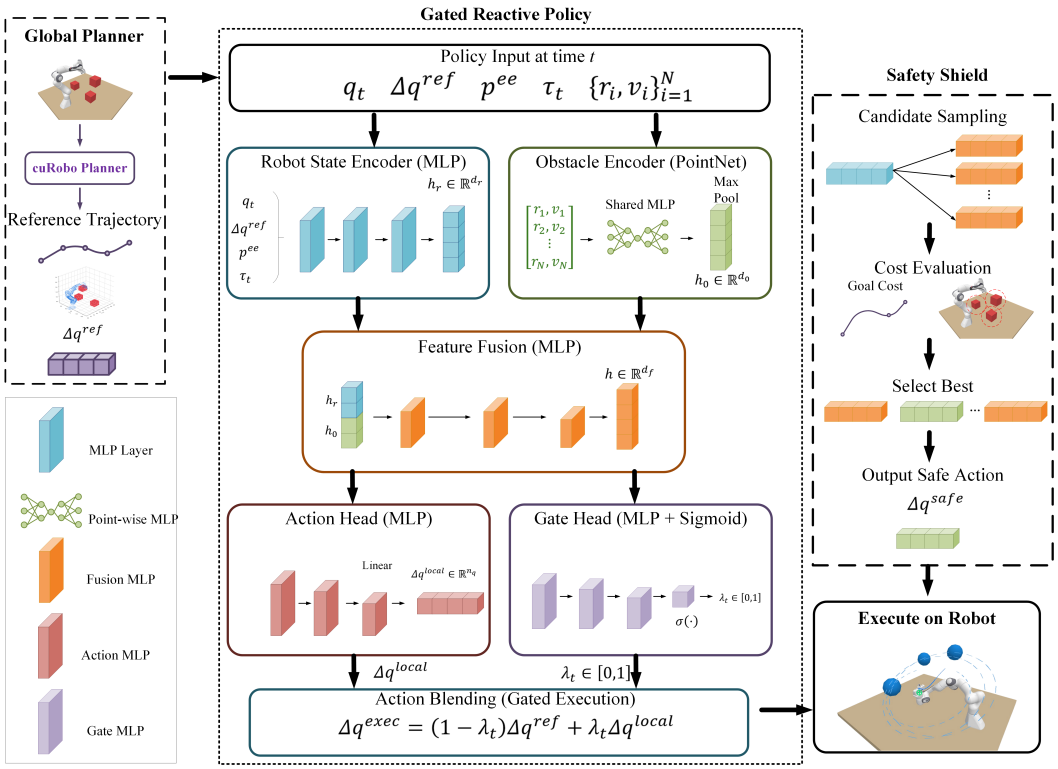


Figure 2: Overview of PAC. A gated reactive policy combines a planner reference with a learned local correction. When residual collision risk is detected, the MPC safety shield selects a safer action for execution.

then predicts local corrective actions in response to dynamic obstacles. A continuous gating mechanism adaptively blends the planner reference and the reactive correction, enabling smooth transitions between goal-directed motion and local avoidance. In difficult cases, an MPC safety shield evaluates short-horizon candidate actions and corrects potentially unsafe motions during execution.

3.2 Global Planner Guidance

The global planner in PAC is used to provide task-level guidance rather than to solve every dynamic interaction online. Given an initial robot configuration q_0 and a target end-effector pose p^{goal}, we use cuRobo (Sundaralingam et al., 2023) to generate a reference trajectory under the static environment assumption:

$$\mathcal{Q}^{ref} = \{q_t^{ref}\}_{t=0}^{T}, \tag{1}$$

where $\mathcal{Q}^{ref}$ is the planned reference trajectory, $q_t^{ref} \in \mathbb{R}^{n_q}$ is the reference joint configuration at timestep t, n_q is the number of robot joints, and T is the trajectory horizon. During execution, the robot will temporarily deviate from the reference trajectory to avoid dynamic obstacles. Therefore, the planner-guided update is computed with respect to the current executed state:

$$\Delta q_t^{ref} = q_{t+1}^{ref} - q_t, \tag{2}$$

where $\Delta q_t^{ref} \in \mathbb{R}^{n_q}$ is the planner-guided joint update, $q_t \in \mathbb{R}^{n_q}$ is the current robot configuration, and q_{t+1}^{ref} is the next configuration on the reference trajectory. This design keeps the robot attracted to the global task path while allowing local deviations for obstacle avoidance.

3.3 State Representation

At each timestep, the policy observes the robot state, planner guidance, task goal, and dynamic obstacle information, forming the input vector

$$s_t = [q_t,\, \Delta q_t^{ref},\, p_t^{ee},\, p^{goal},\, \tau_t,\, f_t^{dyn}], \tag{3}$$

which concatenates the current joint configuration q_t, the planner-guided reference update Δq_t^{ref} from Eq. (2), the end-effector position $p_t^{ee} \in \mathbb{R}^3$, the target position $p^{goal} \in \mathbb{R}^3$, the normalized trajectory progress $\tau_t = t/T$, and the dynamic obstacle feature f_t^{dyn} described below. For each dynamic obstacle i, the policy encodes its motion relative to the end-effector as

$$r_{t,i} = p_{t,i}^{obs} - p_t^{ee} \quad \text{and} \quad v_{t,i} = \dot{p}_{t,i}^{obs} - \dot{p}_t^{ee}, \tag{4}$$

where $p_{t,i}^{obs}$ and $\dot{p}_{t,i}^{obs}$ are the obstacle position and velocity, and $\dot{p}_t^{ee}$ is the end-effector velocity. The dynamic feature set is

$$f_t^{dyn} = \{r_{t,i}, \text{and}\, v_{t,i}\}_{i=1}^{N}, \tag{5}$$

where N is the number of dynamic obstacles. We aggregate f_t^{dyn} with a permutation-invariant encoder inspired by PointNet (Qi et al., 2017), allowing the policy to handle a variable number of obstacles.

3.4 Reactive Policy and Continuous Gating

The reactive policy complements the motion planner by predicting a local corrective action rather than replacing the planned trajectory entirely. A shared encoder ϕ_θ, parameterized by learnable weights θ, maps the full state s_t to a compact latent representation

$$h_t = \phi_\theta(s_t). \tag{6}$$

Dynamic obstacle features within s_t are aggregated through a permutation-invariant pooling layer, enabling the encoder to process a variable number of obstacles without assuming any fixed input ordering. Two lightweight prediction heads then operate independently on h_t. The first head produces the local corrective joint update

$$\Delta q_t^{local} = \alpha \tanh(W_q h_t + b_q), \tag{7}$$

where W_q and b_q are learnable weight matrix and bias that project the latent feature into joint space, the $\tanh$ nonlinearity bounds each output dimension to $[-1, 1]$, and the scalar α sets the maximum allowable joint displacement per step. The second head predicts a scalar blending coefficient

$$\lambda_t = \sigma(W_\lambda h_t + b_\lambda), \tag{8}$$

with learnable parameters W_λ and b_λ, and sigmoid activation $\sigma(\cdot)$ constraining $\lambda_t \in [0, 1]$. The gate λ_t continuously interpolates between the planner reference and the reactive correction to form the nominal executed update

$$\Delta q_t^{exec} = (1 - \lambda_t)\, \Delta q_t^{ref} + \lambda_t\, \Delta q_t^{local}, \tag{9}$$

where Δq_t^{ref} is the planner-guided reference step from the previous section. A small λ_t keeps the motion close to the planned trajectory, while a large λ_t shifts authority toward the reactive correction. This continuous interpolation avoids abrupt mode switches between planner-following and reactive control, producing smoother overall motion.

Two-stage policy training. We train the gated reactive policy using a two-stage curriculum. In Stage 1, **planner-imitation pretraining** teaches the policy a feasible and goal-directed motion prior using reference actions generated by cuRobo. Given a batch of B static-scene samples, the policy predicts a planner-like joint update $\widehat{\Delta q_t}^{\,\mathrm{S1}}$, supervised by the cuRobo reference action Δq_t^{ref}. The Stage-1 behavior-cloning objective is

$$\mathcal{L}_{\mathrm{S1}} = \frac{1}{B} \sum_{t=1}^{B} \left\| \widehat{\Delta q_t}^{\,\mathrm{S1}} - \Delta q_t^{ref} \right\|_2^2. \tag{10}$$

Algorithm 1 Reactive Execution with a Risk-Triggered Predictive Safety Shield

Require: Initial state q_0, goal p^{goal}, point clouds $\{pc_t\}$
Require: Reference trajectory $\{q_t^{ref}\}_{t=0}^{T}$, policy π_θ, and thresholds
Ensure: Executed trajectory $\{q_t\}_{t=0}^{T}$

for $t = 0, \dots, T-1$ **do**
  $\Delta q_t^{ref} \leftarrow q_{t+1}^{ref} - q_t$
  $s_t \leftarrow [q_t, \Delta q_t^{ref}, p_t^{ee}, p^{goal}, \tau_t, f_t^{dyn}]$
  $(\Delta q_t^{local}, \lambda_t) \leftarrow \pi_\theta(s_t)$
  $\Delta q_t^{exec} \leftarrow (1-\lambda_t)\Delta q_t^{ref} + \lambda_t \Delta q_t^{local}$
  $(c_t^{pc}, c_t^{blk}, c_t^{look}) \leftarrow \mathrm{RiskScores}(pc_t, q_t)$
  $r_t^{(1)} \leftarrow (\lambda_t > \tau_\lambda) \vee (c_t^{pc} < \tau_{pc})$
  $r_t^{(2)} \leftarrow (c_t^{blk} > \tau_{blk}) \vee (c_t^{look} = 1)$
  if $r_t^{(1)} \vee r_t^{(2)}$ **then**
    $\mathcal{A}_t \leftarrow \mathrm{SampleCandidates}(\Delta q_t^{exec}, K)$
    $\mathcal{A}_t^{safe} \leftarrow \{a \in \mathcal{A}_t : \mathrm{CollisionFree}(q_t + a, pc_t)\}$
    if $\mathcal{A}_t^{safe} \neq \emptyset$ **then**
      $J_t(a) \leftarrow \|p_{ee}(q_t + a) - p^{goal}\|_2^2 + \beta C(q_t + a) + w_{eff}\|a\|_2^2$
      $\Delta q_t^{final} \leftarrow \arg\min_{a \in \mathcal{A}_t^{safe}} J_t(a)$
    else
      $\Delta q_t^{final} \leftarrow \mathbf{0}$
    end if
  else
    $\Delta q_t^{final} \leftarrow \Delta q_t^{exec}$
  end if
  $q_{t+1} \leftarrow q_t + \Delta q_t^{final}$
end for
return $\{q_t\}_{t=0}^{T}$

This stage allows the policy to acquire stable goal-reaching behavior before learning how to respond to dynamic obstacles.

In Stage 2, **dynamic-aware reactive fine-tuning** initializes the policy from the Stage-1 checkpoint and fine-tunes it end-to-end in dynamic-obstacle scenes. The policy jointly predicts a local corrective update Δq_t^{local} and a continuous gate λ_t. Let Δq_t^{*} denote the dynamic-scene corrective-action target and λ_t^{*} denote the danger-aware gate target derived from near-future collision risk. The Stage-2 objective combines action regression and gate supervision:

$$\begin{aligned} \mathcal{L}_{\mathrm{S2}} &= w_{\mathrm{act}}\mathcal{L}_{\mathrm{act}} + w_{\mathrm{gate}}\mathcal{L}_{\mathrm{gate}}, \\ \mathcal{L}_{\mathrm{act}} &= \frac{1}{B}\sum_{t=1}^{B}\left\|\Delta q_t^{local} - \Delta q_t^{*}\right\|_2^2, \\ \mathcal{L}_{\mathrm{gate}} &= -\frac{1}{B}\sum_{t=1}^{B}\left[\lambda_t^{*}\log\lambda_t + (1-\lambda_t^{*})\log(1-\lambda_t)\right], \end{aligned} \tag{11}$$

where w_{act} and w_{gate} balance motion correction and gate prediction. Stage 1 therefore learns how to move feasibly toward the goal, whereas Stage 2 learns when and how strongly to preserve or deviate from the planner-guided motion under dynamic disturbances. The predictive safety shield is applied only during inference and is not an additional training stage.

3.5 Reactive Execution with MPC Safety Shield

Although the gated policy can react to local obstacle motion, its nominal joint update may still become unsafe under fast or unexpected environmental changes. To mitigate this risk, PAC augments the gated policy with a lightweight predictive safety shield that is selectively activated when the current state or nominal action is identified as potentially unsafe, as summarized in Algorithm 1.

Shield activation is triggered by any one of four conditions: the gate value exceeds a high-reactivity threshold ($\lambda_t > \tau_\lambda$), the point-cloud clearance score falls below a minimum margin ($c_t^{pc} < \tau_{pc}$),

the blocking score exceeds an upper bound ($c_t^{blk} > \tau_{blk}$), or the short-horizon lookahead detects an imminent collision ($c_t^{look} = 1$). Let c_t^{pc} denote the free-space clearance measured from the observed point cloud, c_t^{blk} quantify how severely the planned path is obstructed, and $c_t^{look} \in \{0, 1\}$ be a binary flag produced by rolling out the current action over a short horizon; the scalar thresholds $(\tau_\lambda, \tau_{pc}, \tau_{blk})$ are fixed at deployment time.

Once activated, the shield samples a compact candidate set $\mathcal{A}_t = \{\Delta q_t^{(k)}\}_{k=1}^K$ of K joint updates and scores each candidate using the following composite cost:

$$\begin{aligned} J_t^{(k)} = & \left\| p_{\text{ee}}\left(q_t + \Delta q_t^{(k)}\right) - p^{\text{goal}} \right\|_2^2 \\ & + \beta\, C\left(q_t + \Delta q_t^{(k)}\right) \\ & + w_{\text{eff}} \left\| \Delta q_t^{(k)} \right\|_2^2 . \end{aligned} \tag{12}$$

where $p_{ee}(\cdot)$ evaluates forward kinematics to the end-effector position, $C(\cdot)$ measures collision risk against the current point cloud, β weights the safety penalty relative to goal-reaching progress, and w_{eff} discourages unnecessarily large joint displacements. The shield selects the minimum-cost action from the collision-free subset:

$$\Delta q_t^{final} = \arg \min_{\Delta q_t^{(k)} \in \mathcal{A}_t^{safe}} J_t^{(k)}, \tag{13}$$

where $\mathcal{A}_t^{safe} \subseteq \mathcal{A}_t$ retains only the candidates that pass a collision-free check. If the shield is not activated, the robot executes Δq_t^{exec} directly. This selective design keeps inference overhead low in safe regions while providing a reliable safety backstop whenever the policy encounters high-risk situations.

4 Experiments

We evaluate the proposed planner-augmented reactive policy in simulated manipulation environments containing both static and dynamic obstacles. The experiments are designed to answer three questions: (1) whether planner guidance improves goal-reaching reliability, (2) whether the learned gate improves dynamic obstacle avoidance, and (3) whether predictive safety correction further improves robustness during execution.

4.1 Experimental Setup

All experiments are conducted in simulation using a Franka Panda manipulator. Each episode starts from an initial joint configuration, and the robot must reach a target end-effector pose while avoiding static and dynamic obstacles. At each timestep, the controller receives the state s_t and predicts a local corrective motion Δq_t^{local} and a gate value λ_t. The final executed motion is computed using Eq. (9). An episode is considered successful if the robot reaches the goal without collision before the maximum horizon is exceeded.

4.2 Dataset Construction

We construct a synthetic manipulation benchmark using a Franka Panda manipulator with static scene geometry, target end-effector poses, and optional dynamic obstacles. For each episode, we sample a scene layout and use cuRobo (Sundaralingam et al., 2023) to generate a collision-free reference trajectory under static assumptions. During trajectory replay, dynamic obstacles follow parameterized motions, and we record robot states, planner references, end-effector and goal poses, obstacle states, point-cloud observations, and safety labels. These samples are used for behavior cloning pretraining and danger-aware reactive training.

Each valid timestep is treated as a training sample. The planner reference action is

$$\Delta q_t^{ref} = q_{t+1}^{ref} - q_t, \tag{14}$$

where q_t is the executed configuration and q_{t+1}^{ref} is the next configuration on the planner trajectory. We additionally provide normalized trajectory progress $\tau_t = t/T$ and relative obstacle features.

Table 1: Success rates (%) of PAC and the baselines across six scene families. The best result in each column is shown in bold.

Method	B3	DGB	FO	GB	SS	SO	Overall
cuRobo	**100.0**	**75.0**	**100.0**	5.0	95.0	0.0	62.5
cuRobo + Replan	**100.0**	**75.0**	**100.0**	5.0	95.0	0.0	62.5
Stage-1 BC	**100.0**	**75.0**	**100.0**	5.0	90.0	0.0	61.7
Stage-2 Policy	**100.0**	40.0	**100.0**	10.0	**100.0**	0.0	58.3
PAC	**100.0**	71.9	**100.0**	**63.0**	**100.0**	**60.0**	**82.5**

Table 2: Ablation of PAC components across the six scene families. Ref denotes planner-reference guidance, Dyn denotes dynamic-obstacle features, S1 denotes Stage-1 behavior-cloning pretraining, and Shield denotes the risk-triggered safety shield.

Components					Success Rate (%)						
Ref	Dyn	Gate	S1	Shield	B3	DGB	FO	GB	SS	SO	Overall
×	✓	✓	✓	✓	90.0	20.0	95.0	5.0	50.0	0.0	43.3
✓	×	✓	✓	✓	**100.0**	**75.0**	**100.0**	5.0	95.0	0.0	62.5
✓	✓	×	✓	✓	**100.0**	**75.0**	**100.0**	5.0	95.0	0.0	62.5
✓	✓	✓	×	✓	**100.0**	60.0	**100.0**	5.0	95.0	0.0	60.0
✓	✓	✓	✓	×	**100.0**	55.0	95.0	5.0	95.0	0.0	58.3
✓	✓	✓	✓	✓	**100.0**	71.9	**100.0**	**63.0**	**100.0**	**60.0**	**82.5**

The benchmark contains six scene families spanning static geometric constraints and dynamic disturbances: Box3 (B3), Static Shelf (SS), Goal Blocking (GB), Dynamic Goal Blocking (DGB), Floating Obstacle (FO), and Sudden Obstacle (SO). Detailed scene definitions and visualizations are provided in Appendix B.

4.3 Evaluation Metrics

We report success rate as the primary metric. A rollout is successful if the end-effector reaches the target pose without collision. We also track failure types, including static collision, dynamic collision, and goal miss. These metrics distinguish between failures caused by unsafe motion and failures caused by insufficient task progress. For the gate analysis, we additionally measure mean joint acceleration, mean and 95th-percentile joint jerk, action discontinuity, joint-space and end-effector path lengths, and minimum obstacle clearance. Acceleration and jerk are computed by finite differences of the executed joint trajectories using the simulator time step.

4.4 Baselines

We compare PACagainst planner-based and learning-based baselines: 1) **cuRobo.** The robot directly executes the reference trajectory generated by cuRobo without reactive correction. This baseline represents classical planning under static assumptions. 2) **cuRobo + Replan.** The planner replans when the trajectory is invalidated by obstacles. This baseline evaluates whether repeated planning alone is sufficient for dynamic scenes. 3) **Stage-1 BC.** A neural policy trained only with behavior cloning from planner trajectories. This baseline tests whether imitation of the planner is sufficient without danger-aware training. 4) **Stage-2 policy.** The gated reactive policy after danger-aware training, but without predictive safety correction. 5) **PAC.** The full method, combining planner reference guidance, gated local correction, two-stage training, and predictive safety correction.

4.5 Main Results & Ablation Study

Table 1 reports success rates across all scene families. Classical planning performs strongly in static environments but degrades in dynamic scenarios where moving obstacles invalidate the reference

trajectory. The Stage-1 BC model inherits the planner behavior but cannot reliably react to dynamic intrusions. The Stage-2 policy improves adaptability, but still fails in challenging cases where local prediction alone is insufficient. PACreaches the best overall success rate. The gain is most visible in goal-blocking and sudden-obstacle scenarios, where the policy must deviate from the reference trajectory while still preserving task progress.

We conduct ablations in Table 2 to analyze the contribution of PAC's components. Removing the planner reference greatly reduces performance, showing that global guidance is essential for keeping feasible task-directed motion. Removing dynamic obstacle features prevents the policy from anticipating obstacle motion. Removing the gate reduces the model to a less adaptive execution strategy, confirming the importance of continuous interpolation between planner motion and local correction. Finally, removing Stage-1 pretraining destabilizes training because the policy no longer receives a strong initialization from planner-generated trajectories.

Gate comparison protocol. We isolate the effect of gate parameterization by disabling the MPC shield for all compared variants. Continuous blending, hard switching, and planner-only execution use the same policy checkpoint, scene manifest, initial states, and 120 matched evaluation episodes (20 episodes per scene family, seed 42). The hard-switch variant differs from continuous blending only by discretizing the predicted gate at the fixed threshold $\tau = 0.5$. We report pooled episode counts and paired bootstrap confidence intervals over the matched episodes using 10,000 resamples. These intervals capture evaluation-episode variability for a fixed checkpoint, rather than variability across independently trained models.

Continuous gate versus hard switching. In a controlled no-shield comparison using the same checkpoint and 120 matched episodes, continuous gating outperforms hard switching in success rate (57.5% vs. 54.2%) and reduces mean joint jerk from 53.27 to 28.47. Paired bootstrap analysis confirms improvements in both mean jerk (-24.80, 95% CI: $[-56.05, -1.29]$) and success ($+3.3$ percentage points, 95% CI: $[0.8, 6.7]$). The two methods remain statistically comparable in mean acceleration, 95th-percentile jerk, action discontinuity, and minimum clearance. Continuous gating increases joint-space and end-effector path lengths by only 0.080 and 0.011 m, respectively. These results establish continuous gating as the preferred arbitration strategy, providing smoother motion and higher success at a modest path-efficiency cost.

Safety–completion trade-off. Within the no-shield gate comparison, planner-only execution achieves the highest overall success rate (62.5%), compared with 57.5% for continuous gating and 54.2% for hard switching. However, planner-only execution also exhibits the highest dynamic-collision rate (4.2%), whereas continuous gating avoids dynamic collisions in all 120 matched episodes. This difference is most pronounced in the dynamic goal-blocking family: planner-only execution achieves 75% success but incurs 25% dynamic collisions, while continuous gating obtains 45% success, 0% dynamic collisions, and 55% goal misses. Hard switching obtains 25% success, 10% dynamic collisions, and 65% goal misses. These results indicate that reactive arbitration shifts some collision failures toward safer but incomplete executions. Continuous blending performs this trade more effectively than hard switching, although it remains more conservative than planner-only execution.

5 Conclusion

We presented PAC, a hybrid framework that combines globally goal-directed motion planning with learned reactive control for dynamic manipulation. Through continuous risk-aware gating, PAC preserves planner guidance when it is reliable and progressively increases local correction as the environment becomes hazardous, with a predictive safety shield providing additional intervention in critical states. Across diverse static and dynamic scenarios, PAC achieves an overall success rate of 82.5% and consistently improves robustness over planner-only and learning-based baselines. These results support a simple principle: planning and reaction should not be competing control modes, but complementary sources of guidance whose roles adapt with risk. PAC provides a step toward robotic manipulation that remains goal-directed when possible and reacts safely when necessary.

AI USE STATEMENT

Generative AI tools were used for language editing, structural organization, LaTeX formatting, literature discovery, research brainstorming, manuscript drafting, and preliminary code debugging. All AI-assisted content, including methodological suggestions, text, code, citations, and experimental descriptions, was reviewed and verified by the authors against the implemented system and original experimental records. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies collision-aware robotic manipulation and does not involve human-subject data or personally identifiable information. The reported experiments are conducted in simulation. Although PAC is designed to reduce empirical collision risk, it does not provide a formal safety guarantee for unconstrained real-world deployment. Physical deployment should therefore include workspace limits, emergency-stop mechanisms, collision monitoring, and appropriate human supervision.

REPRODUCIBILITY STATEMENT

The PAC architecture, gating mechanism, safety-shield formulation, training objectives, and inference procedure are described in Section 3 and Algorithm 1. The experimental environments, obstacle configurations, baseline definitions, evaluation metrics, and training protocol are reported in Section 4 and the appendix. The appendix further provides scene-family definitions, matched evaluation conditions, training hyperparameters, safety-shield parameters, runtime analysis, and compute resources. Random seeds and evaluation episode counts are reported for the main comparisons and ablation studies to support reproducibility.

REFERENCES

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. RMPflow: A computational graph for automatic motion policy generation. In *International Workshop on the Algorithmic Foundations of Robotics*, pp. 441–457. Springer, 2018.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Robotics: Science and Systems*, volume 12, pp. 10–15607, 2016.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model, 2023.

Adam Fishman, Adithyavairavan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter Fox. Motion policy networks. In *Conference on Robot Learning*, pp. 967–977. PMLR, 2023.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 2002.

Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3067–3074. IEEE, 2015.

Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. ManiSkill2: A unified benchmark for generalizable manipulation skills, 2023.

Shixiang Shane Gu, Timothy Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation*, pp. 4569–4574. IEEE, 2011.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Quan Li, Ji Gan, Hao Zhu, and Xinbo Gao. PointACT: Automatic robot manipulation with cross-modal action chunking transformer on image-point cloud. In *2025 IEEE Smart World Congress (SWC)*, pp. 1426–1433. IEEE, 2025.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning, 2021.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy, 2024.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494. IEEE, 2009.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. CuRobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8112–8119. IEEE, 2023.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017a.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721. IEEE, 2017b.

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, 2025.

Jiafan Zhang. ACT-TSA: Action chunking transformer with two-stage attention for temporal multi-modality in bimanual manipulation tasks. In *2025 IEEE 7th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pp. 764–772. IEEE, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

A Training and Evaluation Details

This section provides additional details on the two-stage training procedure and the evaluation protocol used throughout our experiments.

A.1 Two-Stage Policy Training

PAC is trained using a two-stage curriculum that separates learning goal-directed nominal behavior from learning risk-aware reactive corrections. Stage 1 initializes the policy from planner-generated trajectories, while Stage 2 fine-tunes the resulting policy in dynamic obstacle scenarios.

Stage 1: Planner-imitation pretraining. Stage 1 teaches the policy a feasible and goal-directed motion prior from collision-free trajectories generated by cuRobo. Given a static-scene training sample, the policy predicts a joint-space action $\widehat{\Delta q}_t^{\,\mathrm{S1}}$ and is supervised by the planner reference action $\Delta q_t^{\mathrm{ref}}$. The objective is

$$\mathcal{L}_{\mathrm{S1}} = \frac{1}{B}\sum_{t=1}^{B}\left\|\widehat{\Delta q}_t^{\,\mathrm{S1}} - \Delta q_t^{\mathrm{ref}}\right\|_2^2, \tag{15}$$

where B denotes the batch size. This stage provides a stable initialization that favors feasible motion and progress toward the goal before introducing dynamic-obstacle interactions.

Stage 2: Dynamic-aware reactive fine-tuning. Stage 2 initializes the policy from the Stage 1 checkpoint and exposes it to dynamic-obstacle scenes. The policy jointly predicts a local corrective action $\Delta q_t^{\mathrm{local}}$ and a continuous gate λ_t, allowing it to determine when planner guidance should be preserved and when stronger reactive correction is required.

Training jointly supervises the corrective action and gate using

$$\mathcal{L}_{\mathrm{S2}} = w_{\mathrm{act}}\mathcal{L}_{\mathrm{act}} + w_{\mathrm{gate}}\mathcal{L}_{\mathrm{gate}}, \tag{16}$$

where $\mathcal{L}_{\mathrm{act}}$ supervises the dynamic-scene corrective action and $\mathcal{L}_{\mathrm{gate}}$ supervises the continuous gate using danger-aware targets. This stage teaches PAC to preserve planner-guided motion in nominal states while progressively increasing local correction as the execution becomes hazardous.

The predictive safety shield is not part of either training stage. It is applied only during inference when the predefined risk-trigger conditions are satisfied. Its implementation details are provided in Appendix G.2.

A.2 Evaluation Protocol

We evaluate all methods in closed-loop simulation using predetermined test-scene manifests. Methods within the same comparison are evaluated on matched episodes with identical initial robot configurations, goal poses, and obstacle configurations. Random seeds and episode manifests are fixed before evaluation so that differences in performance reflect the execution method rather than variation in sampled scene difficulty.

Success and failure criteria. Success rate is the primary evaluation metric. An episode is considered successful when the robot reaches the prescribed goal tolerance without an invalid termination. Failed episodes are categorized into three types: static collision, dynamic collision, and goal miss. Static and dynamic collisions distinguish failures involving stationary scene geometry from those caused by time-varying obstacles. A goal miss occurs when the robot remains collision-free but fails to satisfy the goal condition within the execution horizon.

Motion-quality metrics. In addition to task success, we evaluate the quality of the executed motion using mean joint acceleration, mean and 95th-percentile joint jerk, action discontinuity, joint-space path length, end-effector path length, and minimum obstacle clearance. Acceleration and jerk are computed using finite differences of the executed joint trajectory with the simulator timestep. These metrics distinguish successful but aggressive avoidance behavior from smoother goal-directed execution.

Matched comparisons. All methods reported within the same comparison use the same evaluation episodes and initial conditions. The number of episodes used for each reported experiment follows its corresponding evaluation setting and is reported together with the result. This matched protocol is also used for the ablation studies to isolate the contribution of individual PAC components.

B Benchmark Details

B.1 Scene Families

Figure 3 visualizes the six scene families used in our benchmark. The benchmark contains three static families and three dynamic families, covering increasingly challenging geometric and temporal obstacle interactions.

B.2 Scene-Family Definitions

The evaluation benchmark contains six scene families designed to test different geometric and temporal challenges. These families cover static constraints, moving obstacles, restricted goal regions, and unexpected obstacle appearances. Table 3 summarizes the primary challenge represented by each family.

The scene families are complementary rather than simple variations of the same obstacle configuration. B3, GB, and SS primarily evaluate motion under static geometric constraints, whereas DGB, FO, and SO emphasize time-varying interactions. In particular, SO tests limited reaction time, while DGB evaluates sustained interference near the goal. This separation enables failure modes involving static geometry to be distinguished from failures caused by dynamic obstacle motion.

Outcome definitions. An episode is considered successful when the robot reaches the prespecified goal tolerance without an invalid termination. Collision events are separated according to whether contact occurs with static scene geometry or a dynamic obstacle. A goal miss indicates that the robot remains collision-free but fails to satisfy the goal condition within the execution horizon. This category includes cases in which a reactive correction avoids an immediate obstacle but does not recover to a feasible goal-reaching trajectory in the remaining time.

Matched evaluation conditions. Methods compared within the same experiment use matched initial robot configurations, goal poses, and obstacle configurations. This controls for variation

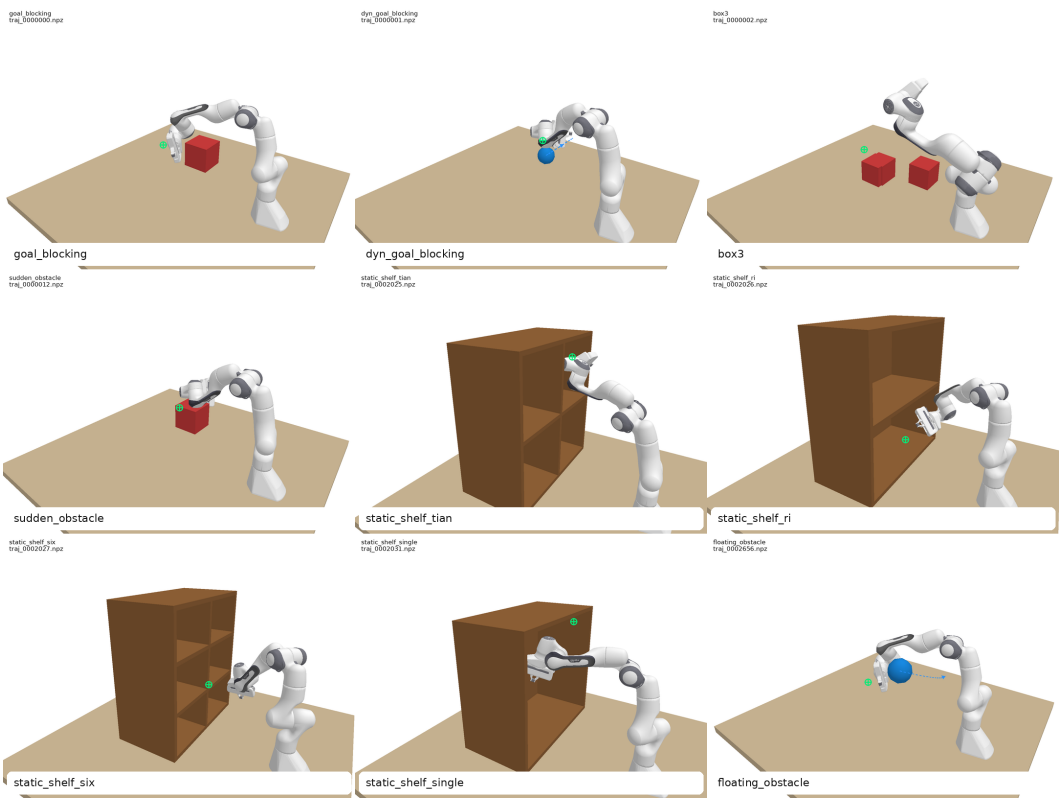


Figure 3: Representative scenes from the proposed benchmark: Box3 (B3), Static Shelf (SS), Goal Blocking (GB), Dynamic Goal Blocking (DGB), Floating Obstacle (FO), and Sudden Obstacle (SO).

Table 3: Definitions and intended challenges of the six evaluation scene families.

Abbreviation	Scene family	Description and primary challenge
B3	Three-box corridor	Three static boxes form a geometrically constrained passage. This family evaluates whether the controller can preserve sufficient clearance while following a feasible planner reference.
DGB	Dynamic goal blocking	A moving obstacle repeatedly restricts the available free space near the target. The controller must coordinate collision avoidance with continued progress toward the goal.
FO	Floating obstacle	A moving obstacle traverses the robot workspace along a predefined motion pattern. This family evaluates reactive adaptation to time-varying obstacle positions.
GB	Goal blocking	A static obstacle is placed near the target region, producing a restricted final approach. This family primarily evaluates geometric reasoning and goal-reaching precision.
SS	Static shelf	The robot operates within a shelf-like structure with limited maneuvering space. The scene tests execution under narrow static clearance and constrained approach directions.
SO	Sudden obstacle	An obstacle appears after execution has begun, leaving limited time for reactive correction. This family evaluates the response to unexpected changes that are not represented in the original planner reference.

in scene difficulty and ensures that differences reflect the execution method rather than different sampled episodes. Random seeds and episode manifests are fixed before evaluation.

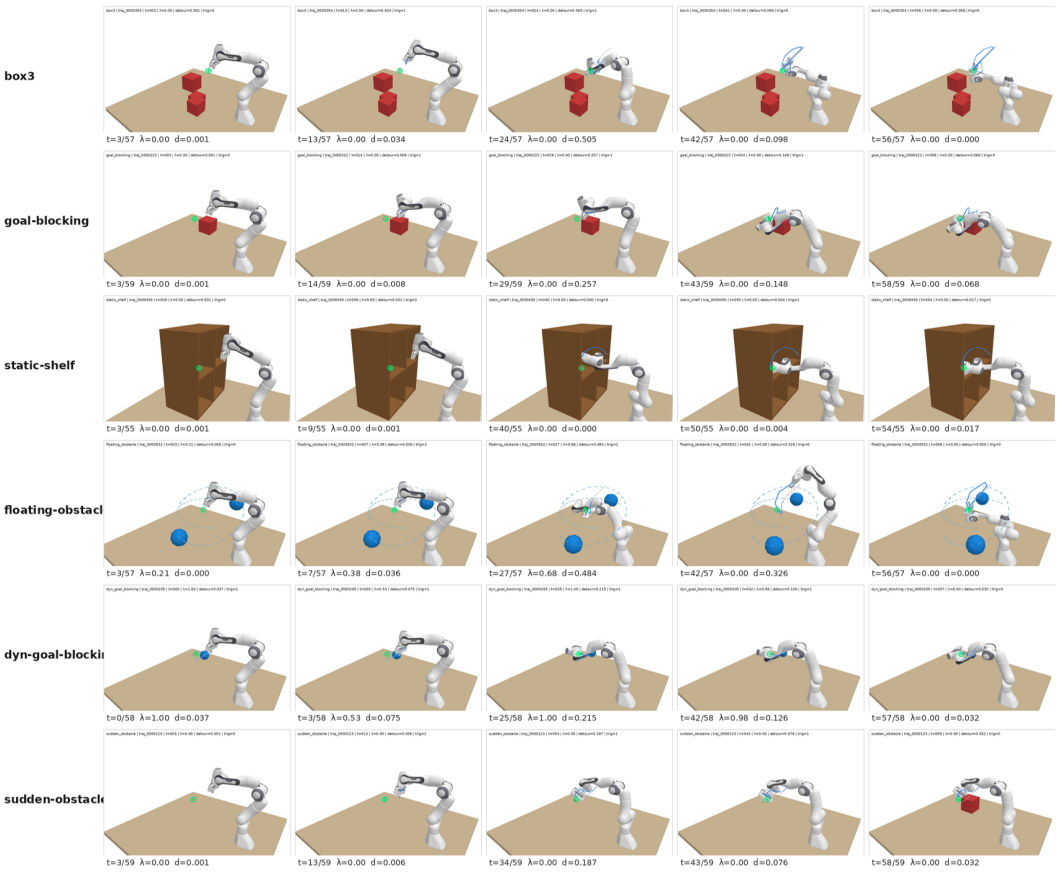


Figure 4: Representative qualitative rollouts across different scene families. Each sequence illustrates the interaction among the global planner reference, local reactive correction, and surrounding obstacle configuration during closed-loop execution.

C Additional Experimental Results

C.1 Qualitative Results

Figure 4 presents representative closed-loop rollouts from different scene families. Quantitative metrics summarize whether an episode succeeds or fails, whereas these sequences illustrate how the executed motion changes as the robot encounters different static and dynamic constraints.

In relatively unconstrained regions, the controller largely preserves the global structure supplied by the planner. When an obstacle interferes with the nominal motion, the learned policy predicts a local correction instead of regenerating the entire trajectory. This allows the system to retain goal-directed guidance while adapting its short-term motion to the current obstacle configuration.

The qualitative examples also demonstrate why combining planning and reactive control is useful. The global reference provides long-horizon task structure, while the learned correction addresses local geometric and temporal changes that may not be represented in the original plan. After the immediate conflict is resolved, the reference continues to guide the robot toward the target.

Because the reactive decision depends primarily on the relationship among the robot state, reference action, and nearby obstacles, similar corrective behavior can be reused across scene families even when their complete layouts differ.

Table 4: Relative runtime breakdown; cuRobo planning is the reference ($1\times$).

Component	Relative cost	Invoked per
cuRobo planner	$1\times$ (reference)	Task
Reactive policy inference	$\ll 1\times$	Control step
MPC shield	$\sim 0.04\times$	Triggered step

C.2 Runtime Analysis

The proposed system separates global planning and local control. The global planner is executed once per task to generate a reference trajectory, while the reactive policy operates at every control step. This design significantly reduces computational overhead, as it avoids repeatedly solving expensive optimization problems. The lightweight policy enables real-time adaptation to local disturbances, while the global plan provides long-horizon guidance.

The MPC shield is activated only at high-risk timesteps. This conditional invocation ensures that additional computation is only incurred when necessary, allowing the system to maintain efficiency in obstacle-free regions while preserving safety in challenging scenarios.

D Policy and Safety Analysis

We analyze how PAC adapts its control behavior as local risk changes. Specifically, we examine the temporal relationship between predicted danger and the continuous gate, and visualize how different risk levels shift control authority between planner guidance, reactive correction, and the predictive safety shield.

E Policy and Safety Analysis

We further analyze how PAC changes its control behavior as local risk evolves during execution. We consider two complementary perspectives. First, we examine the temporal relationship between the predicted danger score and the continuous gate to determine whether the learned controller responds to changes in local risk. Second, we interpret the resulting control behavior under low-, intermediate-, and high-risk conditions, illustrating how control authority shifts between planner guidance, reactive correction, and the predictive safety shield.

E.1 Temporal Gate and Danger Response

Figure 5 shows representative temporal profiles of the continuous gate λ_t and the predicted danger score. These signals describe related but distinct quantities. The danger score characterizes the local execution risk, whereas λ_t determines how control authority is distributed between the planner reference and the learned reactive correction.

When predicted risk is low, the controller places greater emphasis on the planner reference. In these regions, the nominal trajectory remains compatible with the observed environment, and unnecessary reactive deviation could reduce motion efficiency or goal-directed progress. The gate therefore allows PAC to retain the long-horizon structure provided by the planner.

As the robot approaches a potentially unsafe configuration, the contribution of the local correction increases. This enables the executed action to depart from the nominal reference without completely discarding planner guidance. Such behavior is particularly important for dynamic obstacles, where a reference that was feasible at planning time may become locally unsafe during execution.

After the immediate risk decreases, planner influence can increase again, guiding the robot back toward goal-directed execution. This temporal adjustment differs from hard switching between independent planning and reactive modes: PAC continuously changes their relative contributions according to the current execution context.

The danger score and gate are not expected to be numerically identical. The former represents local risk, whereas the gate additionally depends on the robot state, planner reference, and local corrective

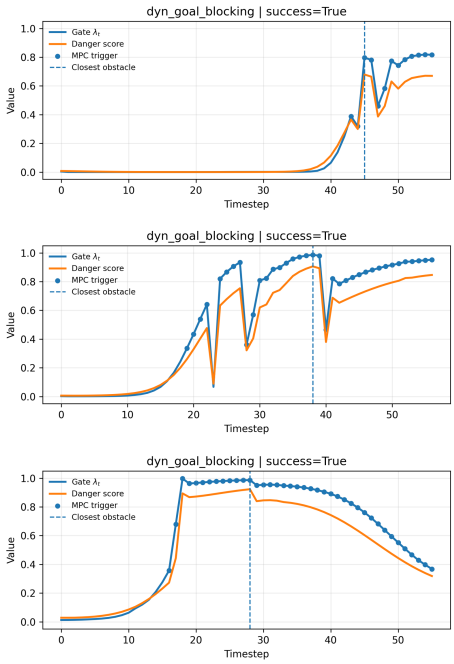


Figure 5: Temporal evolution of the continuous gate λ_t and the predicted danger score in representative evaluation episodes. Increased risk is generally accompanied by increased reliance on the local reactive correction.

action. Their temporal correspondence should therefore be interpreted as behavioral evidence that the learned gate responds to changing risk, rather than as an explicit equality between the two signals.

E.2 Control Behavior under Different Risk Levels

Figure 6 provides a conceptual interpretation of PAC under low-, intermediate-, and high-risk conditions. Unlike Figure 5, which visualizes temporal behavior from representative evaluation episodes, this illustration summarizes the role of the gate and safety shield across different operating regimes.

Low-risk behavior. Under low predicted risk, shown in Fig. 6(a), the controller primarily follows the global reference trajectory. The planner already provides a feasible and goal-directed motion prior, so strong local correction is unnecessary. Preserving the reference in these states avoids excessive deviation and maintains efficient progress toward the target.

Intermediate-risk behavior. At intermediate risk, shown in Fig. 6(b), PAC increases the contribution of the learned correction while retaining planner guidance. The executed action is formed by continuous blending:

$$\Delta q_t^{\text{exec}} = (1 - \lambda_t)\Delta q_t^{\text{ref}} + \lambda_t \Delta q_t^{\text{local}}. \tag{17}$$

The planner component preserves goal-directed structure, while the reactive component modifies the short-term motion according to the current obstacle configuration. This regime allows PAC to perform local avoidance without immediately abandoning the nominal trajectory.

High-risk behavior. At high risk, shown in Fig. 6(c), continued tracking of the nominal reference may no longer be safe. PAC therefore assigns greater control authority to the reactive correction. If the resulting motion still violates the predefined safety criteria, the predictive safety shield is activated to evaluate short-horizon alternatives before execution. The shield is thus reserved for critical states rather than being invoked continuously.

The dashed regions in Fig. 6 indicate stronger obstacle influence rather than exact collision boundaries. Together, the three regimes illustrate the central behavior of PAC: planner guidance dominates

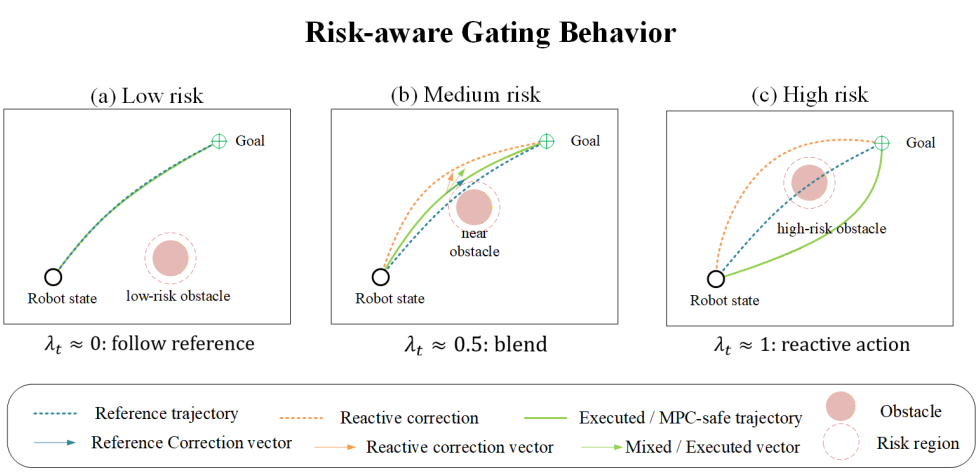


Figure 6: Conceptual illustration of risk-aware gating under low, intermediate, and high predicted risk. At low risk, execution primarily follows the planner reference. At intermediate risk, the reference and local correction are continuously blended. At high risk, greater control authority is assigned to the reactive correction, while the MPC safety shield can provide an additional short-horizon safety intervention.

when the reference remains reliable, reactive correction increases as local risk grows, and predictive intervention is reserved for safety-critical states. This organization preserves task-level guidance while allowing progressively stronger responses to dynamic disturbances.

F Failure Analysis

Failures mainly occur when reaction time or maneuvering space is limited. In sudden-obstacle scenarios, late obstacle appearance may leave insufficient time for avoidance. In dynamic goal-blocking scenarios, moving obstacles constrain the free space near the target, making collision avoidance and goal reaching difficult. These challenges are particularly severe when the robot has already committed to its nominal motion. The available execution horizon may then be insufficient for both avoidance and subsequent recovery.

In some cases, reactive correction avoids the immediate obstacle but deviates too far from the planner reference to reach the goal within the remaining horizon. Future work will explore predictive obstacle modeling, longer-horizon reasoning, and recovery-aware gating to improve both anticipation and post-avoidance recovery.

G Implementation Details

G.1 Training and Hyperparameters

All policies are trained using AdamW. Stage 1 planner-imitation pretraining uses a learning rate of 3×10^{-4}, while Stage 2 dynamic-aware fine-tuning uses a reduced learning rate of 1×10^{-4}. We use a weight decay of 10^{-4} and gradient clipping with a maximum norm of 1.0 in both stages. The reduced learning rate in Stage 2 allows the policy to adapt to dynamic interactions while retaining the goal-directed motion prior acquired during planner-imitation pretraining.

Stage 1 is trained on static-scene samples using the cuRobo reference action as supervision. The purpose of this stage is to initialize the policy with feasible and goal-directed behavior before introducing dynamic disturbances. Stage 2 is initialized from the Stage 1 checkpoint and fine-tuned on dynamic-obstacle samples. During this stage, the policy jointly learns the local corrective action and the continuous gate, enabling it to adjust the contribution of reactive control according to the current execution context.

Table 5: Predictive safety-shield configuration used in all evaluation scene families.

Parameter	Symbol	Value
Gate threshold	τ_λ	0.35
PC distance	τ_{pc}	0.14 m
Blocking threshold	τ_{blk}	0.45
Tube radius	r_{tube}	0.08 m
Candidates	K	24

The two stages therefore serve complementary roles. Stage 1 establishes a stable planner-guided motion prior, whereas Stage 2 introduces the ability to locally deviate from that prior when dynamic obstacles make the nominal motion unreliable. The predictive safety shield is not optimized as part of either training stage and is applied only during closed-loop inference.

G.2 Safety-Shield Configuration

The predictive safety shield provides an additional inference-time correction for states in which the nominal blended action is considered potentially unsafe. Unlike the learned gate, which continuously modulates planner and reactive contributions at every control step, the shield is conditionally activated only when the predefined risk-trigger conditions are satisfied. This design avoids applying the additional predictive computation during nominal execution.

When activated, the shield samples K short-horizon candidate corrections around the current action. Candidate motions that violate the collision constraints are rejected, and the remaining feasible candidates are evaluated according to the objective defined in Algorithm 1. The lowest-cost feasible candidate is selected for execution. If no feasible candidate is available, the fallback behavior defined in Algorithm 1 is applied.

The trigger combines complementary indicators of execution risk. The gate threshold τ_λ identifies states in which the learned controller already assigns substantial authority to reactive correction. The point-cloud distance threshold τ_{pc} captures geometric proximity to observed obstacles, while the blocking threshold τ_{blk} identifies configurations in which obstacle geometry interferes with nominal progress. The tube radius r_{tube} defines the local region used for trajectory-risk checking.

Table 5 summarizes the configuration used in the reported experiments.

These parameters are kept fixed across all scene families rather than being tuned separately for individual evaluation scenarios. Lower trigger thresholds generally lead to earlier and more frequent safety intervention, whereas higher thresholds preserve nominal execution for longer but leave less time for corrective action. We therefore use the same fixed configuration throughout evaluation to avoid introducing scene-specific tuning into the comparison.

Importantly, the safety shield complements rather than replaces the continuous gate. The gate handles routine adaptation by smoothly redistributing control authority between planner guidance and local reactive correction, whereas the shield is reserved for states that require an additional predictive safety check. This separation allows PAC to retain lightweight nominal execution while providing stronger intervention when the estimated risk becomes critical.

H Broader Impact

This work aims to improve the safety and robustness of robotic manipulation in dynamic environments. Potential positive impacts include safer robot operation in cluttered or human-shared workspaces. However, premature deployment of learned controllers without sufficient hardware validation may lead to unsafe behavior under perception errors, unseen obstacle motion, or actuator delays. The current system is evaluated only in simulation and should not be deployed in real-world safety-critical settings without further validation.