CROSS-ENTROPY OPTIMIZATION WITH PRIORITIZED CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

When constraints conflict, an optimizer must determine which requirements to preserve and which to relax. On the one hand, a priority ordering specifies which requirements take precedence. On the other hand, penalty-based formulations encode their relative importance through numerical weights. Depending on these weights, a solution can improve its weighted score while violating intended priorities. We introduce TierCEM, a variant of the cross-entropy method that incorporates strict constraint priorities directly into elite selection without requiring per-constraint importance weights. TierCEM works by sequentially filtering sampled candidates, from highest- to lowest-priority constraint. If and when a constraint eliminates all remaining candidates, TierCEM returns to the last nonempty set and selects elites with the smallest violations of that blocking constraint, recursively preserving satisfaction of all higher-priority constraints. We evaluate TierCEM on 2D navigation and contact-rich pushing tasks in proprioceptive and learned world-model settings. Experiments show that reversing the constraint ordering changes which constraints are violated under conflict. Prioritizing progress toward the task objective also enables TierCEM to relax lower-priority constraints when they would otherwise prevent further progress.

1 INTRODUCTION

When constraints have different priorities, optimizing a scalarized objective turns the priority order into a weighted trade-off and the optimizer can sacrifice a higher-priority constraint to reduce lower-priority violations. In autonomous driving, for example, better lane keeping must not compensate for a collision when collision avoidance takes precedence (Censi et al., 2019). A priority ordering among possibly conflicting constraints makes that distinction explicit. Satisfying a lower-priority requirement must not sacrifice a higher-priority requirement that could otherwise be preserved.

Incorporating multiple constraints into a single optimization problem is often achieved by combining the task objective and constraint penalties into a scalar score, making constraint priorities dependent on numerical trade-offs between the competing terms (Ma et al., 2022; Hu et al., 2023; Kushwaha et al., 2025). Under that task objective, an improvement in objective value or lower-priority penalties can compensate for a higher-priority violation. Preventing such compensation requires weights calibrated to both the prescribed ordering and the scales and attainable values of the competing terms, making constraint satisfaction sensitive to that calibration (Imanov et al., 2026).

Sampling-based optimizers such as the cross-entropy method (CEM) (Botev et al., 2013) and model-predictive path integral (MPPI) (Williams et al., 2017) provide a natural setting for candidate-wise constraint evaluation. In standard CEM, the top ℓ candidates according to a ranking metric are called elites and are used to update the sampling distribution in an iterative fashion. We propose the tiered CEM (TierCEM), which implements ordered constraint relaxation directly in CEM's elite selection logic through a priority-relaxation cascade (cf. Figure 1). At each iteration, TierCEM filters sampled candidate solutions from highest- to lowest-priority constraint. If a filter eliminates every remaining candidate, the cascade returns to the preceding nonempty elite set and selects samples with the smallest violations of the blocking constraint. If all filters retain candidates, those that maximize the distance to the boundary of the lowest-priority constraint are selected as elites.

The selected elites then determine the next sampling distribution through the standard CEM update. The constraint ordering thus governs which requirements are preserved and which violation

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.

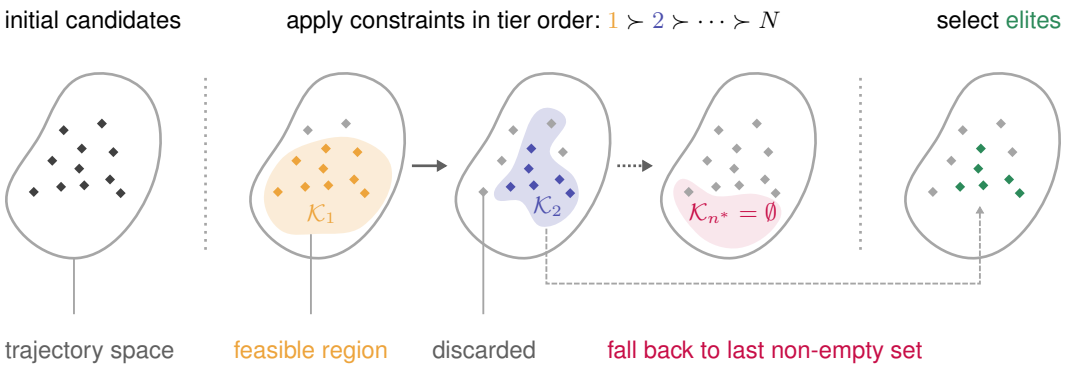


Figure 1: At each iteration, TierCEM samples candidate trajectories from a Gaussian distribution and filters them from highest- to lowest-priority constraint, producing nested candidate sets $\mathcal{K}_n$. If a tier yields an empty set ($\mathcal{K}_{n^*} = \emptyset$), TierCEM returns to the preceding nonempty set $\mathcal{K}_{n^*-1}$ and selects elites with the smallest violations of the constraint at tier n^*, preserving all higher-priority constraints already enforced. Otherwise, elites are selected from the final set according to the smallest violations of the constraint at tier N. The elites update the Gaussian sampling distribution for the next iteration; at the final iteration, a plan is selected for execution.

guides relaxation when the current sample batch contains no jointly satisfying candidate, without per-constraint importance weights.

We evaluate TierCEM in the context of trajectory optimization, where different constraint orderings lead to distinct behavioral tradeoffs. In a controlled two-dimensional navigation problem with explicit dynamics, geometric conflicts prevent simultaneous satisfaction of all spatial constraints while also reaching the goal. Reversing the constraint ordering changes which spatial constraint is violated. We then evaluate contact-rich manipulation in the Push-T environment (Chi et al., 2024), optimizing action sequences through a frozen latent-space world model and evaluating constraints with learned decoder heads that predict constraint violation. World model-based experiments reproduce the priority-dependent relaxation observed in the controlled setting: changing the ordering changes which constraint is violated, even when future states and constraint satisfaction are evaluated using imperfect learned predictions.

2 Related Work

Sampling-based optimization with constraints. Constrained CEM can separate constraint satisfaction from task reward when selecting the candidates that update the sampling distribution (Wen & Topcu, 2021). Extensions incorporate differentiable optimization (Mottahedi & Pavlak, 2022), while related constrained optimization methods adjust feasibility tolerances through adaptive ϵ-constraint handling (Liu et al., 2024). Within model-based optimization, Liu et al. (2020) use ensemble predictions to evaluate safety, selecting feasible candidates by reward and falling back to the lowest-cost candidates when no feasible sample remains. Their fallback ranks violating candidates by an accumulated constraint cost. TierCEM instead uses an ordering over individual constraints to determine which violation guides elite selection and which higher-priority requirements the selected candidates must preserve. The distinction is therefore not the existence of an infeasibility fallback, but how the prescribed priorities govern that fallback and the resulting distribution update.

Lexicographic and hierarchical constraint handling. Lexicographic optimization specifies precedence among objectives or constraints without requiring their priorities to be expressed as weighted trade-offs (Lesser & Abate, 2017; Quan et al., 2025; Lee et al., 2023). Hierarchical quadratic programming applies ordered equality and inequality constraints to humanoid motion generation (Escande et al., 2014), while Rulebooks encode priorities among traffic rules for comparing driving behaviors (Censi et al., 2019). More recently, Jia & How (2026) combine gradient-based dynamic barriers with lexicographic selection to steer trajectories output by pretrained generative robot policies. TierCEM applies the same priority principle to the finite candidate set used in each CEM update, but incorporates it directly into the CEM optimization process to determine which candidates are retained and how the sampling distribution is updated.

Algorithm 1 TierCEM

Require: ordered margins $m_1 \succ \cdots \succ m_N$,
candidate samples K, elites ℓ, iterations I

Initialize μ, Σ
for $i = 1, \ldots, I$ **do**
  Sample k candidates from $\mathcal{N}(\mu, \Sigma)$ and set $\mathcal{K}_0 \leftarrow \{1, \ldots, K\}$; $n \leftarrow 1$
  while $n \leq N$ **do**
    $\mathcal{K}_n \leftarrow$ candidates in $\mathcal{K}_{n-1}$ satisfying constraint n *Equation* (3)
    if $\mathcal{K}_n = \emptyset$ **then break** *retain* $\mathcal{K}_{n-1}$
    end if
    $n \leftarrow n + 1$
  end while
  $r \leftarrow \min(n, N)$ *n on fallback; N otherwise*
  Compute $R(k) = m_r(k)$ for $k \in \mathcal{K}_{n-1}$
  $\mathcal{K}_{\text{elite}} \leftarrow \text{Top}_\ell(\mathcal{K}_{n-1}, R)$
  Refit μ, Σ using the elite action sequences
end for
return μ, Σ

Planning and control with learned predictive models. Learned models supply predictions of future state given potential action sequences. CEM is commonly used for reward-based planning with learned dynamics models (Hafner et al., 2019; Bharadhwaj et al., 2020; Kotb et al., 2023) and safety-aware planning (Liu et al., 2020; Knuth et al., 2021). Dreamer learns policies from imagined latent rollouts (Hafner et al., 2020), while SafeDreamer incorporates safety costs and Lagrangian-based planning into world-model control (Huang et al., 2024). MUARL combines predicted safety costs and model uncertainty in sampling-based candidate selection through adaptive Lagrange multipliers (Kolling et al., 2026). Within this literature, TierCEM targets planning problems with lexicographically-ordered constraints. In our learned-model experiments, a frozen world model and learned constraint heads provide candidate evaluations, while TierCEM determines which requirements to preserve when no predicted candidate satisfies every constraint.

3 Tiered cross-entropy method

TierCEM uses an ordered sequence of constraint filters to determine which candidates update CEM's sampling distribution. The method depends only on constraint evaluations and it is independent of the candidate representation. Algorithm 1 summarizes the procedure, illustrated in Figure 1.

3.1 Priority-relaxation cascade

Problem formulation. Let $\mathcal{X}$ denote the space of candidate solutions. We consider N constraints, each represented by a margin function $m_n \colon \mathcal{X} \to \mathbb{R}$, where $m_n(x) > 0$ indicates satisfaction of constraint n and $m_n(x) \leq 0$ indicates violation. The constraints are ordered by priority, $m_1 \succ m_2 \succ \cdots \succ m_N$. Given a nonempty finite candidate set $\mathcal{K} \subset \mathcal{X}$, the goal is to select candidates according to the priority ordering; that is, to find candidates k^* solving

$$k^* \in \arg\max_{k \in \mathcal{K}} \quad m_{n^*}(k) \tag{1}$$

$$\text{subject to} \quad m_n(k) > 0, \text{ for all } n < n^* \tag{2}$$

where n^* is the highest priority constraint that *cannot be satisfied together with all higher-priority constraints* (or N, the lowest-priority constraint, if all constraints can be satisfied simultaneously).

Sampling. At each iteration, K candidates are sampled from the current Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, forming the candidate set $\mathcal{K}$. TierCEM applies the ordered constraints to $\mathcal{K}$ to determine which candidates are used as elites for updating the sampling distribution.

Filtering. TierCEM enforces constraints from highest to lowest priority by filtering the preceding candidate set. Starting from $\mathcal{K}_0 = \mathcal{K}$, tier n retains

$$\mathcal{K}_n = \{k \in \mathcal{K}_{n-1} \mid m_n(k) > 0\}. \tag{3}$$

Because each filter acts on the preceding set, every candidate in $\mathcal{K}_n$ satisfies constraints $1, \ldots, n$.

Relaxation and ranking. In CEM, candidate samples are ranked according to a metric $R(k)$ (e.g. task reward), and the best ℓ candidate statistics, called elites, are used to update the Gaussian distribution. In TierCEM, if a tier empties the candidate set, its constraint governs elite ranking while the preceding constraints remain enforced. Suppose that $\mathcal{K}_{n^*} = \emptyset$ for some n^*, and that n^* is the first such tier. TierCEM returns to $\mathcal{K}_{n^*-1}$ and ranks its candidates by

$$R(k) = m_{n^*}(k), \qquad k \in \mathcal{K}_{n^*-1}. \tag{4}$$

A larger score corresponds to a smaller worst-step violation of constraint n^*. Every candidate in the retained set still satisfies constraints $1, \ldots, n^* - 1$; lower-priority constraints do not affect elite selection in that iteration. A candidate rejected by an earlier filter therefore cannot re-enter elite selection through better lower-priority margins.

If all filters retain candidates, TierCEM ranks $\mathcal{K}_N$ by $R(k) = m_N(k)$. The ranking then favors a larger margin for the lowest-priority constraint among candidates that satisfy every constraint. In effect, this treats the function m_N as an objective function which can only be optimized once higher priority constraints are satisfied.

The cascade provides a nonempty candidate set for ranking even when no sampled trajectory satisfies every constraint. Relaxation is determined by the prescribed ordering, without comparing violation magnitudes across constraints or requiring the penalty-weight calibration used in weighted control formulations (Wojsznis et al., 2007; Qi et al., 2020).

Elite selection and refitting. TierCEM selects the highest-ranked retained candidates as elites and refits $\mathcal{N}(\mu, \Sigma)$ to their action sequences, following the standard CEM update (Botev et al., 2013). Sampling, filtering, ranking, and refitting are repeated for a fixed iteration budget of I iterations. The best elite from the final iteration is returned, as done by Walker et al. (2026).

4 Mechanism analysis under exact margins

We conduct a series of empirical studies in the domain of trajectory optimization which isolate how TierCEM resolves constraint conflicts. Specifically, we ask whether reversing the prescribed ordering changes which constraint is relaxed, and whether prioritizing task progress prevents the optimizer from preserving constraints by sacrificing progress. We study these mechanisms first in a controlled two-dimensional environment with closed-form dynamics (Section 4.3) and then in the contact-rich Push-T environment (Chi et al., 2024) (Section 4.4). Both settings use exact, noise-free margin functions; Section 5 then evaluates the same mechanism with learned margins in order to measure TierCEM's robustness to errors in learned constraint evaluations.

4.1 States, dynamics, and constraints

We instantiate TierCEM in trajectory optimization. A discrete-time predictive model $x_{t+1} = f(x_t, a_t)$ rolls the current state $x_0 \in \mathcal{X}$ forward under an action sequence $a_{0:H-1} \in \mathcal{A}^H$, producing predicted states $x_{1:H}$. The margin function for each constraint $n \in \{1, \ldots, N\}$ operates on the entire state-action trajectory. For example, a threshold $\bar{a}$ on the maximum control effort can be expressed as $m_n(x_{0:H-1}, a_{0:H-1}) = \min_t(\bar{a} - \|a_t\|)$, with H being the planning horizon.

4.2 Baselines

Scalarized CEM. We compare TierCEM against two scalarized CEM baselines that rank candidate trajectories using a weighted combination of task performance and constraint margins:

$$R^{(k)} = \underbrace{\zeta\Big(-\Big\|p_H^{(k)} - p_{\text{goal}}\Big\|_2\Big)}_{\text{task score}} + \sum_{n=1}^{N} \underbrace{\lambda_n}_{\text{manually tuned weight}} \zeta\Big(\underbrace{m_n(x_t^k)}_{\text{constraint margin}}\Big), \tag{5}$$

where $-\Big\|p_H^{(k)} - p_{\text{goal}}\Big\|_2$ is the negative terminal distance of the position p_H of the task-relevant object to the goal position p_{goal}. The trajectory-level margin for constraint n is defined as $m_n(x_{1:H}) = \frac{1}{H}\sum_{t=1}^{H} \tilde{m}_n(x_t)$, where $\tilde{m}_n(x_t)$ is the signed distance of state x_t to the infeasible region for constraint n, and $\zeta(\cdot)$ standardizes each term over the candidate population at the current CEM iteration. Soft-constraint CEM uses a shared weight, $\lambda_n = \lambda$, whereas hierarchical scalarized CEM uses separately tuned weights λ_n to encode the prescribed priorities. Setting $\lambda_n = 0$ for all n recovers task-only CEM.

Constrained cross-entropy. We also compare against CCE (Wen & Topcu, 2021), which separates feasibility from task optimization rather than combining them through a weighted objective. When too few sampled policies are feasible, CCE updates its sampling distribution according to constraint violation, once sufficient feasible policies are found, it selects according to task performance. CCE therefore provides a feasibility-first comparison, but treats the constraints jointly rather than specifying which constraint should be relaxed when they conflict.

4.3 Priority-controlled relaxation in 2D navigation

We construct a two-dimensional navigation problem in which reaching the goal requires violating one of two spatial constraints. A lane connects the start and goal but intersects a forbidden region: remaining inside the lane requires crossing the forbidden region, whereas avoiding the forbidden region requires leaving the lane. The system has single-integrator dynamics, so a zero action can maintain both spatial constraints indefinitely without making progress toward the goal.

We use this construction to separate the effect of task priority from the effect of the ordering between spatial constraints. In *task_last*, the lane and forbidden-region constraints precede task progress, allowing TierCEM to preserve both spatial constraints by stalling (Figure 2). The two task-first configurations instead require progress and reverse the spatial priorities: *task_obstacle_lane* prioritizes obstacle avoidance over lane keeping, whereas *task_lane_obstacle* does the opposite. All constraint margins are exact signed distances. Planner hyperparameters are reported in Appendix A, Table 5.

Prioritizing task progress converts constraint-preserving stalling into controlled constraint relaxation. CCE preserves both spatial constraints in every episode but never reaches the goal. In contrast, soft-constraint CEM reaches the goal in every episode but frequently violates the higher-priority constraint, since its shared weight does not distinguish the two spatial priorities.

Table 1 shows that TierCEM and hierarchical scalarized CEM both reach the goal while preserving the higher-priority spatial constraint in all 40 episodes. Their difference lies in how the priority is specified. Hierarchical scalarization requires reversing the numerical constraint weights when the

Table 1: TierCEM versus the Soft-Constraint CEM, Constrained Cross-Entropy (Wen & Topcu, 2021) (Section 4.2), and the hierarchical scalarized CEM (Section 4.2) on the task described in Section 4.3 across 20 random seeds, using 10 CEM iterations, initial Gaussian variance per dimension 1.00, planning horizon $H = 8$, 4 committed actions before re-planning, 1200 samples and 24 elites per iteration. "Reach" shows the number of seeds where the method reaches the goal out of the 20 different seeds, $V_\uparrow/V_\downarrow$ count seeds where the higher-/lower-priority constraint are violated and "Steps" reflects the average number of steps required to complete the task. Hierarchical scalarized CEM shows the weights as tuples (λ_O, λ_L) for each constraint priority order.

	`task_obstacle_lane`				`task_lane_obstacle`			
Method	**Reach**	**$V_\uparrow$**	**$V_\downarrow$**	**Steps**	**Reach**	**$V_\uparrow$**	**$V_\downarrow$**	**Steps**
TierCEM	20/20	0	20	34	20/20	0	20	32
CCE (Wen & Topcu, 2021)	0/20	0	0	160	0/20	0	0	160
Soft-constraint CEM, $\lambda = 0.5$	20/20	20	12	30	20/20	11	20	21
Hierarchical scalarized CEM, (0.5, 0.02)/(0.02, 0.5)	20/20	0	20	38	20/20	0	20	16

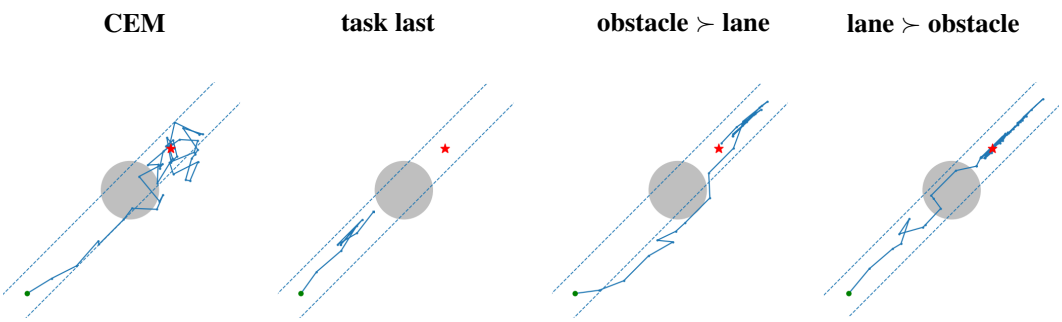


Figure 2: Effect of the prescribed priority ordering in the two-dimensional navigation problem. When task progress has lower priority than the spatial constraints, TierCEM can preserve both by stalling. Prioritizing task progress forces a spatial constraint to relax; reversing the spatial ordering reverses whether the trajectory leaves the lane or crosses the forbidden region.

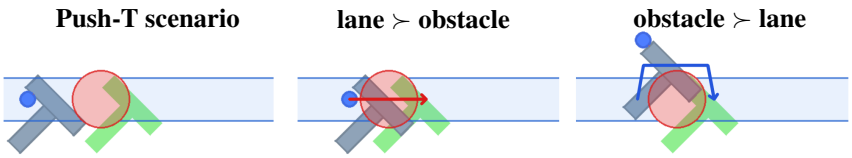


Figure 3: Visualization of the competing constraints used in the Push-T environment. The goal is to move the block toward the goal position, ignoring orientation. First image shows the initial position, the second image shows the expected behavior when staying within the lane boundary is prioritized over the obstacle constraint, and the third image shows the expected behavior when the obstacle constraint is prioritized over staying within the lane boundary.

priority ordering is reversed, whereas TierCEM changes only the ordering of its constraint filters. For TierCEM, reversing that ordering reverses which lower-priority constraint is violated (Figure 2).

The stalling mechanism depends on dynamics that admit a stationary constraint-satisfying solution; it does not arise in the inertial setting examined in Appendix E. We additionally repeat the ordering intervention under alternative constraint geometries in Appendix B.

4.4 Contact-rich manipulation with exact margins

We next test whether the same priority-controlled relaxation persists under contact-rich dynamics, which make the trajectory optimization problems far harder to solve. We transfer the lane-obstacle conflict to the Push-T environment (Chi et al., 2024), where a circular end-effector must push the block toward its goal while navigating a narrow region defined by a lane and a forbidden zone (cf. Figure 3). The geometry permits only a narrow set of trajectories that satisfy both spatial constraints while completing the task. We retain exact simulator state and constraint margins, changing the dynamics while preserving the mechanism isolated in the two-dimensional environment.

Reversing the TierCEM ordering reverses the violation pattern while leaving task performance similar. Prioritizing obstacle avoidance nearly eliminates obstacle violations while allowing the end-effector to leave the lane. In fact, prioritizing lane keeping substantially reduces lane violations while permitting obstacle violations (Table 2). Hierarchical scalarized CEM produces the same qualitative ordering-dependent behavior, but requires tuning the numerical constraint weights, a difficult process especially as the number of constraints grows and if the scaling is different among constraints. Soft-constraint CEM, which uses a shared weight, does not encode which spatial constraint should be preserved.

At the nominal goal (that is, the green region of Figure 3), CCE satisfies both spatial constraints with task error comparable to TierCEM, so this setting alone does not require constraint relaxation. We therefore increase the required task progress by moving the goal farther from the initial block position while keeping planner hyperparameters fixed.

Table 2: Exact-state (proprioceptive) evaluation on Push-T. Values are mean ± standard deviation across 20 random seeds. Position error is the final Euclidean block-center distance to the goal, violation rates are fractions of executed environment steps. Although CCE can converge to feasible solutions with a large number of candidate samples (1200 in the results of this table), its feasibility-first objective can prevent task completion under limited sampling when jointly feasible trajectories are rare. For example, with 300 samples, CCE often fails to discover task-completing trajectories and instead prioritizes constraint satisfaction (Table 4). Instead, TierCEM can relax the lower-priority constraint and retain task completion.

Method	**Config**	**Pos. error (px)**	**O viol. (%)**	**Lane viol. (%)**
TierCEM	task $\succ$ obstacle $\succ$ lane	20.63 ± 16.17	0.5 ± 1.5	64.8 ± 34.3
TierCEM	task $\succ$ lane $\succ$ obstacle	22.34 ± 12.23	44.0 ± 15.9	10.3 ± 9.8
CCE (Wen & Topcu, 2021)	joint worst violation	21.29 ± 18.13	0.4 ± 1.2	0.2 ± 1.1
Soft-constraint CEM	$\lambda = 0.5$	17.30 ± 11.99	19.5 ± 13.3	15.3 ± 11.3
Hierarchical scalarized CEM	$\lambda_O = 0.5, \lambda_L = 0.02$	15.00 ± 13.48	1.2 ± 3.4	71.7 ± 37.1
Hierarchical scalarized CEM	$\lambda_O = 0.02, \lambda_L = 0.5$	18.09 ± 10.26	29.9 ± 13.7	11.7 ± 10.9

Table 3: Goal-distance sensitivity in exact-state Push-T, with all planner hyperparameters fixed. Values are mean ± standard deviation across 20 random seeds. Position error is measured in pixels; obstacle and lane violations are percentages of executed controls. Increasing goal distance turns jointly safe satisfaction into lower task accuracy for CCE, while TierCEM can continue by relaxing its designated lower-priority lane constraint.

Method/configuration	**Metric**	**Goal** $x = 256$	**Goal** $x = 320$	**Goal** $x = 360$
TierCEM, task $\succ$ obstacle $\succ$ lane	Position error (px)	20.63 ± 16.17	16.06 ± 19.06	30.96 ± 34.31
	Obstacle violation (%)	0.50 ± 1.50	3.75 ± 6.78	3.08 ± 6.71
	Lane violation (%)	64.75 ± 34.26	71.58 ± 32.98	57.25 ± 33.70
CCE (Wen & Topcu, 2021), joint constraints	Position error (px)	21.29 ± 18.13	56.35 ± 30.77	87.80 ± 36.79
	Obstacle violation (%)	0.38 ± 1.19	1.51 ± 2.32	3.45 ± 4.56
	Lane violation (%)	0.25 ± 1.09	1.57 ± 2.59	3.34 ± 3.44

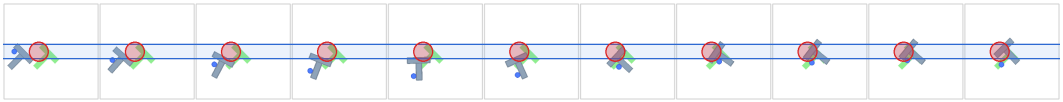
Increasing the required progress separates feasibility-first selection from ordered relaxation. CCE's final position error increases as the goal moves farther away, whereas TierCEM continues toward the goal by relaxing its designated lower-priority constraint (Table 3). Together with the two-dimensional experiment, these results show that the prescribed ordering controls which requirement is relaxed under both kinematic and contact-rich dynamics when constraint evaluations are exact.

5 Planning with learned world-model predictions

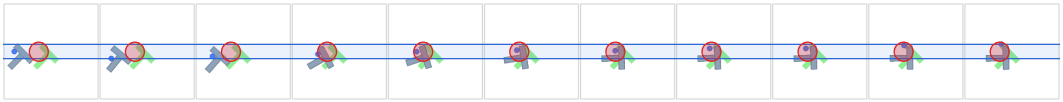
The experiments of Section 4.4 isolate the optimizer from representation and dynamics errors. We now evaluate the same conflictive constraint scenario using learned margins over DINO-WM latents (Zhou et al., 2025) with the same CEM hyperparameter configuration as Terver et al. (2026). This comparison tests whether the priority-relaxation behavior survives when candidate trajectories, task progress, and constraint margins are inferred from predicted visual latents rather than exact simulator states. TierCEM remains effective when the constraints are jointly satisfiable (Appendix I). Under imperfect constraint and dynamics models, errors in the margin function prediction can lead to conservative behavior, whereas dynamics mismatch can result in constraint violations (Appendix F).

The environment and the constraints are identical to Section 4.4. Each learned margin function is implemented as a multi-layer perceptron (MLP) that outputs the probability of violation, which we expose to TierCEM as the margin $m_n(z_t) = \min_{t=1,\dots,H}[0.5 - \mathbb{P}_n(\text{violation} \mid z_t)]$, where z_t is the latent representation of x_t. Additionally, we train an MLP regressor to predict negative distance to goal, $y_t = -\|p_{\text{block},t} - p_{\text{goal}}\|_2$. We define the task progress margin for a candidate trajectory k as the reduction in distance over the horizon: $m^{(k)}_{\text{task}} = y^{(k)}_H - y^{(k)}_0$. Positive values indicate progress toward the goal.

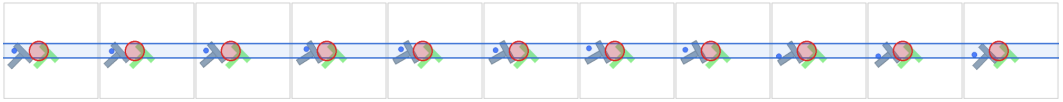
Learned margin function results. Using the MLP hyperparameters (Appendixes G and H), the MLP predicting $\mathbb{P}_n(\text{violation} \mid z_t)$ is capable of identifying violating and non-violating latents for



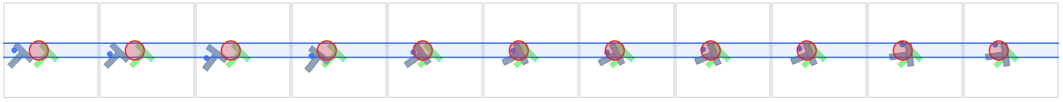
(a) TierCEM, priority order: task $\succ$ obstacle $\succ$ lane.



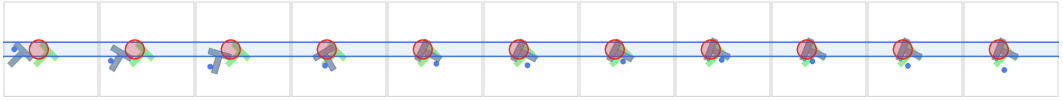
(b) TierCEM, priority order: task $\succ$ lane $\succ$ obstacle.



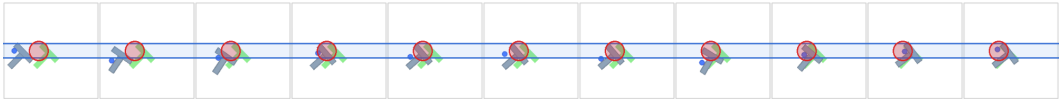
(c) Constrained Cross Entropy (CCE) (Wen & Topcu, 2021)



(d) Soft-Constraint CEM, $\lambda = 0.5$



(e) Hierarchical scalarized CEM, $\lambda_O = 0.5, \lambda_L = 0.02$.



(f) Hierarchical scalarized CEM, $\lambda_O = 0.02, \lambda_L = 0.5$.

Figure 4: TierCEM and baselines execution in the Push-T scenario using DINO-WM latents. Frames progress from left to right. The red disk blocks the blue lane, the green T region is the goal, and the blue T is the block. TierCEM relaxes the lower-priority constraint to complete the task without requiring a per-constraint weight tuning as hierarchical CEM, CCE preserves safety without progressing toward the goal, and the Soft-Constraint CEM does not have any mechanism to prioritize one constraint over the other.

Table 4: TierCEM and baseline methods on the Push-T scenario using DINO-WM latents. Values are mean $\pm$ standard deviation across 20 episodes per configuration. Position error is the final Euclidean block-center distance to the goal. O/Lane violation is the percentage of timesteps in violation.

Method	**Config**	**Pos. error (px)**	**O viol. (%)**	**Lane viol. (%)**
TierCEM	task $\succ$ obstacle $\succ$ lane	57.95 ± 11.09	10.4 ± 17.4	66.4 ± 27.3
TierCEM	task $\succ$ lane $\succ$ obstacle	45.32 ± 7.63	54.3 ± 20.9	3.2 ± 7.7
CCE (Wen & Topcu, 2021)	joint worst violation	105.11 ± 17.54	0.4 ± 1.2	1.6 ± 3.1
Soft-Constraint CEM	$\lambda = 0.5$	21.61 ± 6.26	34.8 ± 11.9	19.9 ± 5.1
Hierarchical scalarized CEM	$\lambda_O = 0.5, \lambda_L = 0.02$	46.49 ± 22.40	12.0 ± 10.8	79.2 ± 16.5
Hierarchical scalarized CEM	$\lambda_O = 0.02, \lambda_L = 0.5$	14.93 ± 12.21	47.4 ± 13.3	26.4 ± 12.3
Hierarchical scalarized CEM	$\lambda_O = 0.5, \lambda_L = 0.1$	27.80 ± 15.61	30.0 ± 11.9	51.8 ± 13.85
Hierarchical scalarized CEM	$\lambda_O = 0.1, \lambda_L = 0.5$	13.70 ± 8.17	37.8 ± 19.5	27.8 ± 11.7

the obstacle and the lane constraints, achieving strong separability of both classes for both constraints. The MLP regressor also achieves a small error in predicting the distance to the goal, with a mean absolute error of 6.2 px. Figure 9 shows a visual representation of the results obtained.

TierCEM behavior. TierCEM preserves the behavior of Section 4.4 despite errors in the learned dynamics and constraint margins. Under the configuration task $\succ$ obstacle $\succ$ lane , the end-effector goes around the obstacle and TierCEM relaxes the lane, while under task $\succ$ lane $\succ$ obstacle, the end-effector stays in the lane boundary and crosses the obstacle. This means, the higher-priority

constraint remains better satisfied while the incompatible lower-priority constraint absorbs the violation. Figure 4 shows an example trajectory generated by TierCEM and each of the baselines.

The results of Table 4 show that the behavior of hierarchical scalarization is sensitive to the choice of penalty weights. For example, under task $\succ$ obstacle $\succ$ lane, with $\lambda_{\mathrm{O}} = 0.5$ fixed, decreasing λ_{L} from 0.1 to 0.02 increases lane violations, while obstacle violations decrease. TierCEM instead only requires the priority order, and obtains lower obstacle and lane violations for the same priority configuration.

Remark 1. *The value of the weights of the hierarchical scalarized CEM do not transfer from task to task in the same environment. Table 4 shows that $\lambda_{\mathrm{O}} = 0.5, \lambda_L = 0.02$ and vice versa obtain the best results in terms of high-priority constraint satisfaction. However, $\lambda_{\mathrm{O}} = 0.5, \lambda_L = 0.1$ and vice versa is the best configuration according to Table 15 for a different Push-T variant with a non-conflictive constraint setup (Appendix I).*

6 Conclusion

When all constraints cannot be satisfied simultaneously, optimizers must determine which constraints to relax. TierCEM addresses this by encoding the constraint priorities as an explicit lexicographic ordering, rather than requiring the relative importance of constraints to be expressed through weights. Our experiments show that, when constraints conflict, TierCEM relaxes the lower-priority constraint, and reversing the priority order changes which constraint is relaxed. We also show that a feasibility-oriented optimizer without an explicit mechanism for choosing which constraint to relax, such as CCE (Wen & Topcu, 2021), are designed to sacrifice task completion in favor of constraint satisfaction. This behavior is consistent with their objective, but can produce constraint-satisfying yet task-irrelevant candidate samples. Our problem formulation allows us to express task progress as a constraint that the optimizer can prioritize, allowing task completion to be retained when constraints conflict. We show that TierCEM can function effectively even when used with learned constraint specifications and in highly complex, contact-rich trajectory optimization problems; scenarios in which TierCEM's performance depends on the accuracy of the learned dynamics and constraint models, meaning that deployment in safety-critical applications would require appropriate safety guarantees.

7 Reproducibility statement

We provide the information necessary to reproduce our experiments throughout the paper and supplementary material. Algorithm 1 gives the complete TierCEM procedure, while the experimental sections and appendices describe the environments, constraint definitions, evaluation protocols, planner configurations, random seeds, and hyperparameter settings used in our experiments. Details of the learned world-model experiments, including the training and evaluation of the constraint and task-progress heads and the construction of the Push-T dataset, are provided in Section 5 and the supplementary material. Additional per-seed results, robustness evaluations, and hyperparameter sensitivity analyses are reported in the appendices. We include the implementation and configuration files as supplementary material. Trained auxiliary models, dataset, and experiment scripts will be released after double blind.

8 LLM use statement

In this work, we used generative AI tools for coding assistance (all generated code was reviewed and verified by the first author), as a LaTeX assistant, and to review drafts of the paper and receive feedback on research methodology, experiments, formulated equations and framing of the paper. We also used LLMs for literature review with all cited material read and vetted by the first author.

9 Ethics statement

This work follows the ICLR Code of Ethics and does not involve human subjects, personal data, or other sensitive information. The proposed method is intended to provide constraint satisfaction

to model-based planning by explicitly prioritizing the most important constraints. Nevertheless, we recognize that the use of learned predictive models and safety-related constraint signals can lead to unsafe behavior under model or perception errors. The method is evaluated exclusively in simulated environments, and any deployment in safety-critical physical systems would require additional validation and appropriate safety guarantees.

REFERENCES

B. Barkley, P. Culbertson, and D. Fridovich-Keil. SCOPED: Score-curvature out-of-distribution proximity evaluator for diffusion. In *ICLR*, 2025. URL https://openreview.net/forum?id=TMLiG9Rk2J.

H. Bharadhwaj, K. Xie, and F. Shkurti. Model-predictive control via cross-entropy and gradient-based optimization. In *L4DC*, 2020. URL https://proceedings.mlr.press/v120/bharadhwaj20a.html.

Z. Botev, D. Kroese, R. Rubinstein, and P. L'Ecuyer. The cross-entropy method for optimization. *Handbook of Statistics*, 2013. doi:10.1016/B978-0-444-53859-8.00003-5.

A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *ICRA*, 2019. doi:10.1109/ICRA.2019.8794364.

C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Int. J. Robot. Res.*, 2024. doi:10.1177/02783649241273668.

A. Escande, N. Mansard, and P. Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *Int. J. Robot. Res.*, 2014. doi:10.1177/0278364914521306.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

C. Hu, J. Pei, J. Liu, and X. Yao. Evolving constrained reinforcement learning policy. In *IJCNN*, 2023. doi:10.1109/IJCNN54540.2023.10191982.

W. Huang, J. Ji, C. Xia, B. Zhang, and Y. Yang. SafeDreamer: Safe reinforcement learning with world models. In *ICLR*, 2024. URL https://openreview.net/forum?id=tsE5HLYtYg.

O. Imanov, D. Erisken, D. Kulali, T. Yilmaz, and R. Turhan. Autonomous AI agents for real-time affordable housing site selection: Multi-objective reinforcement learning under regulatory constraints. *arXiv:2602.03940*, 2026. doi:10.48550/arXiv.2602.03940.

Y. Jia and J. How. Steering generative robot policies with lexicographic preferences. *arXiv preprint*, 2026. doi:10.48550/arXiv.2609.15014.

C. Knuth, G. Chou, N. Ozay, and D. Berenson. Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants. *RA-L*, 2021. doi:10.1109/LRA.2021.3068889.

A. H. Kolling, J. C. De Jesus, V. A. Kich, R. B. Grando, M. G. Mateus, R. da Silva Guerra, and P. L. J. Drews-Jr. Evidential latent world models for safe model-based reinforcement learning. In *ICLR Workshop on World Models*, 2026. URL https://openreview.net/pdf?id=8P1pUpVKvA.

M. Kotb, C. Weber, and S. Wermter. Sample-efficient real-time planning with curiosity cross-entropy method and contrastive learning. In *IROS*, 2023. doi:10.1109/IROS55552.2023.10342018.

A. Kushwaha, K. Ravish, P. Lamba, and P. Kumar. A survey of safe reinforcement learning and constrained mdps: A technical survey on single-agent and multi-agent safety. *arXiv preprint*, 2025. doi:10.48550/arXiv.2505.17342.

J. Lee, J. Kim, and A. Ames. Hierarchical relaxation of safety-critical controllers: Mitigating contradictory safety conditions with application to quadruped robots. In *IROS*, 2023. doi:10.48550/arXiv.2305.03929.

K. Lesser and A. Abate. Multi-objective optimal control with safety as a priority. In *ICCPS*, 2017. doi:10.1145/3055004.3055008.

J. Liu, J. Feng, S. Yang, H. Zhang, and S. Liu. Dynamic ε-multilevel hierarchy constraint optimization with adaptive boundary constraint handling technology. *Applied Soft Computing*, 2024. doi:10.1016/j.asoc.2023.111172.

Z. Liu, H. Zhou, B. Chen, S. Zhong, M. Hebert, and D. Zhao. Constrained model-based reinforcement learning with robust cross-entropy method. *arXiv:2010.07968*, 2020. doi:10.48550/arXiv.2010.07968.

Y. Ma, A. Shen, O. Bastani, and J. Dinesh. Conservative and adaptive penalty for model-based safe reinforcement learning. In *AAAI*, 2022. doi:10.1609/aaai.v36i5.20478.

S. Mottahedi and G. Pavlak. Constrained differentiable cross-entropy method for safe model-based reinforcement learning. In *BuildSys*, 2022. doi:10.1145/3563357.3564055.

X. Qi, S. Li, and Y. Zheng. Enhancing dynamic operation optimization feasibility for constrained model predictive control systems. *ASJC*, 2020. doi:10.1002/asjc.2158.

Y. S. Quan, M. Jeddi, F. Prignoli, and P. Falcone. Priority-driven constraints softening in safe mpc for perturbed systems. *L-CSS*, 2025. doi:10.1109/LCSYS.2025.3580494.

B. Terver, T. Yang, J. Ponce, A. Bardes, and Y. LeCun. What drives success in physical planning with joint-embedding predictive world models? *TMLR*, 2026. doi:10.48550/arXiv.2512.24497.

M. Walker, D. Frisch, and U. Hanebeck. Sample-efficient and smooth cross-entropy method model predictive control using deterministic samples. In *ACC*, 2026. doi:10.48550/arXiv.2510.05706.

M. Wen and U. Topcu. Constrained cross-entropy method for safe reinforcement learning. *Transactions on Automatic Control*, 2021. doi:10.1109/TAC.2020.3015931.

G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control: From theory to parallel computation. *Guidance, Control, and Dynamics*, 2017. doi:10.2514/1.G001921.

W. Wojsznis, A. Mehta, P. Wojsznis, D. Thiele, and T. Blevins. Multi-objective optimization for model predictive control. *ISA Trans.*, 2007. doi:10.1016/j.isatra.2006.10.002.

G. Zhou, H. Pan, Y. Lecun, and L. Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/zhou25t.html.

A SETUP AND BASELINE DETAILS

Table 5: Base configuration. The planner and cascade parameters are those of Algorithm 1.

Parameter	Value	Description
K (samples)	1200	CEM samples per iteration
ℓ (elites)	24	elites kept per iteration
H (horizon)	8	planning horizon
iterations	10	CEM iterations
num_act_stepped	4	actions executed per replan
var_scale	1.0	proposal variance scale
velocity cap	1.2	$\pm$ action clamp
goal tolerance	0.5	arrival radius
disk	$(1.5, 1.5), r = 1.4$	obstacle
lane half-width	0.8	lane bounds

B GEOMETRY ROBUSTNESS

Table 6: Geometry robustness of the tier-order at the recommended configuration (Table 5) across 10 random seeds. The reference disk-on-lane geometry is re-run alongside three perturbations (an off-diagonal disk (center $(1.2, 2.2)$, $r = 1.3$), two overlapping disks, and a lane narrowed to half-width 0.5) by generalizing the exact signed-distance model to an arbitrary list of disks. ***R*** = seeds reaching the goal; $V_\uparrow$ = seeds where the *higher*-priority geometry constraint went negative; $V_\downarrow$ = seeds where the lower-priority constraint went negative; ***Steps*** = mean episode steps. For *task_last* the sacrificed tier is the task (both geometry constraints are protected, so $V_\downarrow$ is blank), and for *plain* the cascade is off and neither constraint is protected.

Geometry	Config	R	$V_\uparrow$	$V_\downarrow$	Steps
base (reference)	`task_obstacle_lane`	10	0	10	46.4
	`task_lane_obstacle`	10	0	10	40.0
	`task_last`	0	0	—	160
	`plain`	9	—	—	74.8
off-diagonal disk	`task_obstacle_lane`	10	0	0	42.0
	`task_lane_obstacle`	10	0	4	34.0
	`task_last`	0	0	—	160
	`plain`	10	—	—	54.8
two overlapping disks	`task_obstacle_lane`	8	2	10	70.8
	`task_lane_obstacle`	10	0	10	38.4
	`task_last`	0	0	—	160
	`plain`	9	—	—	48.4
tight lane (0.5)	`task_obstacle_lane`	2	0	10	153
	`task_lane_obstacle`	8	1	10	60.8
	`task_last`	0	0	—	160
	`plain`	10	—	—	73.2

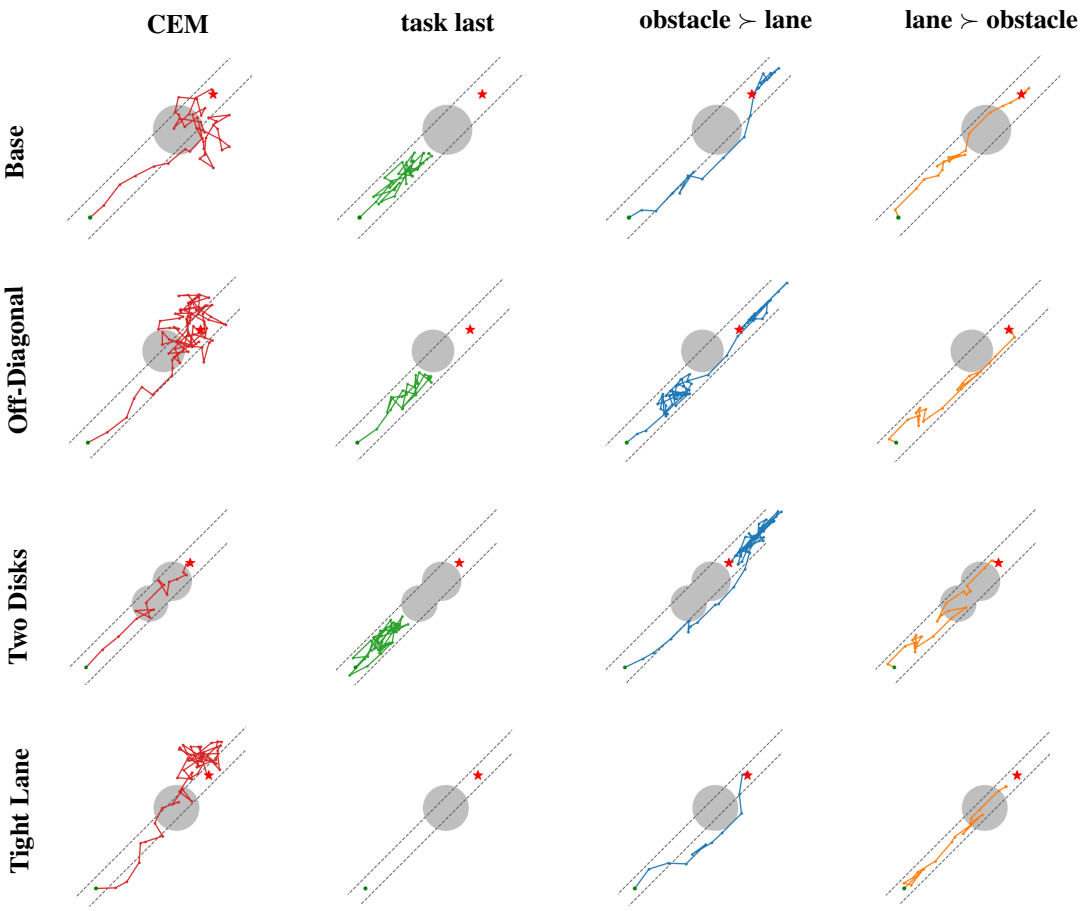


Figure 5: Visualization of trajectories for different constraint configurations: each row is a geometry, each column a configuration. The shaded disks are the obstacles, the dashed lines bound the lane are the lane boundary, the green point is the start and the red star is the goal. Across every geometry *plain* (first column) smashes through at least one constraint, *task_last* (second column) stalls preserving both, *task_obstacle_lane* (third column) arcs around the obstacle sacrificing the lane while the obstacle stays clean, and *task_lane_obstacle* (fourth column) drives straight sacrificing the obstacle while the lane stays clean.

C TierCEM hyperparameter search

Table 7 varies one hyperparameter at a time over seeds 0–4. The best results found for TierCEM were at variance 1.00, 1200 samples, 24 elites, 8 horizon, 10 iterations and commit length 4, where both experiments reach the goal in all evaluated seeds with zero high-priority violations.

Table 7: Grid Search of the mechanism analysis over 5 random seeds as one hyperparameter varies around the base configuration of Table 5. Each row lists the full hyperparameter tuple $(K, \ell, H, \mathrm{iters}, \mathrm{commit}, \sigma)$; bold marks the single value varied away from the base. R = seeds reaching the goal; $V_{\downarrow}$ = seeds where the low-priority geometry constraint went negative (lane for *task_obstacle_lane*, obstacle for *task_lane_obstacle*); $V_{\uparrow}$ = seeds where the higher-priority constraint went negative.

Setting	**Hyperparameters** $(K, \ell, H, \mathrm{it}, \mathrm{cm}, \sigma)$	`task_obstacle_lane`			`task_lane_obstacle`		
		R	$\mathbf{V}_{\downarrow}$	$\mathbf{V}_{\uparrow}$	**R**	$\mathbf{V}_{\downarrow}$	$\mathbf{V}_{\uparrow}$
base	(1200, 24, 8, 10, 4, 0.75)	2	2	0	0	0	0
samples 600	(**600**, 24, 8, 10, 4, 0.75)	2	2	0	1	1	0
samples 2400	(**2400**, 24, 8, 10, 4, 0.75)	0	0	0	0	0	0
samples 4800	(**4800**, 24, 8, 10, 4, 0.75)	0	0	0	0	0	0
elites 12	(1200, **12**, 8, 10, 4, 0.75)	0	0	0	1	1	0
elites 48	(1200, **48**, 8, 10, 4, 0.75)	0	0	0	0	0	0
elites 96	(1200, **96**, 8, 10, 4, 0.75)	0	0	0	0	0	0
var scale 0.50	(1200, 24, 8, 10, 4, **0.50**)	0	0	0	0	0	0
var scale 1.00	(1200, 24, 8, 10, 4, **1.00**)	5	5	0	5	5	0
horizon 4	(1200, 24, **4**, 10, 4, 0.75)	0	0	2	0	0	0
horizon 6	(1200, 24, **6**, 10, 4, 0.75)	0	0	0	0	0	0
horizon 12	(1200, 24, **12**, 10, 4, 0.75)	4	5	0	5	5	0
iter 5	(1200, 24, 8, **5**, 4, 0.75)	0	0	0	0	0	0
iter 20	(1200, 24, 8, **20**, 4, 0.75)	0	0	0	1	1	0
stepped 2	(1200, 24, 8, 10, **2**, 0.75)	0	0	0	0	0	0
stepped 8	(1200, 24, 8, 10, **8**, 0.75)	5	5	4	5	5	0

D BASELINES GRID SEARCH

Table 8: The Soft-Constraint CEM baseline, sweeping the scalar weight λ (same hyperparameters as Table 1 across 20 random seeds). "Reach" is the number of seeds reaching the goal, and $\mathrm{V}_{\mathrm{obs}}$ and $\mathrm{V}_{\mathrm{lane}}$ are the number of seeds where the forbidden disk and the lane are violated respectively. At every $\lambda \in [0.05, 2]$ the disk is clipped in all 20 seeds, while the lane is violated throughout; the only λ that stops the disk from being clipped makes the planner stall (reach drops to 2/20 at $\lambda = 5$).

λ	**Reach**	$\mathbf{V}_{\mathrm{obs}}$	$\mathbf{V}_{\mathrm{lane}}$
0.05	20/20	20	5
0.1	20/20	20	3
0.2	20/20	20	7
0.5	20/20	20	12
1.0	20/20	20	14
2.0	20/20	20	16
5.0	2/20	3	3

Table 9: The results of the hierarchical scalarized CEM (same hyperparameters as Table 1 across 20 random seeds). Each row is a pair $(\lambda_{\mathrm{O}}, \lambda_{\mathrm{L}})$, and `task_obstacle_lane` is run with that pair and `task_lane_obstacle` with the reversed pair, meaning the first weight is always associated with the highest priority constraint of each of the configurations. "Reach" is the number of seeds reaching the goal, $\mathrm{V}_{\uparrow}$ and $\mathrm{V}_{\downarrow}$ are the number of seeds where the highest and lower priorities are violated respectively, and "Steps" is the average number of steps executed to complete the episode.

	`task_obstacle_lane`				`task_lane_obstacle`			
$(\lambda_{\mathrm{O}}, \lambda_{\mathrm{L}})$ (ratio)	**Reach**	$\mathbf{V}_{\uparrow}$	$\mathbf{V}_{\downarrow}$	**Steps**	**Reach**	$\mathbf{V}_{\uparrow}$	$\mathbf{V}_{\downarrow}$	**Steps**
(0.5, 0.02), 25:1	20/20	0	20	38	20/20	0	20	16
(1.0, 0.05), 20:1	20/20	1	20	41	20/20	1	20	23
(0.5, 0.05), 10:1	20/20	1	20	35	20/20	0	20	25
(1.0, 0.1), 10:1	20/20	1	20	22	20/20	1	20	34
(2.0, 0.2), 10:1	20/20	0	20	32	20/20	2	20	35
(5.0, 0.5), 10:1	20/20	2	20	40	19/20	5	20	57
(0.5, 0.1), 5:1	20/20	0	20	29	20/20	6	20	34
(0.5, 0.2), 2.5:1	20/20	1	20	30	20/20	15	20	38
(0.5, 0.35), 1.4:1	19/20	0	20	39	20/20	20	17	35

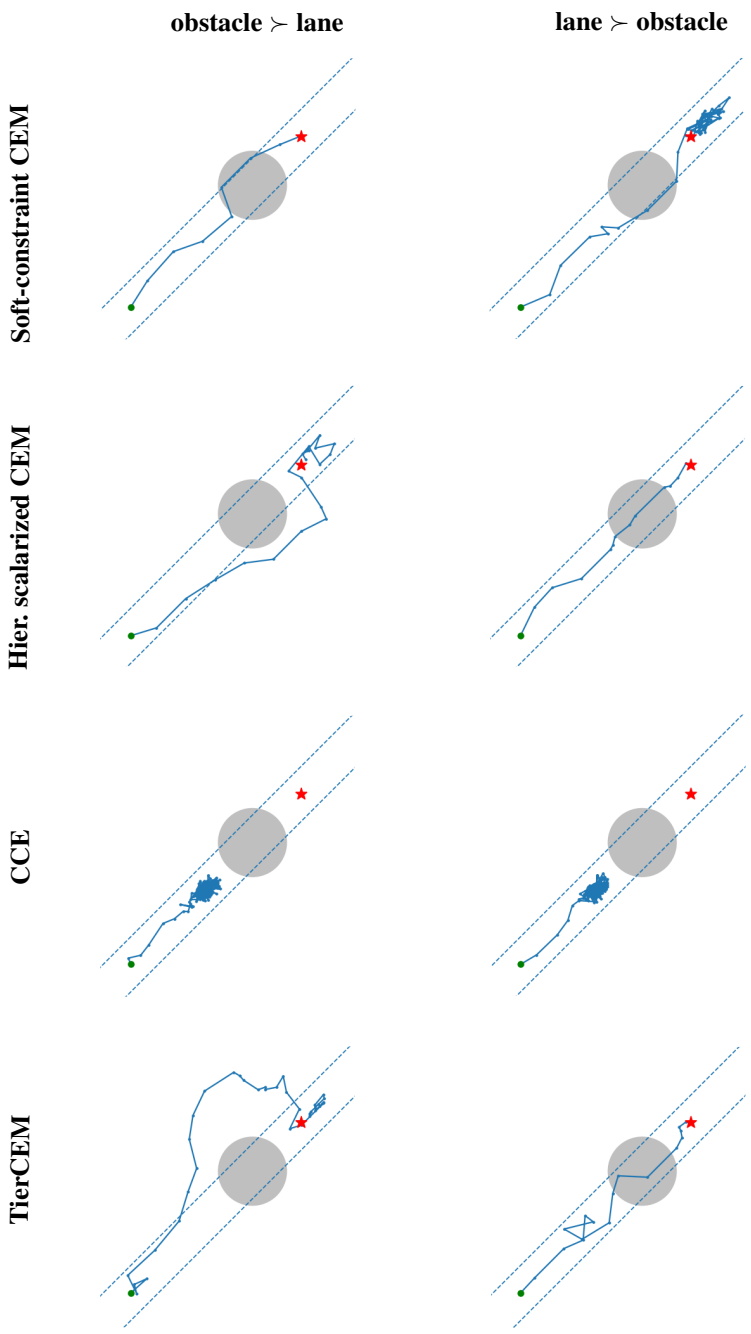


Figure 6: The soft-constraint baseline versus the lexicographic cascade on the mechanism analysis environment (rows: Soft-constraint CEM at $\lambda = 0.5$, the hierarchical scalarized CEM at the tuned 25:1 ratio of Table 9, the Constrained Cross Entropy (CCE), and TierCEM; columns: the two tier orders). The Soft-Constraint CEM carries a single scalar weight and cuts the forbidden disk (negative `min_obs`); TierCEM protects the higher-priority tier and sacrifices only the lower one. The hierarchical scalarized CEM, given per-constraint weights in the correct 25:1 ratio, also protects the higher-priority tier at this seed, but the ratio that achieves this must be tuned per environment whereas TierCEM encodes the ordering directly. CCE stalls to a safe state but never completes the task.

E TierCEM under inertial bodies

A pure hierarchy does not guarantee a tier to become empty because a no-move action keeps the current state fixed in a safe state. This is a controllability assumption, as it only holds for a position-controlled; that is, kinematic, body whose action is a velocity, but not for a body with momentum, whose action is an acceleration.

Setup. This appendix repeats the environment of Section 4.3. The kinematic baseline uses a single integrator. The state is the position $x_t \in \mathbb{R}^2$ and the action a_t is a velocity command clamped element-wise to the range $[-A_{\max}, A_{\max}]$. The rollout is $x_{t+1} = x_t + \min(A_{\max}, \max(a_t, -A_{\max}))\Delta t$ (we use $\Delta t = 1$ for simplicity), with $A_{\max} = 1.2$.

The inertial body has state (x, y, v_x, v_y) and action $a \in \mathbb{R}^2$. The rollout uses Euler integration with $\Delta t = 1$:

$$x_{t+1} = x_t + v_t, \tag{6}$$

$$v_{t+1} = v_t + \mathrm{sat}(a_t), \tag{7}$$

where $\mathrm{sat}(a_t)$ denotes element-wise saturation to $[-A_{\max}, A_{\max}]$ and $A_{\max} = 1.2$. For the "brake-only" case we further restrict $a_t \in [-A_{\max}, 0]^2$, so the body can decelerate to zero but cannot reverse its velocity. Priority order is defined as (i) obstacle and (ii) lane boundary; there is no task-progress tier. See Figure 7.

.

Results. Because the body cannot hold still under the zero action, a tier can empty when no plan keeps the body inside the constraint over the horizon. The stopping distance is what determines which constraint is sacrificed. From speed s, braking at the maximum rate leaves a residual velocity $s - k\,A_{\max}\,dt$, which determines the distance advanced along the lane, meaning the stopping distance is $E(s) = \frac{s^2}{2A_{\max}}$. When $E(s) \leq$ gap, the body can stop before the disk and arc around it to avoid it, sacrificing the lane boundary. When $E(s) >$ gap, there is no time to stop and it goes through the obstacle, as the first priority tier produces an empty set of candidates (see Table 10).

Table 10: Zero-motion escape routes are only available to kinematic bodies. For inertial bodies we report stopping distance $E(s) = s^2/(2A_{\max})$ with $A_{\max} = 1.2$. "Reach" is successes / 20 seeds. "min obs" / "min lane" are the minimum signed distances to the forbidden disk and lane boundary. Negative = constraint violation. Results are across 20 random seeds.

Body	**Speed** s	$E(s)$	**Reach**	**min obs**	**min lane**
Kinematic					
kinematic	0.4	—	0/20	$+0.59$	$+0.31$
kinematic	1.4	—	0/20	$+0.54$	$+0.31$
kinematic	2.0	—	0/20	$+0.64$	$+0.25$
Inertial, gap = 1.60					
inertial	0.4	0.07	16/20	$+0.14$	-9.23
inertial	1.4	0.82	20/20	$+0.14$	-8.37
inertial	2.0	1.67	19/20	-0.16	-7.24
inertial	2.5	2.60	20/20	-0.24	-7.60
inertial	3.0	3.75	19/20	-1.32	-4.51
Inertial, gap = 0.60					
inertial	0.4	0.07	18/20	$+0.12$	-8.25
inertial	1.4	0.82	19/20	-0.10	-8.19
inertial	1.8	1.35	18/20	-1.29	-5.28
inertial	2.0	1.67	20/20	-1.33	-6.68

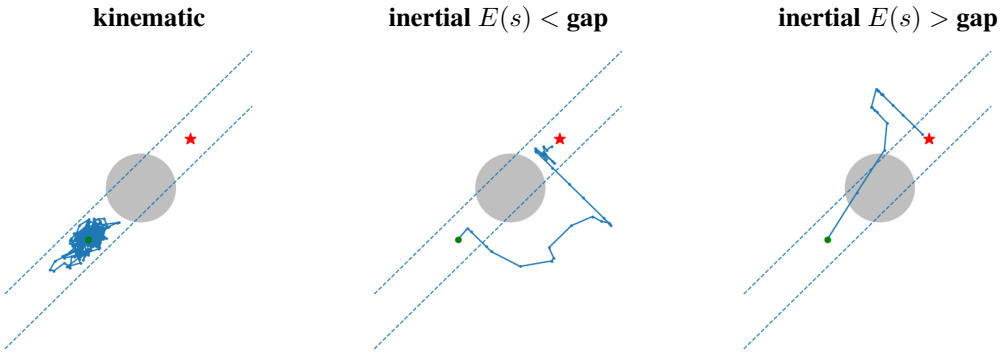


Figure 7: The kinematic control never reaches the goal (left), meaning no tier ever empties and, therefore, the planner stalls and both geometry constraints stay positive. The inertial body sacrifices the lane constraint in order to complete the goal (center), unless the gap between the starting point and the obstacle is too narrow for the initial speed s (right).

F IMPERFECT CONSTRAINT SIGNALS

This section isolates the effect of the constraint signal's quality. We use a second environment using the same setup as in Section 4 but replacing the lane boundary by a squared workspace boundary. We define six different scenarios: plain (conventional CEM), oracle (exact, noise-free margin functions), MLP heads (an MLP predicting the margin function for each constraint), MLP shifted (the same MLP trained only on the half of the workspace left of the obstacle, so the goal region and part of the forbidden disk must be extrapolated), noisy head (exact distances corrupted by $\mathcal{N}(0, 0.35)$ noise), and model mismatch (exact margins, but the optimizer's internal dynamics advances time with an 80% speed error). Figure 8 and Table 11 show that errors in the margins produce over-conservative behavior, while errors in the dynamics model produce dangerous false-confidence, violating multiple times each constraint.

Table 11: Second controlled environment (with workspace-boundary tier). *min dist* is the minimum distance to the goal along the executed path, *min obs* / *min ws* the minimum signed distance to the obstacle / workspace boundary (negative = violated). Results are across 5 random seeds.

Scenario	min dist	min obs	min ws	Verdict
`plain`	0.21	-0.85	$+0.64$	reaches goal, crashes through
`oracle`	0.23	$+0.67$	$+0.85$	reaches goal, always safe
`mlp_head`	0.23	$+0.68$	$+0.80$	reaches goal, always safe
`mlp_shifted`	3.31	$+0.61$	$+0.71$	never arrives, over-cautious
`noisy_head`	3.94	$+1.00$	$+1.25$	never arrives, over-cautious
`model_mismatch`	0.23	-0.27	-0.45	arrives, violates (3/5 seeds)

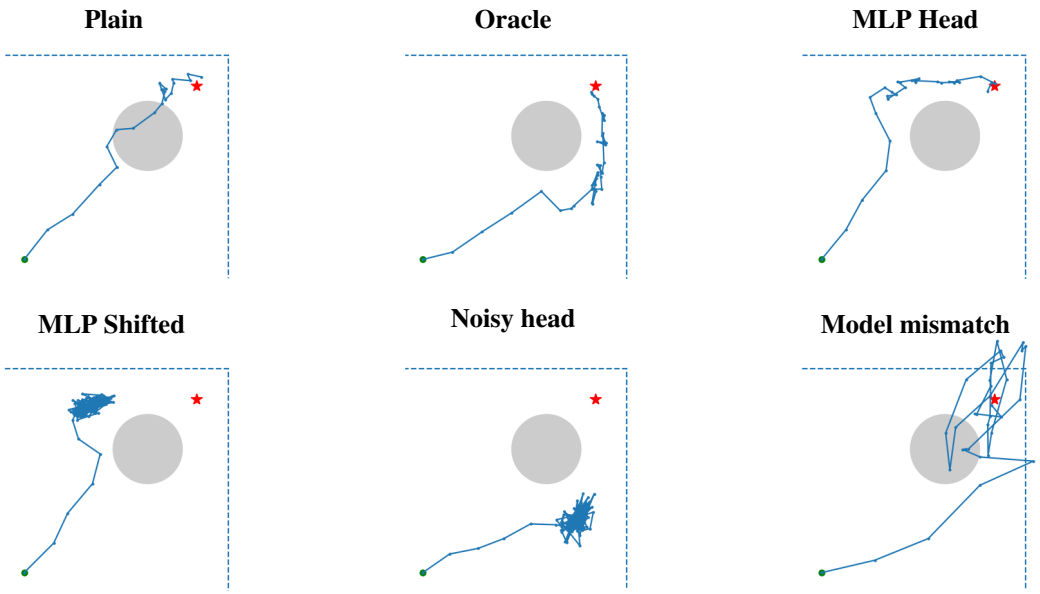


Figure 8: Imperfect constraint signals. *plain* cuts through the disk; *oracle* and *mlp_head* reach the goal safely; *mlp_shifted* and *noisy_head* stall short, hallucinating phantom unsafe regions; *model_mismatch* plans a safe arc but the executed path clips the disk and goes outside the workspace boundary occasionally.

G MLP MARGIN PREDICTOR DETAILS AND HYPERPARAMETERS

Targets and temporal alignment. The deployed model is a shared-backbone binary classifier with one output for each competing constraint; that is, the forbidden disk and the lane boundary. The exact signed margins are positive in safe states:

$$m_{\text{obs}}(s_t) = \|p_{\text{ee},t} - c_{\text{obs}}\|_2 - r_{\text{obs}}\,, \tag{8}$$

$$m_{\text{lane}}(s_t) = w_{\text{lane}} - |p^{y}_{\text{ee},t} - y_{\text{lane}}|\,, \tag{9}$$

where $p_{\text{ee},t}$ is the end-effector position, c_{obs} and r_{obs} are the forbidden-disk center and radius, and w_{lane} is the lane half-width. The binary target for constraint n is:

$$y_{t,n} = \mathbf{1}\left[\min_j m_n(s_{t+j}) < 0\right], \tag{10}$$

which means that a predicted world-model transition is labeled unsafe if the end-effector violates the constraint at any of the environment states throughout the action horizon. Training uses six auto-regressive DINO-WM steps and the action rollouts are normalized to match the distribution the planner sees during inference.

Input and architecture. Each input is a 384-dimensional mean-pooled latent predicted by the frozen DINO-WM. Input features are standardized per dimension using only the training split. The deployed shared classifier has dimensions $384 \to 256 \to 64 \to 2$, and each hidden linear layer is followed by LayerNorm, GELU, and dropout with probability 0.2. The final linear layer produces two heads, ordered as obstacle and lane boundary, and a sigmoid converts each logit to $P_n(\text{violation} \mid z_t)$. In total, this classifier contains 115,778 trainable parameters, while the DINO-WM remains frozen.

Optimization and model selection. We optimize the classifier with unweighted binary cross-entropy; no class weights or resampling are applied. AdamW starts at a learning rate of 3×10^{-4} with weight decay 10^{-4} over 200 epochs. We save and use the weights with the lowest validation binary cross-entropy and stop after 25 epochs without an improvement greater than 10^{-6}.

Table 12: Hyperparameters of the shared DINO-WM classifier used for the forbidden-zone and lane-boundary heads.

Hyperparameter	**Value**
Input dimension	384
Hidden dimensions	$(256, 64)$
Output dimension	2
Hidden activation	GELU
Normalization	LayerNorm after each hidden linear layer
Dropout	0.2
Trainable parameters	115,778
Maximum epochs	200
Batch size	1024
Optimizer	AdamW
Initial/minimum learning rate	3×10^{-4} / 10^{-6}
Learning-rate schedule	Cosine annealing
Weight decay	10^{-4}
Training loss	Unweighted masked binary cross-entropy
Early-stopping patience	25 epochs
Validation improvement tolerance	10^{-6}
Train/validation/test split	$70\%/15\%/15\%$, by episode
Split seed / training seed	42 / 42
Input standardization	Yes, training statistics only
World-model horizon	6 model steps
Violation decision threshold	$P_n(\text{violation}) \geq 0.5$

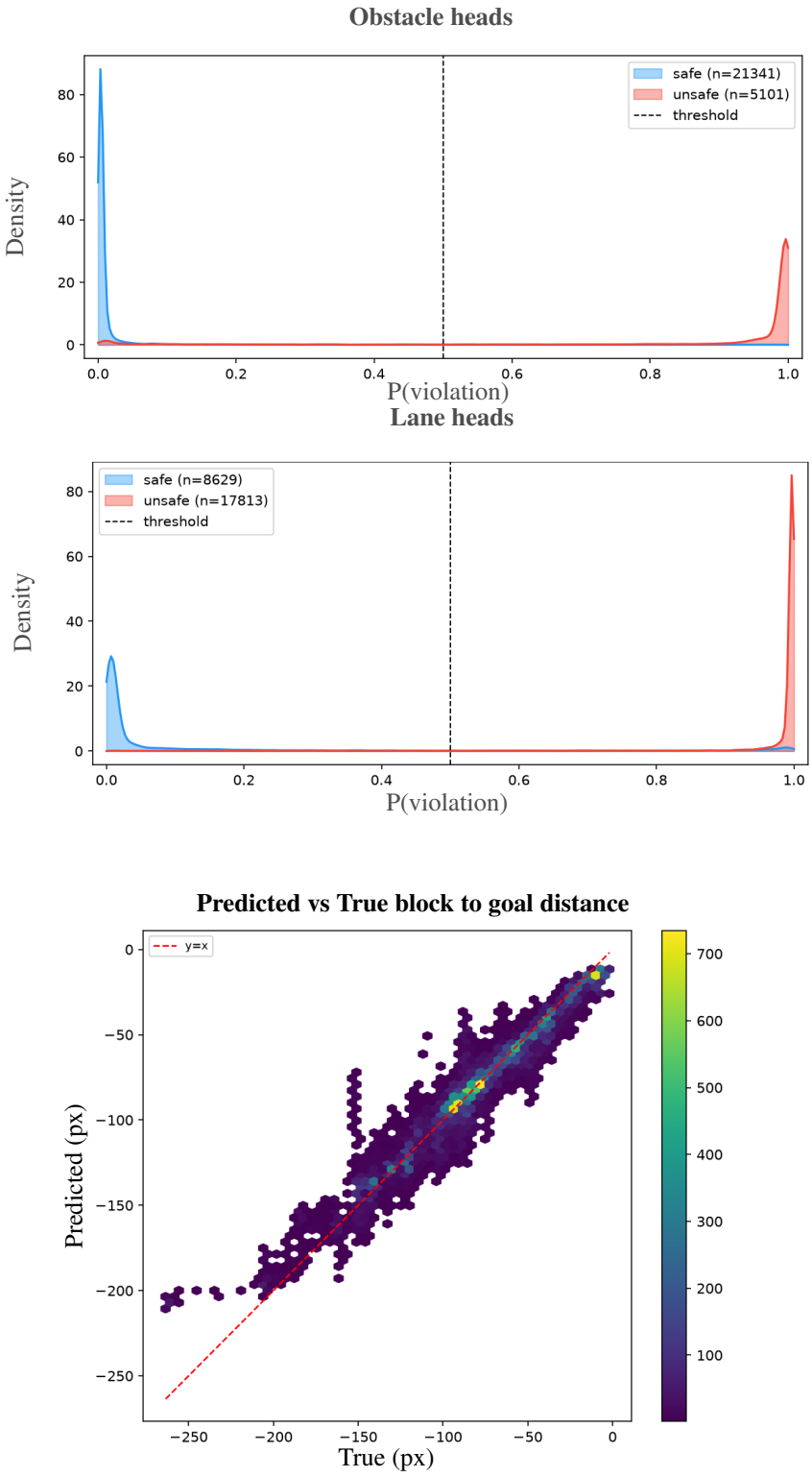


Figure 9: First two rows: resulting KDE of the MLP prediction for the forbidden disk and the lane boundary; third row: the predicted block to goal distance versus the true block to goal distance.

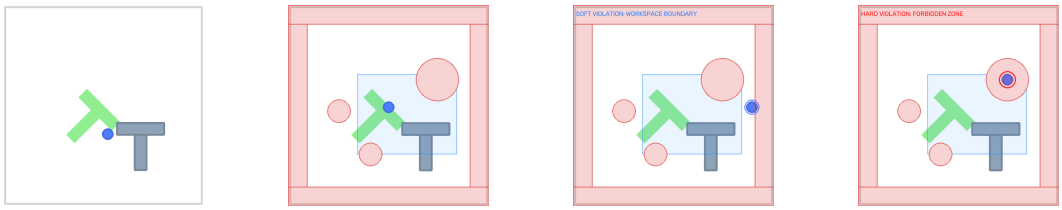
H TASK-PROGRESS MLP HYPERPARAMETERS

Input and architecture. The input is the 384-dimensional mean-pooled visual latent predicted by the frozen DINO-WM. The regressor's hidden layer dimensions are $384 \to 32 \to 32 \to 1$, with a linear layer, LayerNorm, GELU activation, and dropout with probability of 0.5 after each hidden layer. The output layer is linear and has no output activation. This model contains 13,537 trainable parameters; DINO-WM remains frozen.

Optimization and model selection. We minimize mean squared error in standardized target space with AdamW. Table 13 shows the complete configuration. We keep the checkpoint with the lowest validation MSE and stop when validation loss has not improved by at least 10^{-6} for 20 consecutive epochs.

Table 13: Hyperparameters of the DINO-WM task-progress MLP. Values are the defaults used by the fixed-conflict training pipeline.

Hyperparameter	**Value**
Input dimension	384
Hidden dimensions	$(32, 32)$
Output dimension	1
Hidden activation	GELU
Normalization	LayerNorm after each hidden linear layer
Dropout	0.5
Maximum epochs	200
Batch size	1024
Optimizer	AdamW
Learning rate	10^{-3}
Weight decay	10^{-2}
Training loss	Mean squared error
Early-stopping patience	20 epochs
Validation improvement tolerance	10^{-6}
Train/validation/test split	$70\%/15\%/15\%$, by episode
Split seed / training seed	0 / 0
Input and target standardization	Yes, training statistics only
World-model horizon	6 model steps



(a) (b) (c) (d)

Figure 10: Image representation samples of the Push-T environment. 10a shows the image observation used as an input for the world models. For the rest of images, red zones are forbidden regions, while the blue zone represents the workspace boundary. When a violation happens, it is marked around the end-effector with a circle surrounding it (red for an obstacle violation, and blue for a workspace boundary violation). 10b shows an example where all constraints are satisfied, 10c shows a workspace boundary violation, and 10d shows an obstacle violation.

I BEHAVIOR WHEN CONSTRAINTS ARE JOINTLY SATISFIABLE

This appendix reports a Push-T variant that complements the deliberately fixed-conflict experiment of Sections 4.4 and 5. Here, the constraints are an obstacle and a workspace boundary, which admit useful trajectories satisfying both constraints, and the goal is sampled from the expert dataset. The main experiment of Sections 4.4 and 5 instead uses a fixed goal and an obstacle that spans a horizontal lane, forcing an ordered relaxation for task completion.

In this new variant, we do not expect reversing the tiers to change behavior, as both can be satisfied simultaneously while being successful in the task. Its purpose is to establish the scope of the mechanism. We aim to prove that sequential filtering is a useful mechanism also when constraints do not conflict with each other. We train an MLP with a shared backbone with two heads, one for each constraint, as done in the main experiments, and we do not use a task progress margin.

We compare scalarized planners (Tables 14 and 15) and TierCEM (Table 16) under the expert-dataset goal. All results average 100 episodes over seeds 1–5. These sweeps are used to show the non-conflict regime rather than as additional evidence for constraint relaxation. In this setup, we use as $R^{(k)}_{\text{task}}$ the negative MSE of the terminal latent and the goal latent to match the trajectory-selection criterion used by DINO-WM (Zhou et al., 2025) and keep the comparison focused on the effect of the priority ordering mechanism rather than on a different task ranking.

Table 14: Soft-constraint CEM with scalarized penalty, evaluated for 100 episodes across 5 random seeds.

λ	O (%)	WB (%)	Pos Dist (px)	Angle Error (°)
0.05	10.9 ± 1.7	35.0 ± 3.4	15.2 ± 3.9	12.6 ± 3.9
0.1	10.4 ± 1.4	31.8 ± 3.5	15.9 ± 2.3	12.8 ± 2.4
0.3	10.2 ± 1.8	27.7 ± 2.6	16.7 ± 2.8	14.3 ± 2.9
0.5	10.3 ± 2.3	26.6 ± 3.2	17.3 ± 3.1	13.6 ± 3.1
1.0	11.0 ± 2.6	23.1 ± 2.4	19.2 ± 3.2	15.6 ± 4.1

Table 15: Hierarchical scalarized CEM, evaluated for 100 episodes across 5 random seeds.

λ_{O} / λ_{WB}	O (%)	WB (%)	Pos Dist (px)	Angle Error (°)
0.25 / 0.01	11.2 ± 1.7	37.5 ± 2.2	15.6 ± 2.4	13.1 ± 4.0
0.5 / 0.02	9.9 ± 1.6	34.8 ± 2.2	16.3 ± 3.1	13.7 ± 4.2
1 / 0.04	10.1 ± 1.6	32.0 ± 2.5	16.7 ± 3.3	14.0 ± 3.7
0.5 / 0.1	9.8 ± 1.8	29.2 ± 2.9	16.9 ± 2.8	13.7 ± 3.3

Table 16: Grid search over the number of lexicographic cascade iterations on Push-T without OOD detection, evaluated for 100 episodes across 5 random seeds, comparing the normal priority order against reversed priorities. Each row runs $X = 30 - \text{iterations}$ unconstrained CEM warm-up iterations followed by the remaining cascade iterations.

	Obstacle as highest priority				**Workspace boundary as highest priority**			
Cascade iterations	O (%)	WB (%)	Pos Dist (px)	Angle Error (°)	O (%)	WB (%)	Pos Dist (px)	Angle Error (°)
0	12.0 ± 1.1	41.0 ± 2.0	14.8 ± 3.2	12.3 ± 3.6	12.0 ± 1.1	41.0 ± 2.0	14.8 ± 3.2	12.3 ± 3.6
5	11.9 ± 1.2	40.7 ± 1.9	15.0 ± 3.1	12.5 ± 3.6	11.9 ± 1.4	40.7 ± 2.2	14.9 ± 3.6	12.5 ± 3.9
10	11.7 ± 1.1	39.7 ± 2.2	15.2 ± 2.6	12.6 ± 3.4	11.7 ± 1.3	39.7 ± 2.6	14.9 ± 3.2	12.4 ± 3.9
15	10.9 ± 1.0	37.2 ± 1.7	16.3 ± 2.5	13.1 ± 3.9	11.2 ± 1.4	37.6 ± 2.6	16.1 ± 3.2	12.8 ± 4.7
20	10.3 ± 1.7	33.5 ± 2.9	17.7 ± 2.9	13.5 ± 3.1	10.3 ± 1.9	33.0 ± 2.5	18.2 ± 2.6	14.0 ± 3.4
25	9.2 ± 1.6	27.1 ± 2.4	18.6 ± 3.2	14.8 ± 2.6	9.2 ± 1.7	26.9 ± 2.9	18.9 ± 3.5	15.0 ± 3.1
30	9.6 ± 1.3	20.9 ± 2.8	26.7 ± 4.1	23.0 ± 4.9	9.3 ± 1.9	19.7 ± 3.2	28.0 ± 4.0	24.4 ± 4.4

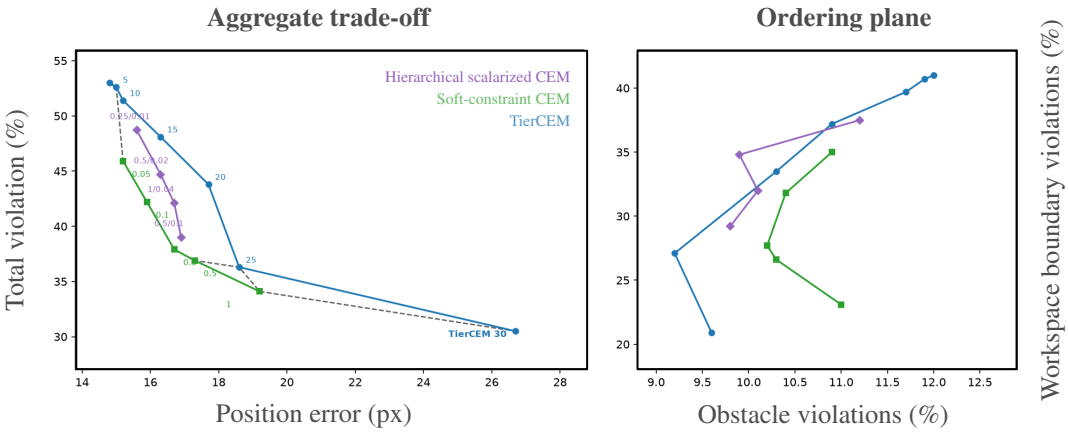


Figure 11: Pareto presentation of the Push-T sweeps of Tables 14, 15, and the obstacle as highest priority column of 16 (means over 100 episodes across 5 random seeds). These results do not use OOD detection. (a) Total violation rate against position error, the dashed step line is the Pareto front. (b) The same sweeps in the ordering plane.

I.1 Number of TierCEM iterations sweep

TierCEM can be activated after a number X of unconstrained CEM iterations. Table 16 sweeps the number of cascade iterations on the Push-T environment (100 episodes, seeds 1–5), with the remaining $30 - X$ iterations running as unconstrained CEM. Workspace-boundary violations fall monotonically from 41.0% to 20.9%, at the expense of task accuracy. It is possible therefore to choose the operating point on this trade-off while the priority ordering itself remains unchanged.

Figure 11 plots the sweeps of Tables 14, 15 and 16 together. The ordering plane of Figure 11 shows where TierCEM is beneficial with respect to the other baselines. TierCEM at 30 iterations reaches 9.6% forbidden-zone and 20.9% workspace-boundary violations, below every scalarized point on both axes, without a weight expressing the trade-off, at the 26.7 px position error that is visualized in the aggregate trade-off.

Table 16 additionally compares the priority order used in Section 5 (obstacle above workspace boundary) against the reversed order (workspace boundary above obstacle). The two columns are nearly identical across all cascade iteration counts. This is because, unlike the deliberately conflicting geometry of Section 4, here the constraints are not in conflict. TierCEM's role on the task is therefore not to choose which constraint to relax, but to enforce both constraints.

I.2 Optional third tier

We additionally implemented an exploratory three-tier ordering, OOD $\succ$ obstacle $\succ$ workspace boundary. The OOD detector was implemented as a diffusion model trained on safe DINO-WM latents that scores candidates with the SCOPED statistic (Barkley et al., 2025). We include this only as evidence that the cascade accepts a third, heterogeneous predicate without changing its selection

rule. OOD detection is not a contribution of this paper and is disabled in all quantitative results above. Results are shown in Table 17.

Table 17: Effect of OOD detection on Push-T performance across methods, DINO-WM latents, averaged across 5 random seeds. "O" and "WB" denote obstacle and workspace boundary constraint violation rates. "Pos" and "Angle" denote average position distance and orientation error of the T-block relative to the goal.

		No OOD				With OOD			
Method	**Config**	**O** (%)	**WB** (%)	**Pos.** (px)	**Angle** ($^\circ$)	**O** (%)	**WB** (%)	**Pos.** (px)	**Angle** ($^\circ$)
Baseline CEM	–	12.0±1.1	41.0±2.0	14.8±3.2	12.3±3.6	8.9±1.6	32.6±4.6	16.0±2.4	13.4±2.5
TierCEM	30 iters	9.6±1.3	20.9±2.8	26.7±4.1	23.0±4.9	7.6±1.0	19.3±2.4	24.7±3.0	21.4±3.2
Hard-constraint flat tier	30 iters	9.9±2.5	21.0±3.9	26.6±4.2	22.7±5.2	7.4±1.5	19.4±3.5	24.8±2.6	21.3±3.3
CCE (Wen & Topcu, 2021)	–	8.7±1.6	20.4±4.1	26.1±1.5	23.7±4.5	8.4±1.2	20.5±3.4	23.8±2.8	21.7±5.2
Soft-constraint CEM	$\lambda = 0.5$	10.3±2.3	26.6±3.2	17.3±3.1	13.6±3.1	9.2±1.2	23.2±3.2	18.2±1.8	14.2±3.2
Hierarchical scalarized CEM	$\lambda_{\mathrm{O}} = 0.5$ $\lambda_{\mathrm{WB}} = 0.1$	9.8±1.8	29.2±2.9	16.9±2.8	13.7±3.3	9.5±1.7	26.7±4.4	17.5±2.0	12.9±3.2