Online Learning for Obstacle Avoidance

Anonymous Author(s)
Affiliation
Address
email

Abstract: We approach the fundamental problem of obstacle avoidance for robotic systems via the lens of online learning. In contrast to prior work that either assumes worst-case realizations of uncertainty in the environment or a stationary stochastic model of uncertainty, we propose a method that is efficient to implement and provably grants *instance-optimality* with respect to perturbations of trajectories generated from an open-loop planner (in the sense of minimizing worst-case *regret*). The resulting policy adapts online to realizations of uncertainty and provably compares well with the best obstacle avoidance policy *in hindsight* from a rich class of policies. The method is validated in simulation on a dynamical system environment and compared to baseline open-loop planning and robust Hamilton-Jacobi reachability techniques. Further, it is implemented on a hardware example where a quadruped robot traverses a dense obstacle field and encounters input disturbances due to time delays, model uncertainty, and dynamics nonlinearities.

Keywords: Regret Minimization, Obstacle Avoidance, Online Learning

1 Introduction

The problem of obstacle avoidance in motion planning is a fundamental and challenging task at the core of robotics and robot safety. Successfully solving the problem requires dealing with environments that are inherently uncertain and noisy: a robot must take into account uncertainty — external disturbances and unmodeled effects, for example — in its own dynamics and those of other agents in the environment. Approaches for tackling the obstacle avoidance problem in robotics typically fall under two categories: (i) methods that attempt to construct stochastic models of uncertainty in the agents' dynamics and use the resulting probabilistic models for planning or policy learning, and (ii) methods that construct plans that take into account worst-case behavior. In Sec. 2 we give a more detailed overview of both classes of approaches.

In this paper, we are motivated by Vapnik's principle: *"when solving a given problem, try to avoid solving an even harder problem as an intermediate step."* Constructing accurate models of disturbances and agent dynamics is perhaps more complicated than the task of obstacle avoidance in motion planning, as practical uncertainties rarely conform to the assumptions made by the two classes of approaches highlighted above. As an example, consider a quadruped robot navigating through an (*a priori* unknown) obstacle field subject to unmodeled dynamics and external disturbances in the input channel (Fig. 1). Constructing an accurate probabilistic model of disturbances and obstacle variations that the robot may encounter is challenging, and may cause the robot to violate safety in out-of-distribution settings. In contrast, making worst-case assumptions may lead to extremely conservative behavior. This motivates the need for *online learning* methods that adapt to the *particular instance* of disturbances encountered by the robot.

Statement of Contributions. In this work, we pose the problem of obstacle avoidance in a *regret minimization* framework and build on techniques from non-stochastic control. Our primary contribution is a trust-region-based online learning algorithm for the task of obstacle avoidance, coupled with provable regret bounds that show our obstacle avoidance policy to be comparable to the *best policy in*

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

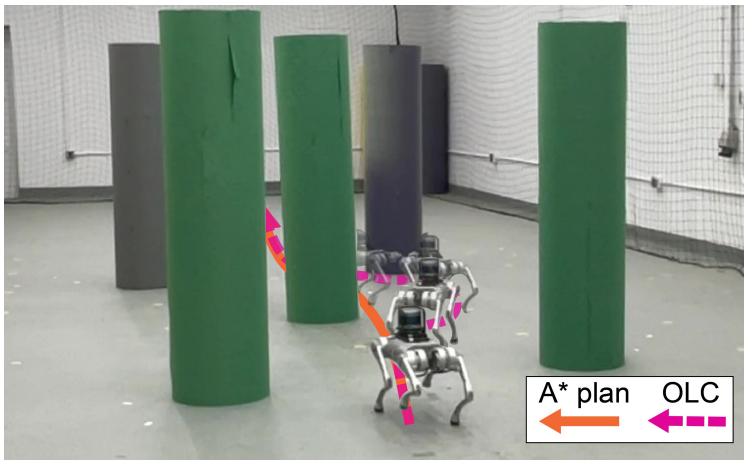

Figure 1: A quadruped robot is tasked with traversing a course with densely placed obstacles to a goal. During traversal, the robot is subject to sensor noise, time delays, input-channel disturbances, and nonlinearities in the closed-loop dynamics, which can render optimistic nominal plans (orange) unsafe. Our online learning control (OLC) algorithm (dashed pink) corrects for this to give a wider margin of error when moving through the course.

hindsight from a given class of closed-loop policies. This type of theoretical performance guarantee is nonstandard, and allows us to flexibly adapt to the behavior of the uncertainty in any *instance* of the obstacle avoidance problem without making *a priori* assumptions about whether the uncertainty is stochastic or adversarial. Further, the resulting method is computationally efficient. The method is applied to dense obstacle environments with complex unmodeled dynamics, and demonstrates improved performance where open-loop planners and overly-robust methods can respectively struggle. We additionally show the efficacy of our method with hardware demonstrations where a quadruped robot has to navigate dense obstacle fields subject to time delays and input-channel disturbances.

2 Related Work

Effective motion planning is a central challenge within robotics that continues to spur further development [1, 2, 3]. While motion planning is well-developed in deterministic settings, robust planning remains a major challenge in the presence of unmodeled or partially-modeled uncertainty. Existing robust planning techniques typically fall into one of two categories: (i) methods that make assumptions on the distribution of uncertainty, or (ii) methods that assume worst-case disturbances. Below, we set our work within this context and discuss techniques from online learning; we will leverage these techniques to develop our novel framework for obstacle avoidance.

Planning under uncertainty. A popular approach to account for uncertainty in motion planning is to assume knowledge of the uncertainty distribution. One early method in this vein utilizes *chance constraints* to bound the probability of collision under stochastic uncertainty [4] and has been subsequently extended to encompass many sources of stochastic uncertainty in robotics [5] - [8]. Further development has utilized partially observable Markov decision processes (POMDPs) to account for state uncertainty [9] - [12]. These approaches are able to provide strong guarantees on safety, albeit under generally restrictive assumptions on the uncertainty distribution (e.g., i.i.d. Gaussian uncertainty); our approach does not assume knowledge of the distribution of uncertainty, yet provides regret bounds even in the presence of non-Gaussian and non-stationary noise.

Recently, learning-based planning techniques relying on *domain randomization* have demonstrated significant empirical success [13] - [19] by specifying a distribution of uncertainty over various simulation parameters to train robust policies. Combining this domain randomization with online identification of uncertain parameters has been proposed [20, 21]; however, despite these methods' empirical successes, they still rely on real-world environments being well-represented by the distribution of uncertainty used in (relatively extensive) training. By contrast, our approach focuses

on settings where it may be challenging to specify the uncertainty, and we do not require expensive training of policies in simulation to provide theoretical guarantees in the form of bounded regret.

Reachability-based robust planning. Hamilton-Jacobi (HJ) Reachability-based [22, 23] and trajectory library-based [24, 25] robust planning techniques assume *worst-case* realization of uncertainty. As such, they provide adversarial certificates of safety for path planning problems via the construction of representations (or outer approximations) of the safe and unsafe regions of the state space conditional on the robot dynamics, obstacle placement, and disturbance size. HJ methods, which generally suffer from the curse of dimensionality [26] (despite the existence of speed-ups in certain settings [27]), use the formalism of the differential game [28] to provide a "global" notion of safe and unsafe sets [29]. In comparison, robust trajectory libraries, which are usually computed using convex programs [30], provide safety guarantees in the form of robust "tubes" or "funnels" [31, 25] that encompass only the nominal trajectory (or hypothesized trajectories) within the space.

The key distinction of our work from these methods arises from the desire in our setting for "instance-optimality;" namely, the use of online learning and regret minimization allows us to adapt to the *specific* nature of observed disturbances. Our method recovers effective performance in *both* stochastic and non-stochastic (adversarial) regimes; we do not sacrifice too much performance in "benign" environments to provide guarantees on robust performance in more adversarial cases, and vice versa.

Online learning for control. Our work makes significant use of *online learning* [32] to make guarantees on *regret*, which is the difference between the algorithm's performance and that of the *best policy in hindsight* (once disturbances are realized) from a given class of closed-loop policies. Several canonical control-theoretic results have recently been cast as problems in online learning [33, 34], providing interesting generalizations to established control results like the linear-quadratic regulator [35, 36, 37] and H_∞ robust control [38]. Results in optimal sample complexity [39] and synthesis for unknown linear systems [40] illustrate further generalizations of standard control theory.

Standard control formulations are efficiently solvable due to a convex objective. However, "higher-level" decision-making tasks like obstacle avoidance often have non-convex objective functions (e.g., maximizing the distance to the nearest obstacle). Fortunately, some non-convex objectives admit "hidden convexity" — that they can be reformulated (via transformations, relaxations, or conversions to a dual formulation — see [41] for a survey) into equivalent optimization problems that *are convex*. This allows for efficient solutions (e.g., [42, 43, 44]) to problems that nominally would be hard to solve (e.g., [45]). Our work gives such a formulation for the task of obstacle avoidance.

3 Problem Formulation and Preliminaries

Consider a discrete-time dynamical system with state $\bar{\mathbf{x}}$ and control input $\bar{\mathbf{u}}$. A planning oracle $\mathcal{O}_{\mathcal{T}}(\bar{\mathbf{x}}_0)$ takes in an initial state and generates a nominal state trajectory with associated control inputs $\mathcal{T} = \{\bar{\mathbf{x}}_t^0, \bar{\mathbf{u}}_{t-1}^0\}_{t=1}^T$. We design a robust obstacle avoidance controller that will update the trajectory online to avoid local obstacles. Intuitively, this is a faster "safety inner loop" for the slower trajectory planning stage $\mathcal{O}_{\mathcal{T}}$, keeping the agent safe from external features unseen at planning time. For analytical tractability, we assume that the dynamics of perturbations of the nominal trajectory are discrete-time linear[1]. Defining $\mathbf{x}_t = \bar{\mathbf{x}}_t - \bar{\mathbf{x}}_t^0$ and $\mathbf{u}_t = \bar{\mathbf{u}}_t - \bar{\mathbf{u}}_t^0$, this assumption becomes

$$\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t + \mathbf{w}_t, \tag{1}$$

where $\mathbf{w}_t$ is a bounded, unknown, possibly-adversarial disturbance. For many practical systems the linear dynamics assumption is reasonable; one example is a control-affine system with feedback linearization. Additionally, $\mathbf{w}_t$ can encompass small, unmodeled nonlinearities. Our task is to construct this "residual" controller generating $\mathbf{u}_t$ to avoid obstacles. As such, $\mathcal{O}_{\mathcal{T}}$ is the optimistic, goal-oriented planner (in practice, an off-the-shelf algorithm, e.g., [46]) and our controller is the safety mechanism that becomes active *only when needed* and in a provably effective manner.

[1]The results here are presented for linear time-invariant (LTI) systems. We believe that the results extend to linear time-varying (LTV) systems under reasonable but slightly more technical assumptions; however, this formal analysis is left as future work.

3.1 Safety Controller Objective

A controller trying to avoid obstacles needs to maximize the distance to the nearest obstacle, subject to regularization of state deviations and control usage. Assume a sensor mechanism that reports all "relevant" obstacle positions (e.g., within a given radius of the agent). The optimization problem, for a trajectory of length T, safety horizon of length $L \leq T$, and obstacle positions $\mathbf{p}_t^j$ denoting the j^{th} sensed obstacle at time t, is

$$\max_{A \in \mathcal{A}} C_{\text{obs}}(A), \quad \text{where:} \tag{2}$$

$$C_{\text{obs}}(A) := \sum_{t=1}^{T} \min_{\tau \in [L]} \min_{j} \|\mathbf{x}_{t+\tau}^A - \mathbf{p}_t^j\|_2^2 - \|\mathbf{x}_t^A\|_Q^2 - \|\mathbf{u}_t^A\|_R^2.$$

Here, $\mathcal{A}$ is the set of online algorithms that choose actions for the controller, and $\mathbf{x}_t^A$ denotes the realized state trajectory conditioned on actions $\mathbf{u}_t^A \sim A \in \mathcal{A}$. The last two terms represent quadratic state and action costs; these costs[2] are very common objectives in the control literature, but serve primarily here to regularize the solution to the obstacle avoidance task. Note that, though the collision avoidance objective is relaxed, it remains a nonconvex quadratic penalty term rendered additionally complex due to the discrete selection over time and obstacle indices of the minimal-distance obstacle in the first term of C_{obs}. From here, we model the optimal policy search in the online control paradigm [36]. This allows us to define the regret-based safety metric with respect to the best achievable performance in hindsight. For a sufficiently powerful policy comparator class, we achieve meaningful guarantees on the safety of the resulting controller.

3.2 Regret Framework for Obstacle Avoidance

Leveraging the expressiveness of linear dynamic controllers (LDCs) as comparators [38, 36], we use a disturbance-action controller parameterization. The action is the sum of a stabilized nominal action and a residual obstacle-avoiding term

$$\mathbf{u}_t = K\mathbf{x}_t + \mathbf{b}_t + \sum_{i=1}^{H} M_t^{[i]} \mathbf{w}_{t-i}, \tag{3}$$

where $[i]$ indexes the history length and t denotes the time index of the decision. It will be useful to define the intermediate quantities $\tilde{\mathbf{u}}_t = \mathbf{u}_t - K\mathbf{x}_t$ and $\tilde{A} = A + BK$. Then the system dynamics are

$$\mathbf{x}_{t+1} = \tilde{A}\mathbf{x}_t + B\tilde{\mathbf{u}}_t + \mathbf{w}_t. \tag{4}$$

The comparator class Π will be the class of LDCs parameterized by M; this class has provably expressive approximation characteristics [36]. The regret is defined using quantities from Eqn. 2:

$$\text{Reg}_T(A) = \sup_{w_1,\ldots,w_T} \left\{ \max_{M \in \Pi} C_{\text{obs}}(M) - C_{\text{obs}}(A) \right\}. \tag{5}$$

A sublinear bound on Eqn. 5 implies that the adaptive sequence M_t selected by low-regret online algorithm A will perform nearly as well as the *best fixed policy M^* in hindsight, for all realizations of uncertainty* within the system. A sublinear bound on the regret thus establishes finite-time (near) optimality for *all* disturbance realizations: the policy will perform almost as well as a policy that has *a priori* knowledge of the disturbance realizations.

3.3 Trust Region Optimization

To provide guarantees on regret, the method presented in Sec. 4.2 will construct sequential *trust region optimization* instances; we provide a brief overview of this class of problems here. A trust region optimization problem [47] is a nominally non-convex optimization problem that admits "hidden convexity". One can reformulate a trust region instance (see Def. 1) via a convex relaxation in order to solve it efficiently [42, 48].

[2]Here, we omit the fully general LQR costs (time-varying Q_t and R_t) for simplicity of presentation; however, the results we show will *also* hold for this more general setting.

Definition 1 (Trust Region Instance)**.** *A trust region instance is defined by a tuple* $(P, \mathbf{p}, D)$ *with* $P \in \mathbb{R}^{d\times d}$*,* $\mathbf{p} \in \mathbb{R}^d$*, and* $D > 0$ *as the optimization problem*

$$\max_{\|\mathbf{z}\|\leq D} \left\{\mathbf{z}^T P\mathbf{z} + \mathbf{p}^T\mathbf{z}\right\}. \tag{6}$$

Throughout the remainder of this paper, we will use "trust region instance" to refer interchangeably to instances of Def. 1 and the implicit, equivalent convex relaxation.

4 Methodology, Algorithm, and Regret Bound

We provide a brief outline of the regret minimization motivation and methodology for non-convex online learning, from which we outline our algorithm called online learning control (OLC), and discuss its key properties.

4.1 Intuitive Decomposition of the OLC Algorithm Control Signal

The intuition of our control scheme is to allow for online, optimization-based feedback to correct for two key sources of failure in obstacle avoidance: (1) non-robust (risky) nominal plans, and (2) external disturbances. The former can be thought of as errors in planning – that is, they arise *when the nominal path is followed exactly.* Paths that move very close to obstacles or that pass through them (e.g., due to sensor noise) would be examples of this problem; indeed, sensing errors become very relevant on hardware, as shown in Sec. 5.2. The presence of the bias term in the OLC framework accounts for control input to correct these errors. The latter challenge concerns *deviation from nominal trajectories* caused by errors in modeling, time discretization, and physical disturbances. The linear feedback term in the OLC framework accounts for these errors. The OLC algorithm, then, is tasked with addressing these objectives online to reduce instances of collision with obstacles.

4.2 Regret Minimization Methodology

Many existing results in online learning and control are formulated by posing the specific problem as an online convex optimization [49] instance, which can be solved in an efficient manner [50]. This will not suffice for our setting due to the non-convexity of the objective. However, optimal regret bounds for online *non-convex* optimization are given by "Follow the Perturbed Leader" (FPL) methods for problems with an optimization oracle [51, 52]; these bounds have been extended to the setting with memory [38]. The key challenge is to devise a tractable oracle for our setting, and then to combine the results of [52, 38] (subject to some technical modifications to meet the requirements of our application) with the guarantees for online convex optimization.

4.3 Algorithm Exposition and Regret Bound

We formulate the obstacle avoidance controller as an online non-convex FPL algorithm in Alg. 1. At time t, the agent generates a control input $\tilde{\mathbf{u}}_t$ via Eqn. 3 by playing $M_t^{[1:H]}$. The state dynamics are then propagated to reveal the new state $\mathbf{x}_{t+1}$, and the realized $\mathbf{w}_t$ is reconstructed in hindsight from the state. The robot simultaneously uses its sensors to observe local obstacle positions.

The key conceptual step is then the following: given past reconstructed disturbances and obstacle locations, one can construct a reward function (Eqn. 8) at time $t+1$. This is a function of a *counterfactual* set of gains $\tilde{M}_t^{[1:H]}$, where $\mathbf{x}_{t+1}^{\tilde{M}}$ and $\tilde{\mathbf{u}}_t^{\tilde{M}}$ correspond to the state and control input that *would have resulted* from applying $\tilde{M}_t^{[1:H]}$ given the realized disturbances, obstacle locations, and the initial state $\mathbf{x}_0$. The key algorithmic step is to update $M_{t+1}^{[1:H]}$ by solving the optimization problem in Eqn. 9, which is the Follow-the-Perturbed-Leader component of the algorithm (the inner product $(M \bullet P_0)$ is the regularizing perturbation). The resulting sublinear regret bound is summarized in the following theorem; the optimization procedure and the proof of the optimization correctness are deferred to Supp. B, and the proof of the regret bound is deferred to Supp. A.

Theorem 2 (Regret Bound for Online Obstacle Avoidance)**.** *Consider an instance of Alg. 1, with Alg. 2 (see Supp. B) acting as a subroutine optimizing Eqn. 9. Then the regret attained by Memory FPL will be*

$$\tilde{\mathcal{O}}(poly(\mathcal{L})\sqrt{T}), \tag{7}$$

where $\mathcal{L}$ is a measure of the instance complexity.

Algorithm 1 Online Learning Control (OLC) for Obstacle Avoidance

Input: Partially observed obstacle positions $\{\mathbf{p}_t^j\}_{j=1}^K$, planning horizon L, history length H.
Input: Full horizon T, algorithm parameters $\{\eta, \epsilon, \lambda\}$, initial state $\mathbf{x}_0$.
Input: Open-loop plan: $\bar{\mathbf{u}}_t^o$ for $t = 1, ..., T$.
Initialize: Closed-loop correction $M_0^{[1:H]}$, fixed perturbation $P_0 \sim \text{Exp}(\eta)^{d_u \times H d_w}$.
Initialize: Play randomly for $t = \{0, ..., H-1\}$, observe rewards, states, noises, and obstacles.
for $t = H...T-1$ **do**
  Play $M_t^{[1:H]}$, and observe state $\mathbf{x}_{t+1}$ and obstacles $\{\mathbf{p}_{t+1}^j\}_{j\in[k]_{t+1}}$.
  Reconstruct disturbance $\mathbf{w}_t$ using observed $\mathbf{x}_{t+1}$.
  Construct the reward function:

$$\ell_{t+1}(\tilde{M}_t^{[1:H]}) = \min_{j\in[k]} \left\{ \|\mathbf{x}_{t+1}^{\tilde{M}} - \mathbf{p}_{t+1}^j\|_2^2 \right\} - \|\mathbf{x}_{t+1}^{\tilde{M}}\|_Q^2 - \|\tilde{\mathbf{u}}_t^{\tilde{M}}\|_R^2. \tag{8}$$

  Solve for $M_{t+1}^{[1:H]}$ as the solution to:

$$\underset{\|M^{[1:H]}\|\le D_M}{\arg\max} \left\{ \sum_{\tau=1}^{t+1} \ell_\tau(M^{[1:H]}) + \lambda(M^{[1:H]} \bullet P_0) \right\}. \tag{9}$$

end for

5 Experiments

We demonstrate the effectiveness of our method in simulated and physical environments. The simulated environment in Sec. 5.1 considers a 2D racing problem. The hardware experiments in Sec. 5.2 are conducted on a Go1 quadruped robot [53] avoiding obstacles in the presence of unmodeled nonlinear dynamics, sensor noise, and time delays, as shown in Fig. 1. In both settings, we demonstrate the benefits of our approach as compared to baselines that use, resp., RRT*/A* for optimistic planning, and Hamilton-Jacobi (HJ) reachability for robust planning. We illustrate how our method acts as an adaptive intermediary, providing a computationally tractable algorithm that nonetheless maintains improved safety properties relative to purely optimistic planners.

5.1 Simulation Experiments

Experiment overview. In the simulated environment, a racing vehicle (using double integrator dynamics) observes all obstacles within a fixed sensor range (Fig. 2). The nominal dynamics are perturbed by disturbances with varying levels of structure (described further below). A "centerline" environment shown in Fig. 2 is utilized to demonstrate key safety and adaptivity criteria. The nominal trajectory is fixed to be a straight path through obstacles, requiring the obstacle avoidance behavior to emerge via the online adaptation of Alg. 1. For speed, the implementation of the environment and algorithm is set up in JAX [54], using Deluca [55] as a framework for the control-theoretic simulation environment. All individual simulations take 1-2 minutes to run on a single CPU.

Comparison with baselines. We utilize a HJ reachability planner [56] to generate robust trajectories, as well as a kinodynamic RRT* implementation [57] to generate "optimistic" plans. We intend to demonstrate two key behaviors: our algorithm (1) performs (nearly) as effectively with respect to standard LQR costs in states and inputs as RRT* when disturbances are "benign," and (2) performs (nearly) as well as HJ methods when disturbances are adversarial.

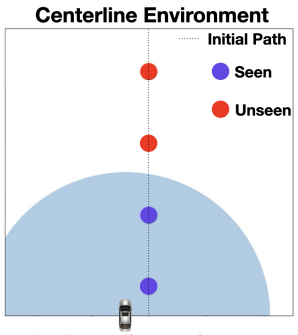

Figure 2: Centerline environment. The nominal path passes vertically through obstacles; the sensor range is denoted by the blue shaded region.

Plan \ Dist.	Rand	Sin	Adv
RRT*	$\mathbf{0.27} \pm 0.05$ 0.60	—- 1.00	—- 1.00
OLC (ours)	0.51 ± 0.09 0.06	$\mathbf{0.49} \pm 0.13$ 0.04	1.57 ± 0.65 0.26
HJ Plan	0.55 ± 0.05 0.00	0.59 ± 0.14 0.00	$\mathbf{1.01} \pm 0.03$ 0.00

Table 1: Planner performance for each disturbance type. Costs are given in terms of linear-quadratic (LQ) costs (top) and fraction of failures (bottom). Best-performing cases for each column are **bold**. LQ costs are only computed for successful passes; as such, RRT* is intentionally blank for two entries.

For each algorithm both stochastic ['Rand'] and non-stochastic (sinusoidal ['Sin'] and adversarial ['Adv']) disturbance profiles are tested, and metrics for both the safety (number of collisions) and performance (LQR state and input costs) are collected for runs spanning 50 centerline obstacles. Results are presented for each algorithm and disturbance profile in Table 1. For space considerations, figures illustrating trajectories of simulated runs referenced below are deferred to Supp. D.

Several aspects of Table 1 reflect the expected behavior of each algorithm. First, RRT* follows efficient paths, but fails to handle disturbances effectively, with a high failure (collision) rate. Second, HJ paths are robust, but performance improvements are limited as 'adversariality' is reduced, and the method fails to take advantage of structure in the sinusoidal disturbances. In particular, the sinusoid case incentivizes the racer to pass the obstacles on a specific side; the HJ planner passes on each side in equal proportion (see Fig. 8) because it does not account for this structure. In contrast, OLC adapts to the structure of the sinusoidal disturbances to take the "easier route" with lower cost (see Fig. 10).

Having illustrated the drawbacks of both RRT** and HJ planners, we summarize the intermediate position OLC occupies, as demonstrated in these experiments. Specifically, as shown in Table 1, OLC significantly reduces collisions across all disturbance profiles relative to RRT*, while also reducing control usage and state costs due to structured disturbances or stationary disturbance profiles relative to HJ planners. This intermediate solution also provides computational speed-up to HJ, allowing it to be run more efficiently online and allow for feedback on sensory information (as opposed to the privileged map information that HJ will require for the hardware experiments in Sec. 5.2). Finally, we ran OLC on several other environments to test its obstacle-avoidance performance; results from these simulations are included in Supp. D.

5.2 Hardware Experiments

Experiment overview. For our hardware experiments, we use the Unitree Go1 quadruped robot, shown in Fig. 1. The robot is equipped with LIDAR and an inertial measurement unit (IMU) which enable obstacle detection and localization using LIO-SAM [58]. The robot's task is to traverse a dense course of cylindrical obstacles, while encountering time delays and residual, nonlinear dynamics. The plant is modeled as a Dubins' car; the equations of motion are included in Supp. C. The high-level inputs are then translated to joint-level torque commands by the robot's low-level controller.

Controller architecture. All sensing and computation is performed onboard the robot. We use a Euclidean clustering algorithm for obstacle detection [59] based on the LIDAR measurements. This detection algorithm runs onboard the robot and provides updated obstacle locations to the controller in real time (note that not all obstacles are initially sensed due to occlusions and larger distances). At each replanning step, the robot generates a nominal set of waypoints from the estimated state and detected obstacles using an A* algorithm [60] over a discretization of the traversable space, which is converted to a continuous path by a down-sampled smoothing spline. We then wrap the additional OLC feedback controller of the form presented in Eqn. 3 to generate corrective actions using Alg. 1.

Baselines. We compare our online learning-based obstacle avoidance controller with the same baselines used for the simulated experiments (Sec. 5.1), substituting A* for RRT* to improve speed and stability of the nominal planner. The first baseline uses A* in a receding horizon re-planning scheme with the nominal dynamics model and obstacle padding of 0.25m to account for the Go1's longitudinal dimensions. The robot executes the control inputs provided by the planner, with periodic replanning at 1 Hz and an action frequency of 4 Hz. The second baseline is a robust controller that is generated using Hamilton-Jacobi (HJ) reachability. This baseline is provided an *a priori* map of the obstacles, and uses a provided model of the disturbances, taking into account velocity-dependent effects. This second baseline acts as a safety 'oracle,' and is not run online; rather, optimal trajectories are synthesized from the true map. As before, we intend to demonstrate that our algorithm improves the safety of a pure A* while providing shorter and more intuitive paths than HJ methods.

Planner \ Metric	Path Length (m)	Max Deviation (m)	Collisions
A*	$\mathbf{10.61} \pm 0.40$	$\mathbf{0.62} \pm 0.33$	12
Online	$\mathbf{10.59} \pm 0.41$	$\mathbf{0.63} \pm 0.28$	7
HJ Plan	11.13 ± 0.45	1.16 ± 0.57	**0** [simulated]

Table 2: Planner performance along several metrics for the hardware examples. We observe that our method ('Online') performs very similarly to A* in terms of path efficiency while reducing collisions by over 40%. Note that HJ methods choose significantly longer paths to account for worst-case uncertainty; HJ was not run on hardware and the optimal paths were calculated offline using the true obstacle positions.

Results. We run physical experiments using A* and our online method for 21 runs each, consisting of three trials over seven obstacle layouts (see the supplemental video and Supp. C for additional details). In each instance, the robot is required to traverse 10m forwards. The optimal HJ path is synthesized *a priori* and analyzed offline (not run on hardware). For all runs, a 'crash' is defined as any contact being made with an obstacle. As shown in Table 2, HJ methods, which require an *overestimation* of the true disturbances to achieve safety, chooses an overly-conservative route that is generally inefficient (but safe). Conversely, A* – even with moderate obstacle padding of 0.25m – does not sufficiently account for the disturbances. As such, on many runs it contacts an obstacle. In contrast to HJ, A* and our online algorithm take significantly shorter paths (note that the shortest possible path is at minimum 10m in length). However, our algorithm reduces the number of collisions by nearly half versus the naive obstacle-padded A* approach, a result significant at $p = 0.1$ using a Boschloo exact test (see Supp. C). Additionally, our algorithm can run online (and onboard) and allow for a feedback mechanism, whereas HJ methods for this problem could not, as each computation of the backward reachable sets took several seconds to complete.

6 Conclusion and Limitations

We develop a regret minimization framework for the problem of online obstacle avoidance. In contrast to prior approaches that either assume worst-case realization of uncertainty or a given stochastic model, we utilize techniques from online learning in order to *adaptively* react to disturbances and obstacles. To this end, we prove regret bounds that demonstrate that our obstacle avoidance policy is comparable to the best policy *in hindsight* from a given class of closed-loop policies. Simulation and hardware experiments demonstrate that our approach compares favorably with baselines in terms of computational efficiency and performance with varying disturbances and obstacle behaviors.

Some limitations that we hope to address in future work include the limited classes of applicable dynamics (though our hardware experiments apply the method to time-varying linear dynamics) and obstacle geometries. Additionally, the cost function does not have multi-time-step lookahead (i.e., MPC-like); more lookahead would yield better foresight in the planning stage and likely improve performance. Finally, the algorithm relies on solutions to optimization problems that run real-time – ensuring stability of the optimization and automating selection of hyperparameters (ours are satisficing but likely suboptimal) would be useful extensions that may further improve the runtime performance.

References

[1] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[2] J. D. Gammell and M. P. Strub. Asymptotically optimal sampling-based motion planning methods. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:295–318, 2021.

[3] Z. Kingston, M. Moll, and L. E. Kavraki. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems*, 1:159–185, 2018.

[4] A. Charnes and W. W. Cooper. Chance-Constrained Programming. *Management Science*, 6(1): 73–79, Oct. 1959. ISSN 0025-1909. doi:10.1287/mnsc.6.1.73. URL https://doi.org/10.1287/mnsc.6.1.73.

[5] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *in Proceedings of the American Control Conference*, 2006.

[6] N. E. Du Toit and J. W. Burdick. Probabilistic Collision Checking With Chance Constraints. *IEEE Transactions on Robotics*, 27(4):809–815, Aug. 2011. ISSN 1941-0468. doi:10.1109/TRO.2011.2116190. Conference Name: IEEE Transactions on Robotics.

[7] L. Janson, E. Schmerling, and M. Pavone. Monte Carlo motion planning for robot trajectory optimization under uncertainty. In *Robotics Research: Volume 2*, pages 343–361. Springer, 2017.

[8] S. Jha, V. Raman, D. Sadigh, and S. A. Seshia. Safe autonomy under perception uncertainty using chance-constrained temporal logic. *Journal of Automated Reasoning*, 60:43–62, 2018.

[9] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, May 1998. ISSN 00043702. doi:10.1016/S0004-3702(98)00023-X. URL https://linkinghub.elsevier.com/retrieve/pii/S000437029800023X.

[10] S. Prentice and N. Roy. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, Nov. 2009. ISSN 0278-3649. doi:10.1177/0278364909341659. URL https://doi.org/10.1177/0278364909341659. Publisher: SAGE Publications Ltd STM.

[11] R. Platt, L. Kaelbling, T. Lozano-Perez, and R. Tedrake. Non-Gaussian belief space planning: Correctness and complexity. In *2012 IEEE International Conference on Robotics and Automation*, pages 4711–4717, May 2012. doi:10.1109/ICRA.2012.6225223. ISSN: 1050-4729.

[12] H. Kurniawati. Partially observable markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:253–277, 2022.

[13] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters. Robot learning from randomized simulations: A review. *arXiv preprint arXiv:2111.00956*, 2021.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[15] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[16] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. *Artificial Intelligence*, 314: 103811, 2023.

[17] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[19] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. *arXiv preprint arXiv:2212.03238*, 2022.

[20] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[21] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022.

[22] I. M. Mitchell and C. J. Tomlin. Overapproximating Reachable Sets by Hamilton-Jacobi Projections. *Journal of Scientific Computing*, 19(1):323–346, Dec. 2003. ISSN 1573-7691. doi:10.1023/A:1025364227563. URL https://doi.org/10.1023/A:1025364227563.

[23] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, Melbourne, Australia, Dec. 2017. IEEE Press. doi:10.1109/CDC.2017.8263977. URL https://doi.org/10.1109/CDC.2017.8263977.

[24] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone. Robust online motion planning via contraction theory and convex optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5883–5890, May 2017. doi:10.1109/ICRA.2017.7989693.

[25] A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017. doi:10.1177/0278364917712421. URL https://doi.org/10.1177/0278364917712421.

[26] J. Darbon and S. Osher. Algorithms for overcoming the curse of dimensionality for certain Hamilton-Jacobi equations arising in control theory and elsewhere. *Research in the Mathematical Sciences*, 3(1):19, Sept. 2016. ISSN 2197-9847. doi:10.1186/s40687-016-0068-7. URL https://doi.org/10.1186/s40687-016-0068-7.

[27] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin. FaSTrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522, Melbourne, Australia, Dec. 2017. IEEE Press. doi:10.1109/CDC.2017.8263867. URL https://doi.org/10.1109/CDC.2017.8263867.

[28] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Dover books on mathematics. Wiley, 1965. ISBN 978-0-471-42860-2. URL https://books.google.com/books?id=gtlQAAAAMAAJ.

[29] I. Mitchell, A. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50 (7):947–957, July 2005. ISSN 1558-2523. doi:10.1109/TAC.2005.851439. Conference Name: IEEE Transactions on Automatic Control.

[30] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, May 2003. ISSN 1436-4646. doi:10.1007/s10107-003-0387-5. URL https://doi.org/10.1007/s10107-003-0387-5.

[31] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential Composition of Dynamically Dexterous Robot Behaviors. *The International Journal of Robotics Research*, 18(6):534–555, June 1999. ISSN 0278-3649. doi:10.1177/02783649922066385. URL https://doi.org/10.1177/02783649922066385. Publisher: SAGE Publications Ltd STM.

[32] E. Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[33] J. Hannan. Approximation to Bayes Risk in Repeated Play. In *Approximation to Bayes Risk in Repeated Play*, pages 97–140. Princeton University Press, 1958. ISBN 978-1-4008-8215-1. doi:10.1515/9781400882151-006. URL https://www.degruyter.com/document/doi/10.1515/9781400882151-006/html?lang=en.

[34] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006. doi:10.1017/CBO9780511546921.

[35] A. Cohen, A. Hasidim, T. Koren, N. Lazic, Y. Mansour, and K. Talwar. Online Linear Quadratic Control. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1029–1038. PMLR, July 2018. URL https://proceedings.mlr.press/v80/cohen18b.html. ISSN: 2640-3498.

[36] N. Agarwal, B. Bullins, E. Hazan, S. Kakade, and K. Singh. Online Control with Adversarial Disturbances. In *Proceedings of the 36th International Conference on Machine Learning*, pages 111–119. PMLR, May 2019. URL https://proceedings.mlr.press/v97/agarwal19c.html. ISSN: 2640-3498.

[37] N. Agarwal, E. Hazan, and K. Singh. Logarithmic Regret for Online Control. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper/2019/hash/78719f11fa2df9917de3110133506521-Abstract.html.

[38] U. Ghai, D. Snyder, A. Majumdar, and E. Hazan. Generating Adversarial Disturbances for Controller Verification. In *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, pages 1192–1204. PMLR, May 2021. URL https://proceedings.mlr.press/v144/ghai21a.html. ISSN: 2640-3498.

[39] S. Dean, H. Mania, N. Matni, B. Recht, and S. Tu. On the Sample Complexity of the Linear Quadratic Regulator. *Foundations of Computational Mathematics*, 20(4):633–679, Aug. 2020. ISSN 1615-3375. URL https://resolver.caltech.edu/CaltechAUTHORS:20190806-130716007. Number: 4 Publisher: Springer.

[40] E. Hazan, S. Kakade, and K. Singh. The Nonstochastic Control Problem. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, pages 408–421. PMLR, Jan. 2020. URL https://proceedings.mlr.press/v117/hazan20a.html. ISSN: 2640-3498.

[41] Y. Xia. A Survey of Hidden Convex Optimization. *Journal of the Operations Research Society of China*, 8:1–28, Jan. 2020. doi:10.1007/s40305-019-00286-5.

[42] A. Ben-Tal and M. Teboulle. Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, 72(1):51–63, Jan. 1996. ISSN 1436-4646. doi:10.1007/BF02592331. URL https://doi.org/10.1007/BF02592331.

[43] H. Konno and T. Kuno. Multiplicative Programming Problems. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*, Nonconvex Optimization and Its Applications, pages 369–405. Springer US, Boston, MA, 1995. ISBN 978-1-4615-2025-2. doi:10.1007/978-1-4615-2025-2_8. URL https://doi.org/10.1007/978-1-4615-2025-2_8.

[44] J. Voss, M. Belkin, and L. Rademacher. The hidden convexity of spectral clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2108–2114, Phoenix, Arizona, Feb. 2016. AAAI Press.

[45] T. Matsui. NP-hardness of linear multiplicative programming and related problems. *Journal of Global Optimization*, 9(2):113–119, Sept. 1996. ISSN 1573-2916. doi:10.1007/BF00121658. URL https://doi.org/10.1007/BF00121658.

[46] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. doi:10.1109/MRA.2012.2205651. https://ompl.kavrakilab.org.

[47] D. C. Sorensen. Newton's Method with a Model Trust-Region Modification, Sept. 1980. URL https://digital.library.unt.edu/ark:/67531/metadc283479/. Number: ANL-80-106 Publisher: Argonne National Laboratory.

[48] E. Hazan and T. Koren. A linear-time algorithm for trust region problems. *Mathematical Programming*, 158(1):363–381, 2016.

[49] E. Hazan. *Introduction to Online Convex Optimization*. Dec. 2021. URL http://arxiv.org/abs/1909.05207. arXiv: 1909.05207.

[50] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi:10.1017/CBO9780511804441.

[51] N. Agarwal, A. Gonen, and E. Hazan. Learning in non-convex games with an optimization oracle. In *Conference on Learning Theory*, pages 18–29. PMLR, 2019.

[52] A. S. Suggala and P. Netrapalli. Online Non-Convex Learning: Following the Perturbed Leader is Optimal. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, pages 845–861. PMLR, Jan. 2020. URL https://proceedings.mlr.press/v117/suggala20a.html. ISSN: 2640-3498.

[53] Unitree. unitreerobotics, 2023. URL https://github.com/unitreerobotics.

[54] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[55] P. Gradu, J. Hallman, D. Suo, A. Yu, N. Agarwal, U. Ghai, K. Singh, C. Zhang, A. Majumdar, and E. Hazan. Deluca–a differentiable control library: Environments, methods, and benchmarking. *Differentiable Computer Vision, Graphics, and Physics in Machine Learning (Neurips 2020 Workshop)*, 2020.

[56] S. ASL. Hj-reachability in jax. https://github.com/StanfordASL/hj_reachability, 2021.

[57] H. Zhou. Path planning. https://github.com/zhm-real/PathPlanning, 2020.

[58] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142. IEEE, 2020.

[59] S. Sun. Lidar obstacle detector, 12 2021. URL https://github.com/SS47816/lidar_obstacle_detector.

[60] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi:10.1109/tssc.1968.300136. URL https://doi.org/10.1109/tssc.1968.300136.

[61] E. Hazan. Approximate Convex Optimization by Online Game Playing. *arXiv:cs/0610119*, Oct. 2006. URL http://arxiv.org/abs/cs/0610119. arXiv: cs/0610119.

[62] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997. ISSN 0890-5401. doi:https://doi.org/10.1006/inco.1996.2612. URL https://www.sciencedirect.com/science/article/pii/S0890540196926127.

A Full Regret Proof

A.0 Outline of Proof

1. Reduce path planner to linear dynamical systems model
2. Demonstrate that a general instance of Alg. 2 is a trust region problem
3. Reduce Eqn. 9 in Alg. 1 to an instance of Alg. 2
4. Justify necessary analogous quantities in our problem to those of the proofs in [38, 36]
5. Apply the results for Nonconvex FPL with Memory from [38] to Alg. 1 to obtain the regret bound

A.1 Reduction of Path Planned case to Standard Controls case

Assume that the planner devises a nominal path (denoted with a $(\cdot)^0$ notation) in coordinates $\mathbf{x}$ and inputs $\mathbf{u}$: so the path $\mathcal{P}$ is fully specified as $\mathcal{P} = \{\bar{\mathbf{x}}_t^0, \bar{\mathbf{u}}_t^0\}_{t=0}^T$. Assume that the path is chosen so that at every $\mathbf{x}$ on or near the path, the following dynamics hold around perturbations of the path:

$$\mathbf{x}_t - \bar{\mathbf{x}}_t^0 = A(\mathbf{x}_{t-1} - \bar{\mathbf{x}}_{t-1}^0) + B(\mathbf{u}_{t-1} - \bar{\mathbf{u}}_{t-1}^0) + D\mathbf{w}_{t-1}. \tag{10}$$

Using this change of coordinates, we can essentially negate the path and study the relevant perturbation dynamics $\delta\mathbf{x}_t := \mathbf{x}_t - \bar{\mathbf{x}}_t^0$ and $\delta\mathbf{u}_t := \mathbf{u}_t - \bar{\mathbf{u}}_t^0$, we recover the desired equation:

$$\delta\mathbf{x}_t = A\delta\mathbf{x}_{t-1} + B\delta\mathbf{u}_{t-1} + D\mathbf{w}_{t-1}. \tag{11}$$

For shorthand, we will define $\mathbf{x} := \delta\mathbf{x}$ and $\mathbf{u} = \delta\mathbf{u}$ to ease exposition, remembering that they represent perturbations from the nominal path. Intuitively, this is a reasonable model for 'quasi-static' systems (e.g., a drone or car or aircraft using path planning for non-aggressive maneuvers).

A.2 Algorithm 2 is a Trust Region Solver

The proof that Alg. 2 is a trust region solver is given in Supp. B.1- B.3.

A.3 Algorithm 2 Solves Equation 9

The proof that Eqn. 9 is a special instance of admissible arguments to Alg. 2 is shown in Supp. B.4.

A.4 Technical Notes

A.4.1 Continuity and Conditioning Parameters

We begin with an analysis of the Lipschitz constant for the approximate cost functions (this will follow a similar path to [38]).

First, note that the diameter of the decision set is $2D_M$ and that the gradient of the quadratic cost above is $\nabla_{\mathbf{m}}\ell_t = (P + P^T)\mathbf{m} + \mathbf{p}$. As such,

$$\begin{aligned} L &:= \max_{\mathbf{m},t} \{\|\nabla_{\mathbf{m}}\ell_t(\mathbf{m})\|_\infty\} \\ &\leq \max_{\mathbf{m},t} \{(\|P\|_1 + \|P\|_\infty)D_M + \|\mathbf{p}\|_\infty\} \\ &\leq 2Hd_wRD + R \end{aligned}$$

We consider as well a bound on the conditioning number of the optimization problem. Because the size of the optimization grows linearly in time, the condition number grows at most linearly as well. Therefore, the run-time of the algorithm is polynomial (neither the condition number nor the dimension grows too rapidly).

Finally, we note bounds on the elements of P and $\mathbf{p}$ in the trust region instance. The bounds on costs, states, inputs, and disturbances together imply that the elements of P_t are bounded by $C_u^2\kappa^2\xi$, and the elements of $\mathbf{p}$ are bounded by $C_u^2\kappa^2\xi\beta$ (this again follows [38]).

A.4.2 Truncated State Approximation

The idea of this proof follows directly from [38]; however, we show the proof in detail because in our case the truncated state history affects the resulting vectors $\mathbf{a}_j$ in the optimization, leading to a different instantiation of the problem.

To give a sense of the added subtlety for obstacle avoidance, observe that in certain scenarios, small perturbations in the observed relative obstacle positions could yield large changes in the optimal policy. For example, imagine that there is one obstacle, located directly on the centerline of the nominal planned motion. Then a small perturbation of the obstacle to the right makes the optimal action "Left," while a small perturbation of the obstacle to the left makes the optimal action "Right."

This phenomenon is not a problem in the regret outline because, while the optimal decision is fragile, the loss incurred of choosing incorrectly is bounded by the quadratic (and therefore, continuous) nature of the cost functions themselves.

For the dynamics and control we have assumed that

$$\begin{aligned} x_{t+1} &= A\mathbf{x}_t + B\mathbf{u}_t + D\mathbf{w}_t \\ \mathbf{u}_t &= K\mathbf{x}_t + \mathbf{b}_t + M_t\tilde{\mathbf{w}}_t \\ &= K\mathbf{x}_t + \sum_{i=1}^{H} M_t^{[i]}\mathbf{w}_{t-i}, \end{aligned} \tag{12}$$

where the bias is included by one-padding the disturbance vector. For simplicity we will omit the explicit bias from ensuing analysis; in all cases it can be understood to be incorporated into the measured disturbance. We can then show (as in [38]) that the state can be expressed as the sum of disturbance-to-state transfer function matrices $\Psi_{t,i}$:

$$\begin{aligned} \mathbf{x}_{t+1}^{\mathcal{A}} &= \tilde{A}^{H+1}\mathbf{x}_{t-H}^{\mathcal{A}} + \sum_{i=0}^{2H} \Psi_{t,i}\mathbf{w}_{t-i}, \text{ where} \\ \tilde{A} &= A + BK \text{ and} \\ \Psi_{t,i} &= \tilde{A}^i D\mathbb{1}[i \le H] + \sum_{j=0}^{H} \tilde{A}^j B M_{t-j}^{[i-j]}\mathbb{1}[i-j \in \{1,...,H\}]. \end{aligned}$$

We define the state estimate and cost as

$$\begin{aligned} \mathbf{y}_{t+1} &:= \sum_{i=0}^{2H} \Psi_{t,i}\mathbf{w}_{t-i} \\ \ell_t(M_{t-H:t}) &= c_t(\mathbf{y}_{t+1}(M_{t-H:t}), \tilde{\mathbf{u}}_t) \end{aligned}$$

where $\tilde{\mathbf{u}}_t = M_t\tilde{\mathbf{w}}_t$ (the residual input on top of the closed-loop controller).

Now, assume that $\|\tilde{A}\| \le 1-\gamma$, that $\|\tilde{A}\|, \|B\|, \|D\|, \|K\| \le \beta$, and that for all t it holds that $\|w_t\| \le C_w$, $\|u_t\| \le C_u$, and $\|Q_t\|, \|R_t\| \le \xi$. Then we can show that the approximation error of the costs is sufficiently small. Let the condition number be defined as $k = \|\tilde{A}\|\|\tilde{A}^{-1}\|$.

A.4.3 Bounding the States Along a Trajectory

Note that $\tilde{\mathbf{u}}_t = M_t\tilde{\mathbf{w}}_t$; this implies that $\|\tilde{\mathbf{u}}_t\| \le HDC_w$. This implies further that $\|B\tilde{\mathbf{u}}_t + D\mathbf{w}_t\| \le 2\beta HDC_w$ by the triangle inequality. Assuming that there exists τ such that

$$\|\mathbf{x}_\tau\|_2 \le \frac{2\beta HDC_w}{\gamma},$$

we have that for every $t > H + \tau + 1$, $\|\mathbf{x}_{t-H-1}^{\mathcal{A}}\|_2 \le \frac{2\beta HDC_w}{\gamma}$. (WLOG, we can assume the initial state $\mathbf{x}_0$ is bounded in this domain - that is, that the assumption is satisfied with $\tau = 0$; the region defined above is the long-term reachable set of the state $\mathbf{x}_t$ driven by bounded disturbances $\mathbf{w}_t$ and (implicitly bounded) residual inputs $\tilde{\mathbf{u}}_t$ [the norm is limited by the stability parameter γ of the closed-loop $\tilde{A}$-matrix]).

A.4.4 Bounding the Change in Costs

Now, we analyze the change in costs

$$|c_t(\mathbf{x}^{\mathcal{A}}_{t+1}, \tilde{\mathbf{u}}_t) - \ell_t(M_{t-H:t})| = |\min_{j\in[p]} \|\mathbf{a}_{j,t} + BM_t\tilde{\mathbf{w}}_t\|_2^2 - \min_{j\in[p]} \|\hat{\mathbf{a}}_{j,t} + BM_t\tilde{\mathbf{w}}_t\|_2^2|$$

Noting the definition of $\hat{\mathbf{a}}_{j,t}$ and of $\mathbf{a}_{j,t}$, we can bound the difference between them as a function of the error in approximation of $\mathbf{x}_t$ (see [38]):

$$\begin{aligned}
&\mathbf{a}_{j,t} := \mathbf{p}_{j,t} - \mathbf{x}_t \\
\implies &\hat{\mathbf{a}}_{j,t} - \mathbf{a}_{j,t} = (\mathbf{p}_{j,t} - \hat{\mathbf{x}}_t) - (\mathbf{p}_{j,t} - \mathbf{x}_t) \\
&= \mathbf{x}_t - \hat{\mathbf{x}}_t \\
\implies &\|\hat{\mathbf{a}}_{j,t} - \mathbf{a}_{j,t}\|_2 = \|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2 \\
&\leq kC_x e^{-\gamma H}
\end{aligned}$$

Now, we argue that the loss incurred due to the noise in $\hat{\mathbf{x}}_t$ is less than simply twice the change in cost due to the error in $\hat{\mathbf{a}}_{j,t}$. Let $\hat{j}^* := \arg\min_{j\in[p]}\{\|\hat{\mathbf{a}}_{j,t} - BM_t\tilde{\mathbf{w}}_t\|_2^2\}$. Let j^* be defined analogously. If $j^* = \hat{j}^*$, then the difference in cost is less than or equal to the extra loss incurred by the error in $\hat{\mathbf{a}}$. If $j^* \neq \hat{j}^*$, then it is possible that the true 'binding obstacle' was biased away, and that the 'guessed' binding obstacle was 'biased towards'; therefore, the cost error is possibly due to deviations up to twice the error in the $\hat{\mathbf{a}}_{j,t}$ vectors. This means that, defining δ_t such that $\|\delta_t\|_2 = 2\|\mathbf{x}_t - \hat{\mathbf{x}}_t\|_2$, we have that the following holds:

$$\Delta = |c_t(\mathbf{x}^{\mathcal{A}}_{t+1}, \tilde{\mathbf{u}}_t) - \ell_t(M_{t-H:t})| = |\min_{j\in[p]} \|\mathbf{a}_{j,t} + BM_t\tilde{\mathbf{w}}_t\|_2^2 - \min_{j\in[p]} \|\hat{\mathbf{a}}_{j,t} + BM_t\tilde{\mathbf{w}}_t\|_2^2|$$

$$\leq \max_{\delta_t:\|\delta_t\|_2\leq 2kC_x e^{-\gamma H}} \left\{\|(\hat{\mathbf{a}}_{j,t} + \delta_t) + BM_t\tilde{\mathbf{w}}_t\|_2^2 - \|\hat{\mathbf{a}}_{j,t} + BM_t\tilde{\mathbf{w}}_t\|_2^2\right\}$$

$$= \delta_t^T\delta_t + 2\delta_t^T\hat{\mathbf{a}}_{j,t} - 2\delta_t^T(BM_t\tilde{\mathbf{w}}_t)$$

$$\leq \|\delta_t\|_2^2 + 2(C_x + \|\delta_t\|_2)\|\delta_t\|_2 + 2\|\delta_t\|_2 C_w\beta D_M$$

$$= 3\|\delta_t\|_2^2 + 2C_x\|\delta_t\|_2 + 2C_w\beta D_M\|\delta_t\|_2$$

$$\leq 5C_x\|\delta_t\|_2 + 2C_w\beta D_M\|\delta_t\|_2$$

$$\leq 5(k^2C_x^2e^{-\gamma H}(1 + \beta D_M C_w)).$$

Letting $H = \lceil\gamma^{-1}\log(5k^2C_x(1+\beta D_M C_w)T)\rceil$, we have that

$$\Delta \leq \frac{C_x}{T}.$$

Remark 3. ***Recursive Definition of*** H ***and*** C_x***:***
Currently, there is a recursive nature to the definition of H *and* C_x*;* $H := \lceil \gamma^{-1} \log(5k^2 C_x(1+\beta D C_w)T) \rceil$ *and* $C_x := \frac{2\beta H D C_w}{\gamma}$*. However, this is not problematic because the definitions will have a solution (that can be found efficiently); namely:*

$$H \geq c_1 \log(c_2 C_x)$$
$$C_x = k_1 H$$

$$\implies H \geq c_1 \log(c_2 k_1 H)$$

And for any $c_1, c_2, k_1 \in \mathbb{R}^+$ *and fixed* $T > 0$*, there exists a positive integer* H *such that the above result holds (e.g., following from the fact that* $\log H = o(H)$*). Further, the resulting* H *will not be too large wrt* T *for sufficiently large* T *(e.g., large enough* T *to overcome the constants).*

A.5 Finalizing the Regret Bound

A.5.1 Apply Nonconvex Memory Follow-the-Perturbed-Leader

This result is from [38], Theorem 13 (Corollary 14 gives an equivalent result to our setting in the asymptotic regret behavior; our optimal choice of η and ϵ differs slightly).

A.5.2 Completing the Bound

Finally, we use Alg. 1 (which acts as an efficient ϵ-oracle) with an approximate trust region implementation of our desired optimization problem (Alg. 2) acting as a subroutine, in order to compose the regret components into a complete bound.

$$\begin{aligned}
\text{Regret}(\mathcal{A}) &:= \max_{M \in \Pi} \sum_{t=H}^{T} c_t(x_t^M, \tilde{u}_t(M)) - \sum_{t=H}^{T} c_t(x_t^{\mathcal{A}}, \tilde{u}_t(\mathcal{A})) \\
&\leq \max_{M \in \Pi} \sum_{t=H}^{T} (f_t(M, M, ..., M) + \frac{C_x}{T}) - \sum_{t=H}^{T} (f_t(M_{t-H:t}) + \frac{C_x}{T}) \\
&= \Big[\max_{M \in \Pi} \sum_{t=H}^{T} f_t(M, M, ..., M) - \sum_{t=H}^{T} f_t(M_{t-H:t}) \Big] + \mathcal{O}(\log T) \\
&\leq \tilde{\mathcal{O}}(\text{poly}(\mathcal{L})\sqrt{T})
\end{aligned} \tag{13}$$

To clarify the steps: the second line incorporates the approximation error from Section A.4.2 (which is logarithmic in T, as noted in the third line) and the final line follows from the Nonconvex Memory FPL result of [38].

B Convex-Concave Game: Algorithm and Correctness

For completeness and easier reference, we include a copy of Alg. 1 below. To improve clarity, references to equations within this section (Supp. B) will use their numbering as given in Supp. B, rather than in the main text. The key technical results of this section are to demonstrate: (1) that Alg. 2 is a trust region instance that can be solved efficiently, and (2) that the optimization procedure in Eqn. 15 is solved correctly and efficiently by the trust region procedure Alg. 2. Given these results, our obstacle avoidance algorithm will be computationally efficient and attain low regret.

Algorithm 1 Online Learning for Obstacle Avoidance

Input: Partially observed obstacle positions $\{\mathbf{p}_t^j\}_{j=1}^K$, planning horizon L, history length H.
Input: Full horizon T, algorithm parameters $\{\eta, \epsilon, \lambda\}$, initial state $\mathbf{x}_0$.
Input: Open-loop plan: $\bar{\mathbf{u}}_t^o$ for $t = 1, ..., T$.
Initialize: Closed-loop correction $M_0^{[1:H]}$, fixed perturbation $P_0 \sim \text{Exp}(\eta)^{d_u \times H d_w}$.
Initialize: Play randomly for $t = \{0, ..., H-1\}$, observe rewards, states, noises, and obstacles.
for $t = H...T-1$ **do**
Play $M_t^{[1:H]}$, and observe state $\mathbf{x}_{t+1}$ and obstacles $\{\mathbf{p}_{t+1}^j\}_{j\in[k]_{t+1}}$.
Reconstruct disturbance $\mathbf{w}_t$ using observed $\mathbf{x}_{t+1}$.
Construct the reward function:

$$\ell_{t+1}(\tilde{M}_t^{[1:H]}) = \min_{j\in[k]} \left\{ \|\mathbf{x}_{t+1}^{\tilde{M}} - \mathbf{p}_{t+1}^j\|_2^2 \right\} - \|\mathbf{x}_{t+1}^{\tilde{M}}\|_Q^2 - \|\tilde{\mathbf{u}}_t^{\tilde{M}}\|_R^2. \tag{14}$$

Solve for $M_{t+1}^{[1:H]}$ as the solution to:

$$\arg\max_{\|M^{[1:H]}\| \le D_M} \left\{ \sum_{\tau=1}^{t+1} \ell_\tau(M^{[1:H]}) + \lambda(M^{[1:H]} \bullet P_0) \right\}. \tag{15}$$

end for

Algorithm 2 (General) Hidden-Convex Formulation for Objective in Eqn. 9

Input: Set of vectors $\{\mathbf{a}_j^{[\tau]}\}_{j=1,\tau=1}^{k,H_t}$, matrix B, vectors $\mathbf{b}, \mathbf{b}_0$, time history $H_t \le t$
Input: Iterations N, learning rate η, approx. error ϵ, perturbation P_0, diameter D_M.
Initialize: Vector $\mathbf{c}_0^{[\tau]} = \frac{1}{k}\mathbb{1}^k$, $\tau = 1, \ldots, H_t$.
for n=0...N **do**
(1) Solve for M_n

$$\begin{aligned} M_n = \arg\max_{\|M\| \le D_M} \Big\{ & \sum_{\tau=1}^{H_t} \sum_{j=1}^{k} \mathbf{c}_n(j)^{[\tau]} \|\mathbf{a}_j^{[\tau]} + BM\mathbf{b}^{[\tau]}\|_2^2 \\ & - \|\mathbf{b}_0^{[\tau]} + BM\mathbf{b}^{[\tau]}\|_Q^2 - \|M\mathbf{b}^{[\tau]}\|_R^2 + \lambda(M \bullet P_0) \Big\}. \end{aligned} \tag{16}$$

(2) Update $\mathbf{c}_{n+1}$

$$\mathbf{c}_{n+1}^{[\tau]} = \prod_{\Delta_k} \left[\mathbf{c}_n^{[\tau]} e^{-\eta \nabla_{\mathbf{c}} \left(\sum_j \mathbf{c}_n^{[\tau]}(j) \|\mathbf{a}_j^{[\tau]} + BM\mathbf{b}^{[\tau]}\|_2^2 \right)} \right], \forall \tau \in \{1, \ldots, H_t\}. \tag{17}$$

end for
return M_N

B.1 Non-convex Memory FPL for Obstacle Avoidance

Intuitively, Alg. 1 operates by updating the gain matrices $M_{t+1}^{[1:H]}$ via counterfactual reasoning: *in hindsight*, given the actual observed disturbances and obstacle locations, what gain matrices *would*

have resulted in good performance (in terms of obstacle avoidance and the state-input penalties)? In Supp. A, we demonstrate that this algorithm results in low regret as formalized in Eqn. 5 of the main text. For reference: Eqn. 9 (main text) corresponds to Eqn. 15 (Supp. B) henceforth; similarly, Eqn. 8 (main text) corresponds to Eqn. 14 (Supp. B).

B.2 Efficient Solution of Eqn. 15 (Part 1): Reduction of Alg. 2 to Trust Region Instance

We now prove that Alg. 2 does indeed solve Eqn. 9. Consider the relaxed optimization problem

$$\max_{M \in \mathcal{M}} \sum_{\tau=1}^{H_t} \sum_{j} \lambda_j^{[\tau]} \|\mathbf{a}_j^{[\tau]} + BM\mathbf{b}^{[\tau]}\|_2^2 \tag{18}$$

We will first describe some useful quantities and (physically-motivated) assumptions. The physical quantities of interest have the following characteristics: $\mathbf{x} \in \mathbb{R}^{d_x}$, $\mathbf{u} \in \mathbb{R}^{d_u}$, and $\mathbf{w} \in \mathbb{R}^{d_w}$.

Assumption 4. *$B \in \mathbb{R}^{d_x \times d_u}$, with $d_u \leq d_x$, and rank $(B) = d_u$. This corresponds to the following physical assumptions: (1) there are no more inputs than states, and (2) there are no 'extraneous' inputs (if there are such inputs, then we can find a minimal realization of B and wlog set extraneous inputs to always be zero). Similarly, if (1) fails, then we can remove extraneous inputs by again setting them equal to zero uniformly (just as in (2)).*

The dimension of several other variables of interest are: $\mathbf{b} \in \mathbb{R}^{Hd_w}$ and for the decision variable M, $M \in \mathbb{R}^{d_u \times Hd_w}$. We want to show that the optimization in Eqn. 18 is equivalent to a convex trust region problem, which is efficiently solvable.

Further, let the quantity $B^T B$ have a singular value decomposition denoted by:

$$B^T B = U^T \Lambda U.$$

We see that this decomposition has an orthogonal $U \in \mathbb{R}^{d_u \times d_u}$ and positive definite Λ because rank$(B) = d_u$ and thus the symmetric $B^T B \in \mathbb{R}^{d_u \times d_u}$ has $B^T B \succ 0$.

Now, consider the problem of Eqn. 18, assuming that $\forall \tau \in \{1, \ldots, H_t\}$, there are fixed $\lambda^{[\tau]} \in \Delta_p, B, \{\mathbf{a}_j^{[\tau]}\}_{j=1}^k, \mathbf{b}^{[\tau]}$. For now, choose a single time element, notated as $[i]$. For simplicity, we will only include this notation at the beginning and end, where we sum the result back together. We rearrange the objective for this case as follows:

$$\begin{aligned}
\text{OBJ}_{\text{partial}} &= \sum_j \lambda_j^{[i]} \|\mathbf{a}_j^{[i]} + BM\mathbf{b}^{[i]}\|_2^2 \\
&= \sum_j \lambda_j (\mathbf{a}_j + BM\mathbf{b})^T (\mathbf{a}_j + BM\mathbf{b}) \\
&= \sum_j \lambda_j (\mathbf{a}_j^T \mathbf{a}_j + 2\mathbf{a}_j^T BM\mathbf{b} + \mathbf{b}^T M^T B^T BM\mathbf{b}) \\
&\equiv \sum_j \lambda_j (2\mathbf{c}_j^T M\mathbf{b} + \mathbf{b}^T M^T U^T \Lambda U M\mathbf{b})
\end{aligned}$$

Here, we are searching for the argmax M^*, so the $\mathbf{a}_j^T \mathbf{a}_j$ is irrelevant. Further, we have defined $\mathbf{c}_j^T = \mathbf{a}_j^T B$. Now, let $M_r := UM$, and decompose $M_r = [\mathbf{m}_1^T; \mathbf{m}_2^T; ...; \mathbf{m}_{d_u}^T]$. The optimization

objective is now:

$$\begin{aligned}
\text{OBJ}_{\text{partial}} &= \sum_j \lambda_j \|\mathbf{a}_j + BM\mathbf{b}\|_2^2 \\
&\equiv \sum_j \lambda_j (2\mathbf{c}_j^T M\mathbf{b} + \mathbf{b}^T M^T U^T \Lambda U M\mathbf{b}) \\
&= \sum_j \lambda_j (2\mathbf{c}_j^T M\mathbf{b} + \mathbf{b}^T M_r^T \Lambda M_r \mathbf{b}) \\
&= [\sum_j \lambda_j (2\mathbf{c}_j^T M\mathbf{b})] + \mathbf{b}^T M_r^T \Lambda M_r \mathbf{b}
\end{aligned}$$

The last simplification follows from the fact that $\sum_j(\lambda_j) = 1$ (because $\lambda \in \Delta_{d_u}$). Now, using our knowledge of the diagonal nature of Λ and the column partition of M_r, we can see that

$$M_r^T \Lambda M_r = \sum_{j=1}^{d_u} \sigma_j^2 \mathbf{m}_j \mathbf{m}_j^T$$

Substituting into the OPT formulation, we can further simplify all the way to the desired form:

$$\begin{aligned}
\text{OBJ}_{\text{partial}} &= \sum_i \lambda_i \|\mathbf{a}_i + BM\mathbf{b}\|_2^2 \\
&\equiv \Big[\sum_i \lambda_i (2\mathbf{c}_i^T M\mathbf{b})\Big] + \mathbf{b}^T M_r^T \Lambda M_r \mathbf{b} \\
&= \Big[\sum_i \lambda_i (2\mathbf{c}_i^T M\mathbf{b})\Big] + \mathbf{b}^T (\sum_{j=1}^{d_u} \sigma_j^2 \mathbf{m}_j \mathbf{m}_j^T)\mathbf{b} \\
&= \Big[\sum_i \lambda_i (2\mathbf{c}_i^T U^T M_r\mathbf{b})\Big] + (\sum_{j=1}^{d_u} \sigma_j^2 \mathbf{b}^T \mathbf{m}_j \mathbf{m}_j^T \mathbf{b}) \\
&= 2\Big[\sum_i \lambda_i (\sum_j \tilde{c}_{i,j} \mathbf{m}_j^T \mathbf{b})\Big] + (\sum_{j=1}^{d_u} \sigma_j^2 \mathbf{m}_j^T \mathbf{b}\mathbf{b}^T \mathbf{m}_j) \\
&= \sum_j \Big[(2\sum_i \lambda_i \tilde{c}_{i,j})\mathbf{b}^T\Big]\mathbf{m}_j) + (\sum_{j=1}^{d_u} \mathbf{m}_j^T (\sigma_j^2 \mathbf{b}\mathbf{b}^T)\mathbf{m}_j) \\
&= \mathbf{m}^T P\mathbf{m} + \mathbf{p}^T \mathbf{m}.
\end{aligned}$$

where $\mathbf{m}$ is a vector concatenation of the transposed rows of M_r. Now, to combine the results over time, utilizing the convexity-preserving property of function addition, we simply reintroduce the $[i]$-indexing and sum:

$$\begin{aligned}
\text{OBJ}_{\text{partial}} &= \sum_j \lambda_j^{[i]} \|\mathbf{a}_j^{[i]} + BM\mathbf{b}^{[i]}\|_2^2 \\
&= \mathbf{m}^T P^{[i]}\mathbf{m} + (\mathbf{p}^{[i]})^T \mathbf{m} \\
\implies \text{OBJ}_{\text{full}} &= \sum_i \sum_j \lambda_j^{[i]} \|\mathbf{a}_j^{[i]} + BM\mathbf{b}^{[i]}\|_2^2 \\
&= \sum_i \mathbf{m}^T P^{[i]}\mathbf{m} + (\mathbf{p}^{[i]})^T \mathbf{m} \\
&= \sum_i \mathbf{m}^T P\mathbf{m} + \mathbf{p}^T \mathbf{m}
\end{aligned}$$

Once we solve for $\mathbf{m}^*$ using a trust region solver, we unpack it into M_r^*, and get $M^* = U^T M_r^* (= U^T(UM^*) = M^*)$ as desired. Further, we can translate a norm bound on M into an equivalent one on $\mathbf{m}$ (at least, for appropriate choice of norm bound - e.g., the Frobenius norm on M becomes the 2-norm on $\mathbf{m}$).

B.3 Solving the FPL Sub-Problem

In order to feasibly engage the obstacle avoidance algorithm, it is necessary to solve for optimal solutions to the objective in Eqn. 15, which is an instance of the following max-min problem:

$$\arg\max_{\|M\| \leq D_M} \sum_{\tau=1}^{t+1} \min_j \|\mathbf{a}_j^{[\tau]} + BM\mathbf{b}^{[\tau]}\|_2^2 - \|\mathbf{b}_0^{[\tau]} + BM\mathbf{b}^{[\tau]}\|_Q^2 - \dots$$
$$\dots - \|M\mathbf{b}^{[\tau]}\|_R^2 + \lambda(M \bullet P_0),$$

where $\mathbf{a}_j^{[\tau]}, B, \mathbf{b}_0^{[\tau]}, \mathbf{b}^{[\tau]}$ depend on the dynamics (Eqn. 4, main text) and the obstacle locations. The resulting algorithm is shown in Alg. 2, which converges to $M_{t+1}^{[1:H]}$ in Eqn. 15 of Alg. 1.

Finally, we need to demonstrate that Alg. 2 will converge to a pair $\{\mathbf{c}_N, M_N\}$ that corresponds to the optimal solution of Eqn. 15. This follows immediately from Theorem 7, Part II of [61]. We have an instance of a repeated game in which an optimization oracle efficiently solves Eqn. 16, and then the low-regret exponentiated gradient algorithm [62] iteratively updates $\mathbf{c}_n$ (Eqn. 17). Again, we utilize the fact that the operation of function addition preserves, respectively, the convexity and concavity properties of pairs of operand functions as a necessary tool to allow for the results to still hold when applying the summation over time steps.

B.4 Efficient Solution of Eqn. 15 (Part 2): Reduction of Obstacle Avoidance to Alg. 2

This proof works as a reduction, where we show that the obstacle avoidance problem (Eqn. 15) constitutes a particular set of inputs to the general formulation of Alg. 2. Recall that at time t, the algorithm A has access to the state trajectory $\{\mathbf{x}_\tau^A\}_{\tau=1}^t$, the disturbance history $\{\mathbf{w}_\tau\}_{\tau=1}^{t-1}$, and the sets of sensed obstacles $\{\mathbf{p}_\tau^j\}_{\tau=1}^t$. For any $\tau \in \{H, ..., t\}$, the loss function can be written as an instance of Alg. 2. Specifically, let $\mathbf{a}_\tau^j = \tilde{A}\mathbf{x}_{\tau-1} + D\mathbf{w}_{\tau-1} - \mathbf{p}_\tau^j$, $\mathbf{b}_\tau = \mathbf{w}_{\tau-H:\tau-1}$, $\mathbf{b}_{0,\tau} = \tilde{A}\mathbf{x}_{\tau-1} + D\mathbf{w}_{\tau-1}$. Then, for an appropriate $\mathbf{c}_\tau \in \Delta_{k_\tau}$, the optimization problems are equivalent. Concatenating over the τ-formulations, we have an instance of Eqn. 16. Specifically, for some choice of $\mathbf{c}$, we will have an equivalent problem as Eqn. 15; here $\mathbf{c}$ is an encoding – unknown *a priori* – of the relevant (nearest) obstacles at each time step.

C Hardware Experiment Details

C.1 Equations of Motion

The equations of motion for the high-level Go1 control are:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \psi_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \psi_t \end{bmatrix} + dt \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_\psi \end{bmatrix} + \begin{bmatrix} w_{x,t} \\ w_{y,t} \\ w_{\psi,t} \end{bmatrix}, \tag{19}$$

where u_x is the commanded forward velocity and u_ψ is the commanded yaw rate. The disturbances w_x, w_y and w_ψ capture unmodeled disturbances including imperfect velocity tracking by the robot's low-level controller and deviations due to localization noise.

C.2 Boschloo Test for Significance

In order to evaluate the improvement of OLC vs A* in our hardware experiments, we perform a Boschloo Exact Test on the outcomes of the experiment. This test (for our purposes) essentially provides a statistical evaluation of the difference in means of two Bernoulli variables in the small-data regime. In our experiments, we have a 2x2 test matrix in which OLC and A* are each run 21 times, with the number of failures a random variable depending on each algorithm's behavior subject to the realized obstacle layouts. While we do not claim that our distribution of layouts exactly matches the 'true distribution in the world,' we believe it to be similar enough such that the statistic should be meaningful here.

To be precise, the statistic encapsulates the probability, in the null hypothesis that the two means are equal (or that one mean is greater), of achieving a small sample of realizations that is *more extreme* than that observed in the true data. By choosing the alternative hypothesis "collision rate of A* is higher," a significant result (say, at $\alpha = 10\%$ or $\alpha = 5\%$ significance) would mean that we reject the null hypothesis. This, then, would provide evidence that the use of OLC meaningfully reduced the collision rate.

Using the existing scipy implementation (scipy.stats.boschloo_exact), we find that for the data presented in Table 2, the test statistic is 0.1073 with $p = 0.074$, indicating significance at $\alpha = 0.1$ but not $\alpha = 0.05$. Though not conclusive, this provides reasonable evidence that the reported improvement in the collision rate is not due to random chance in the obstacle layout realizations.

C.3 Obstacle Layouts

We provide a bird's eye view of the layout configurations used during the experiments in Figure 3. Additionally see the supplementary video for the physical instantiation of the layouts used in the experiments.

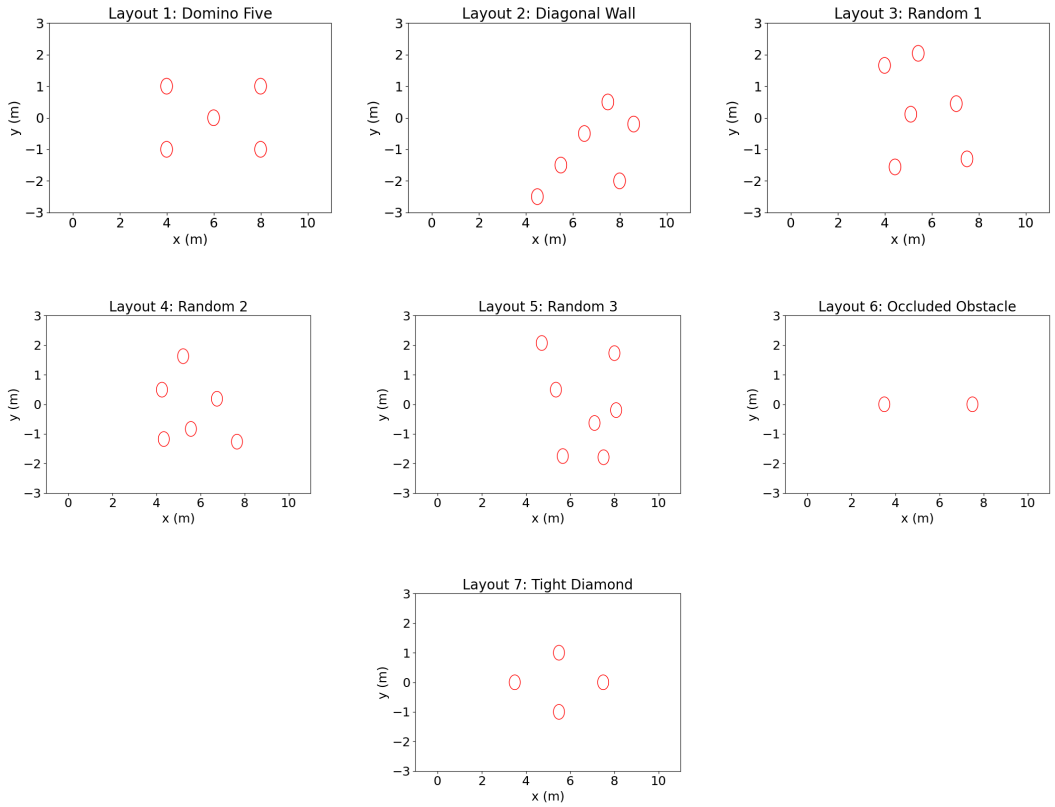

Figure 3: Obstacle layouts from a bird's eye view used during the experiments. Obstacles are denoted by the red circles. The quadruped was placed at (0, 0) and tasked with traversing 10m in the the x direction.

D Simulation Details and Resources

D.1 Simulation Implementation: Parameters, Setup, and Runtime

In this section, we report the hyperparameters used for the experiments results in the main text. We implemented our algorithm and environments in JAX. All experiments were carried out on a single CPU in minutes.

We set the full horizon T to 100 and the history length H to 10. For random perturbation across environments, we sample noise from Gaussian distribution with mean 0 and standard deviation 0.5. For directional perturbation, we sample Gaussian noise with mean 0.5 and standard deviation 0.5. A random seed of 0 is used for all experiments. We obtain the nominal control from LQR with Q set to 0.001 and R set to 1. We then learn the residual obstacle-avoiding parameter M via gradient descent. The learning rate of gradient descent is 0.008 in the centerline environment and is 0.001 for the other environments.

An important note for these experiments: we do not implement existing heuristic techniques like obstacle padding to improve the RRT* collision-avoidance performance. As such, this performance is not meant to suggest that RRT* *cannot* work robustly in these settings, only that its nominal (and theoretically grounded) form does not account for disturbances or uncertainty and is therefore "optimistic" as compared to HJ methods, etc.

D.2 Additional Figures and Trajectories – Centerline Environment

This appendix includes sample trajectories and other relevant visualizations for each algorithm.

D.2.1 RRT* / A*

Here, we demonstrate some sample paths for RRT* in each disturbance regime. Fig. 4 shows uniform random noise, Fig. 5 shows sinusoidal noise, and Fig. 6 shows adversarial noise. In each case, the shift at the final time (goal position, top of image) causing a horizontal shift in the path should be ignored.

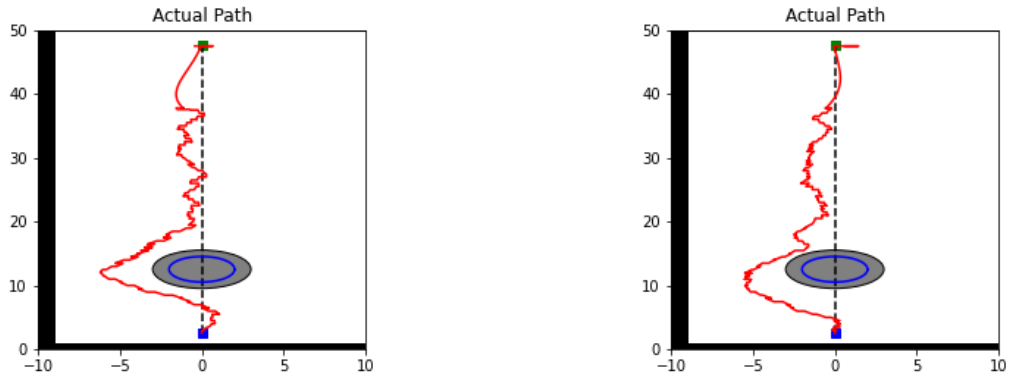

Figure 4: RRT* Planner trajectories against uniform random disturbances. Obstacle is the gray sphere, with the nominal trajectory a dashed black (vertical) line.

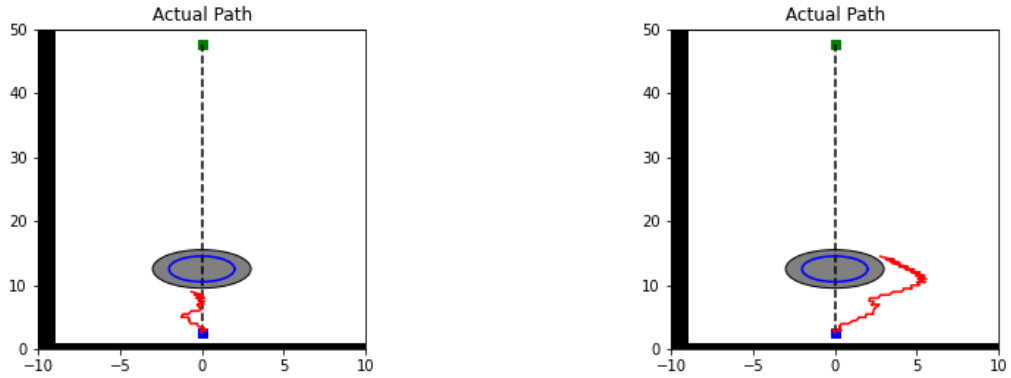

Figure 5: RRT* Planner trajectories against sinusoidal disturbances. Obstacle is the gray sphere, with the nominal trajectory a dashed black (vertical) line.

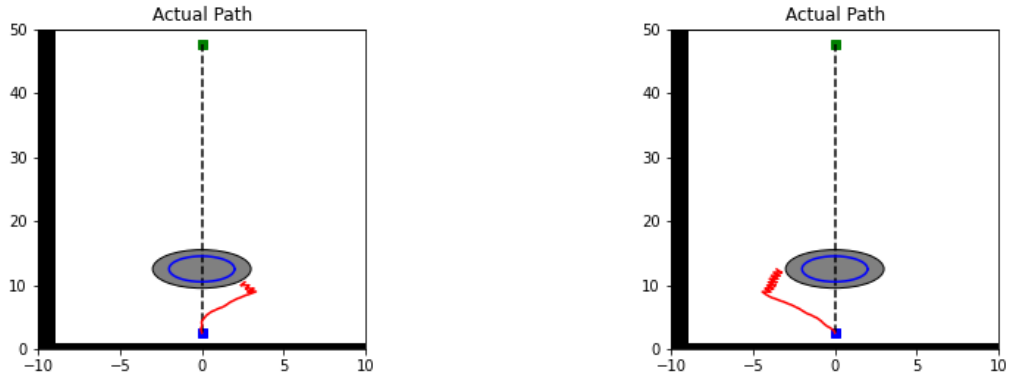

Figure 6: RRT* Planner trajectories against adversarial disturbances. Obstacle is the gray sphere, with the nominal trajectory a dashed black (vertical) line.

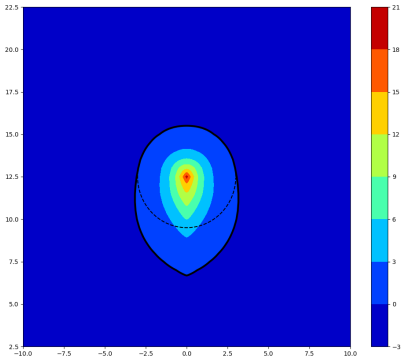

Figure 7: Racer backwards reachable set (inside thick black line) and the obstacle (dashed black line).

D.2.2 HJ Reachability Planner

For the centerline example, the HJ Reachability planner constructs in Fig. 7 the backwards-reachable set for a given obstacle (dashed line), subject to the dynamics constraints imposed on the racer.

Note that every positive-value region denotes an unsafe region. The interpretation is that there is a "pseudo-cone" in front of the obstacle from which the vehicle cannot escape hitting the obstacle *if the disturbances are sufficiently adversarial*. Note that this means that HJ planning is independent of the *actual* disturbances. For each of the disturbance patterns (random, sinusoid, adversarial), we plot a collapsed view of sample trajectories around an obstacle for the HJ planner in Fig. 8. Note how similar each plot is, due to this independence of the control from the actual observed disturbances.

D.2.3 Online Learned Planner

The key illustration here is that the trajectories of the online planner follow the structure of the disturbances, as illustrated by the following comparison of the uniform random and sinusoidal disturbances in Fig. 9 and Fig. 10.

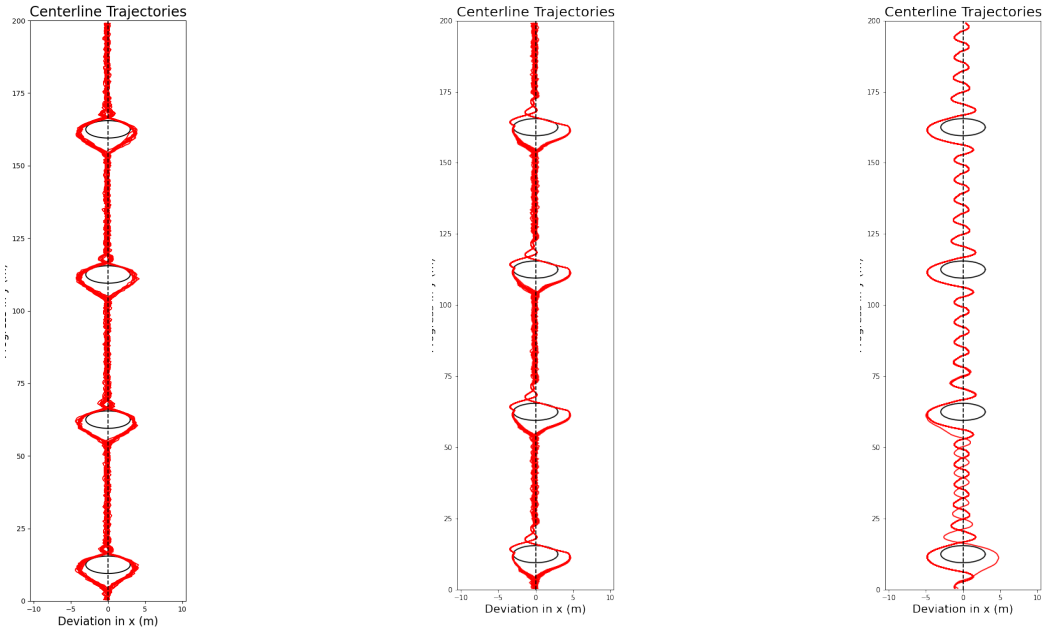

Figure 8: HJ Planner trajectories against (L) uniform random, (C) sinusoid, and (R) adversarial disturbances. Obstacles are black spheres, with the nominal trajectory a dashed black (vertical) line.

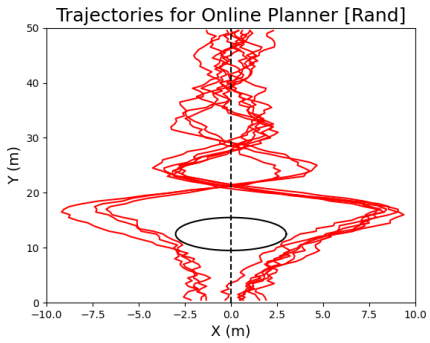

Figure 9: Collapsed trajectories of the racer using the online planner with random disturbances. The racer passes on each side evenly.

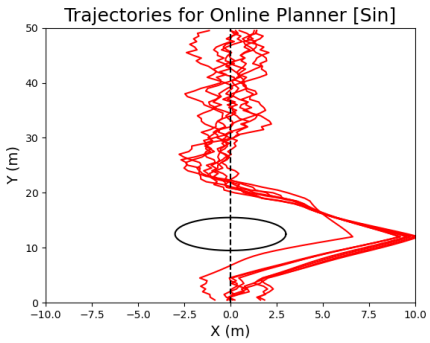

Figure 10: Collapsed trajectories of the racer using the online planner with sin disturbances. The racer learns to pass on the right.

D.3 Failures in the Slalom Setting

We include an image of the four major environments in Fig. 11. Our simulated performance was strong in all environments except for the slalom environment, which we discuss further below.

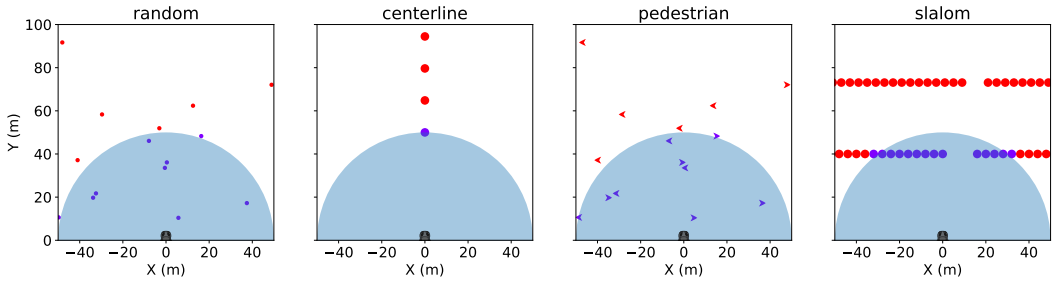

Figure 11: Illustration of the four environments used as a proof-of-concept for our Online algorithm.

The "slalom" setting allows us to tune its difficulty by varying the x-position (i.e. offset) of the gates or by narrowing their width. Fig. 12 illustrates the effect of increasing gate offset from center (i.e., error in the nominal planned trajectory – x-axis) and decreasing gate width (i.e., greater sensitivity to disturbances – y-axis) on failure rate through the slalom gates. As expected, reduced gate width and increased offset broadly increase failure rates. This is due to the online planner being forced to overcome a poor nominal planned trajectory; in combination with the gated passageways, this requires very precise sequences of inputs and a longer memory of previously observed gates (due to the limited sensing horizon). This is discussed further in Supp. D.3.1.

D.3.1 Discussion

The first answer is relatively direct: in all of our examples, we are implicitly acting in a kind of Frenet frame, where all obstacle positions and other referencing is to the ego vehicle (racer) position. As such, the nominal planned trajectory can always be thought of as mapped to a straight line ahead of the racer. In this context, some slalom gates represent a 20m deviation *from the nominal trajectory.* However, this flies in the face of the central modeling intuition of the online framework – that obstacle avoidance is local, with local sensing, local deviations from the nominal trajectory, and "reactive" control to disturbances as they arise. In this vein, the nominal slalom is a challenging task, precisely because it stretches the limits of what can be met by our setup. Concretely: limited sensing makes each slalom wall a kind of "gradient-less" observation (shifting left and right yields only a continuation of the wall *unless the gap is already sensed*), meaning that choosing the correct Left/Right action is difficult. Additionally, the map displays memory, because going the wrong way early through one gate can render the next gate infeasible.

It is in light of these considerations that we argue that the slalom case is actually a case for our model, because it interpretably creates a setting in which the key assumptions are broken. Just like an actual skier who overshoots through one gate and cannot recover for the next gate, so too does our obstacle avoidance algorithm run the risk of "dooming" itself due to a wrong turn – but this is, as described, fundamental to the hardness of the obstacle avoidance problem! As such, we consider the slalom gate as a fundamentally hard problem, and consider a case for future work a fuller characterization of how our planner works for slaloms of varying difficulty, as measured by the sensor range, the distance between gates (both laterally and longitudinally), and the fundamental "cost memory" as it depends on these and other parameters.

D.3.2 Experimental Parameter Sweep – Slalom Course

In an effort to better represent the effects of the slalom setting and the dependence on gate width and offset, Fig. 12 illustrates the effect of increasing gate offset from center (i.e., error in the nominal planner trajectory – x-axis) and decreasing gate width (i.e., greater sensitivity to disturbances – y-axis)

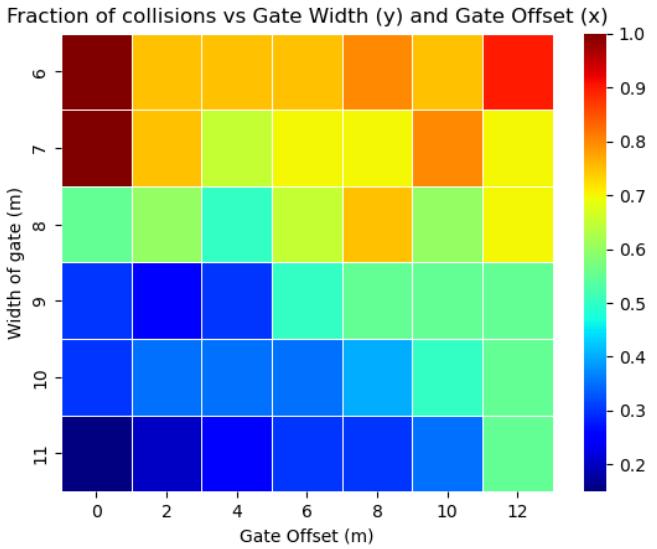

Figure 12: Failure rate heatmap for increasing gate offset (left to right) and decreasing gate width (bottom to top). As expected, failure increases with narrower gates (top) and larger offsets (right).

on failure rate through the slalom gates. As expected, reduced gate width and increased offset broadly increase failure rates.

We note that for narrow gates, a zero-offset slalom is actually quite challenging to ensure - we believe this is due to the fundamental H_∞ limit of stabilization of disturbances for this system; namely, a too-narrow gate requires too-strong robustness about the setpoint (origin), causing failure. This also explains why failure rates are high but not one for moderate offsets in the narrow-gate environment as well: specifically, they allow some freedom away from the zero-offset regularization problem. Besides this, the trend quite clearly demonstrates the fundamental increases in difficulty observed for narrower gates and larger offsets, as expected.