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.

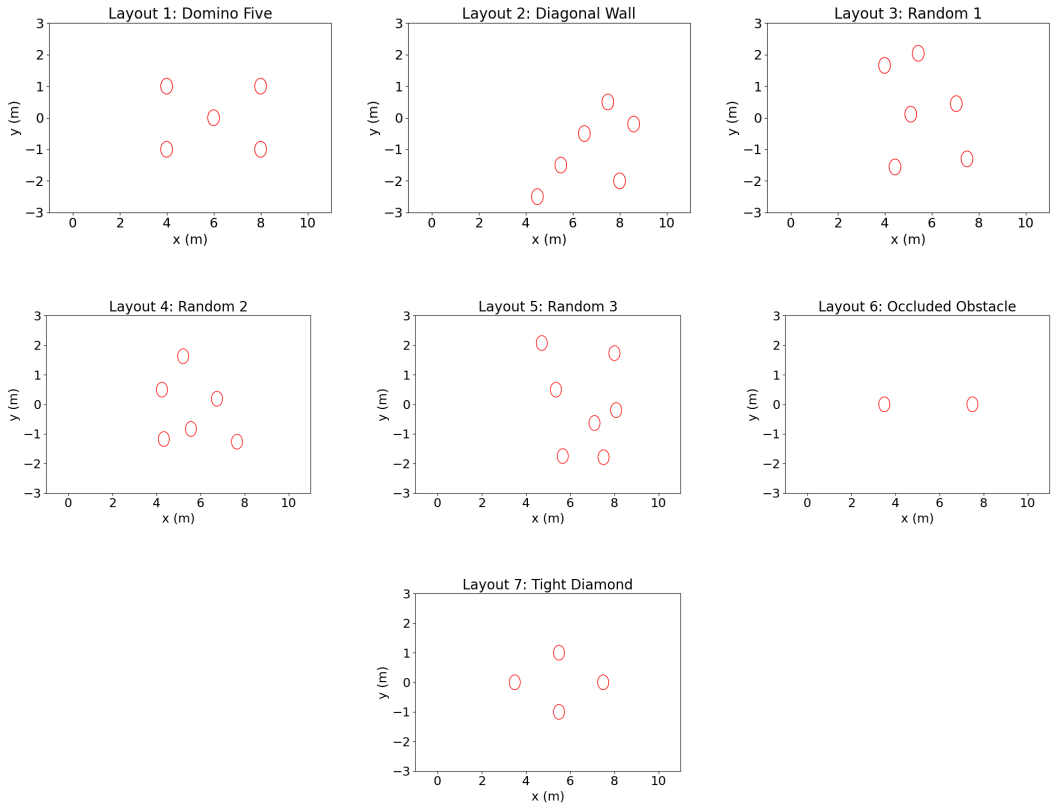

Figure 3: Obstacle layouts from a bird's eye view used during the experiments. Obstacles are denoted by the red circles. The quadruped was placed at (0, 0) and tasked with traversing 10m in the the x direction.

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.

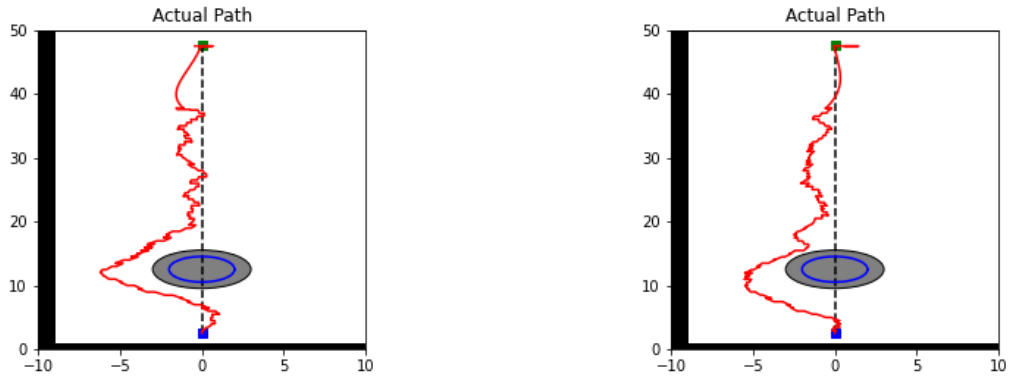

Figure 4: RRT* Planner trajectories against uniform random disturbances. Obstacle is the gray sphere, with the nominal trajectory a dashed black (vertical) line.

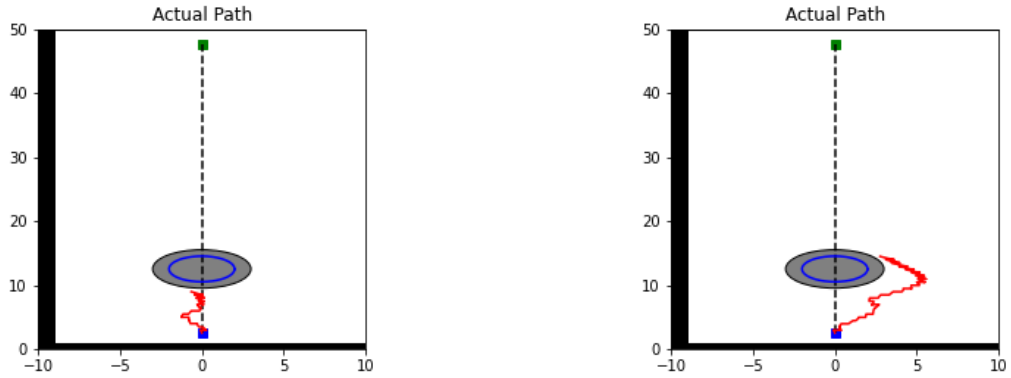

Figure 5: RRT* Planner trajectories against sinusoidal disturbances. Obstacle is the gray sphere, with the nominal trajectory a dashed black (vertical) line.

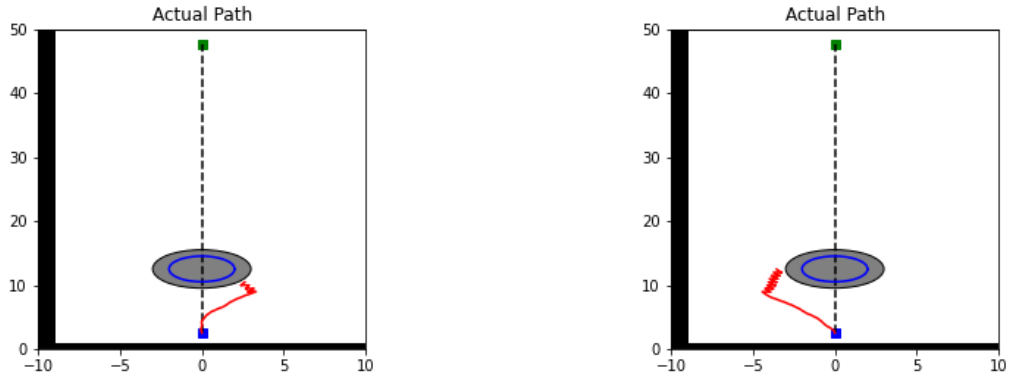

Figure 6: RRT* Planner trajectories against adversarial disturbances. Obstacle is the gray sphere, with the nominal trajectory a dashed black (vertical) line.

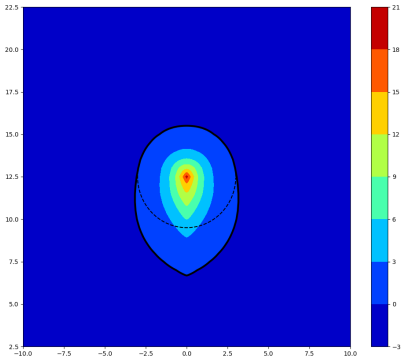

Figure 7: Racer backwards reachable set (inside thick black line) and the obstacle (dashed black line).

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.

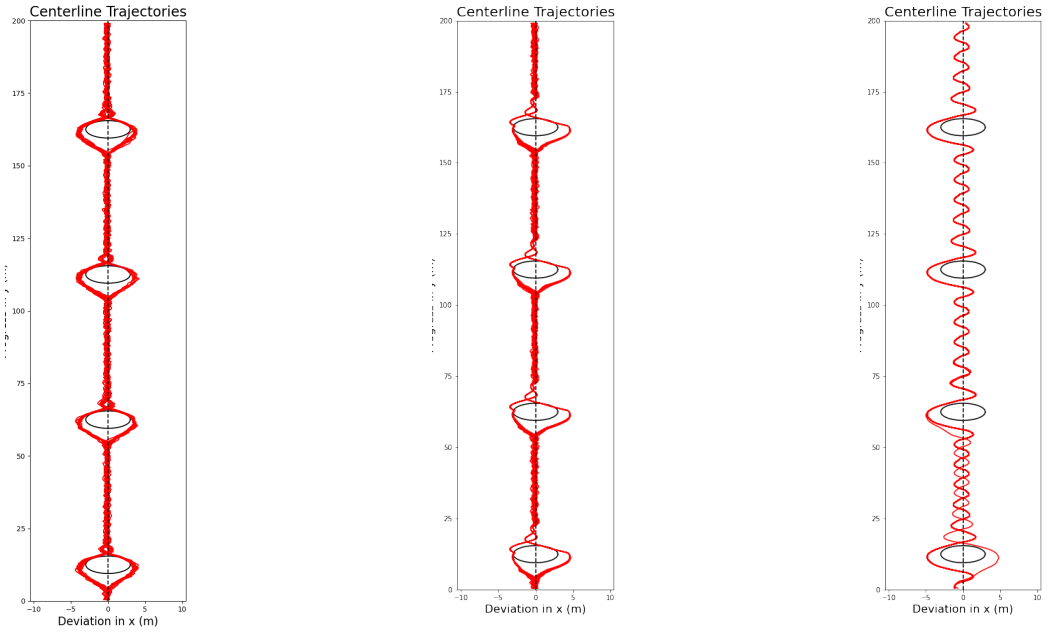

Figure 8: HJ Planner trajectories against (L) uniform random, (C) sinusoid, and (R) adversarial disturbances. Obstacles are black spheres, with the nominal trajectory a dashed black (vertical) line.

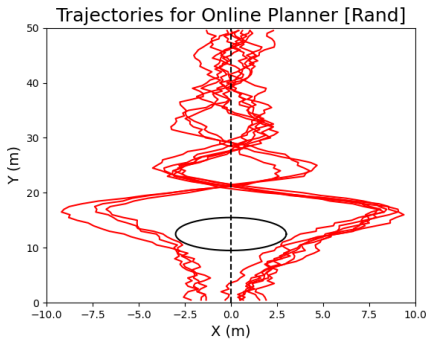

Figure 9: Collapsed trajectories of the racer using the online planner with random disturbances. The racer passes on each side evenly.

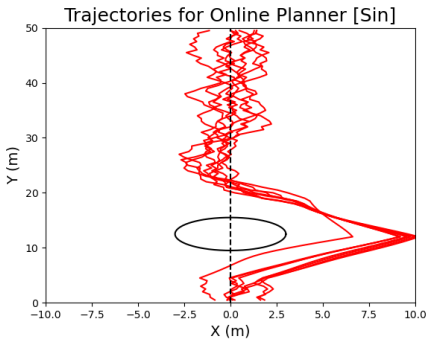

Figure 10: Collapsed trajectories of the racer using the online planner with sin disturbances. The racer learns to pass on the right.

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.

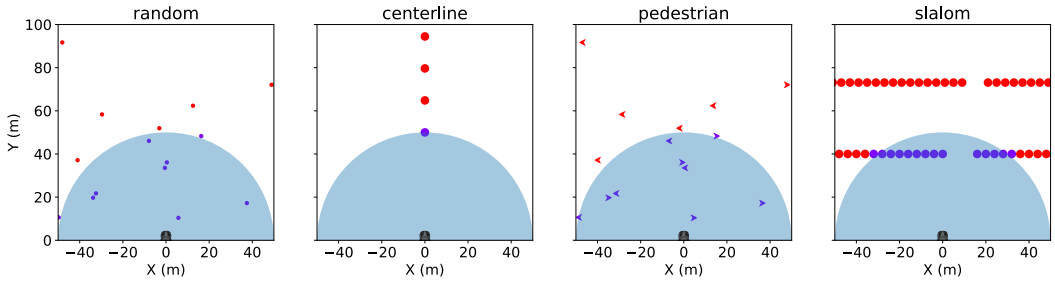

Figure 11: Illustration of the four environments used as a proof-of-concept for our Online algorithm.

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)

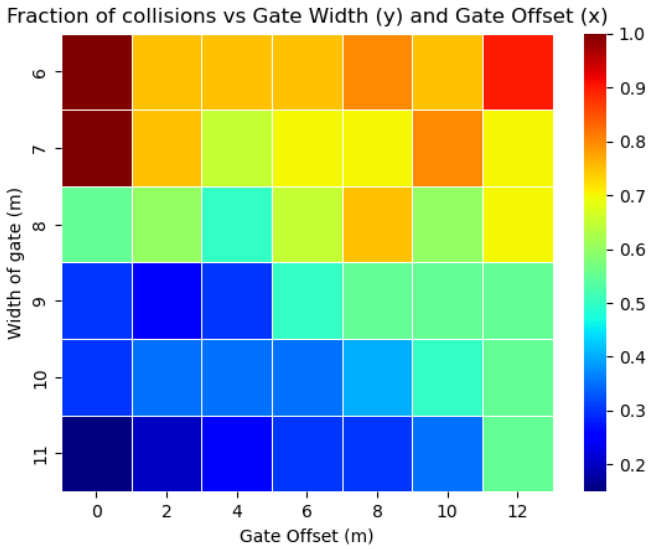

Figure 12: Failure rate heatmap for increasing gate offset (left to right) and decreasing gate width (bottom to top). As expected, failure increases with narrower gates (top) and larger offsets (right).

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.