APPENDIX I
COST REDUCTION DERIVATIONS

We provide more involved derivations of all of the cost function reductions presented in Section III of the paper. In the following, the density function for the uniform distribution for set A is defined as

$$\mathcal{U}_A(x) = \begin{cases} u_A & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $u_A = \frac{1}{\texttt{vol}(A)}$, where $\texttt{vol}(A)$ defines the volume of the set. A Dirac-delta distribution $\delta_a(x)$ for state $a \in \mathcal{X}$ is a special case of a uniform distribution where the support is limited to a single state $A = \{a\}$. In this case $\delta_a(x) = 1$ if $x = a$ and 0 otherwise.

A. Goal Set Indicator

We set the terminal state uncertainty distribution when following trajectory τ to be the Dirac-delta distribution (i.e. known state/deterministic dynamics), $\delta_{x_T|\tau}(x)$, where $x_T \in \mathcal{X}$ is the state reached after following trajectory τ. We set the goal distribution to be a uniform distribution $U_{\mathcal{G}}(x)$ over the goal set $\mathcal{G}$. Minimizing the I-projection of the KL divergence between these distributions we recover

$$\begin{aligned} \operatorname*{argmin}_{\tau} D_{\text{KL}}(\delta_{x_T|\tau}(x) \parallel \mathcal{U}_{\mathcal{G}}(x)) &= \operatorname*{argmin}_{\tau} \int_{x \in \mathcal{X}} \delta_{x_T|\tau}(x) \log \frac{\delta_{x_T|\tau}(x)}{\mathcal{U}_{\mathcal{G}}(x)} dx && (11) \\ &= \operatorname*{argmin}_{\tau} \int_{x \in \{x_T\}} \log \frac{1}{\mathcal{U}_{\mathcal{G}}(x)} dx && (12) \\ &= \operatorname*{argmin}_{\tau} \begin{cases} -\log u_{\mathcal{G}} & \text{if } x_T \in \mathcal{G} \\ \infty & \text{otherwise} \end{cases} && (13) \end{aligned}$$

where Equation 12 follows from the fact that $\delta_{x_T|\tau}(x) = 1$ for $x \in \{x_T\}$ and $\delta_{x_T|\tau}(x) = 0$ for $x \in \mathcal{X} \setminus \{x_T\}$. Hence the minimum is obtained with a constant cost if the terminal state x_T of trajectory τ reaches any point in the goal set $\mathcal{G}$ while any state outside of $\mathcal{G}$ receives infinite cost. This function is non-differentiable as expected from the set-based goal definition. We can treat a single goal state naturally as a special case of this function. While the KL divergence provides these specific costs, this cost function is equivalent to any classifier or indicator that provides distinct costs for inside the goal set versus outside (e.g. $\{0, 1\}$ or $\{-1, +1\}$). Another common equivalent formulation provides a cost of 1 (true) inside the goal set as and outside the set as 0 (false) and define a maximization.

B. (Weighted) Euclidean Distance:

Consider a Gaussian goal distribution $\mathcal{N}\left(x \mid g, \Lambda_g^{-1}\right)$ with mean centered at a desired goal state g, and assume a Dirac-delta distribution over the current state (i.e. known state/deterministic dynamics), $\delta_{x_T|\tau}(x)$, where again x_T is the resulting state from following trajectory τ. Minimizing the I-projection recovers a weighted squared Euclidean distance:

$$\begin{aligned} \operatorname*{argmin}_{\tau} D_{\text{KL}}\left(\delta_{x_T|\tau}(x) \parallel \mathcal{N}\left(x|g, \Lambda_g^{-1}\right)\right) &= \operatorname*{argmin}_{\tau} \int_{x \in \mathcal{X}} \delta_{x_T|\tau}(x) \log \frac{\delta_{x_T|\tau}(x)}{\mathcal{N}\left(x \mid g, \Lambda_g^{-1}\right)} dx && (14) \\ &= \operatorname*{argmin}_{\tau} \int_{x \in \{x_T\}} \log \frac{1}{\mathcal{N}\left(x \mid g, \Lambda_g^{-1}\right)} dx && (15) \\ &= \operatorname*{argmin}_{\tau} -\log \mathcal{N}\left(x = x_T \mid g, \Lambda_g^{-1}\right) && (16) \\ &= \operatorname*{argmin}_{\tau} C + \frac{1}{2}(x_T - g)^T \Lambda_g (x_T - g) && (17) \\ &= \operatorname*{argmin}_{\tau} \|x_T - g\|^2_{\Lambda_g} && (18) \end{aligned}$$

where Λ_g defines the precision matrix of the goal distribution. By setting the precision matrix to be the identity matrix, we recover the standard squared Euclidean distance between the terminal state x_T and goal state g. We also note that for any dimension d set to 0 to ignore this dimension results in an associated variance $\sigma_d^2 = \infty$.

C. Maximum Probability of Reaching Goal Point

Minimizing the M-projection with a Dirac-delta distribution $\delta_g(x)$ at a goal state $g \in \mathcal{X}$ and having arbitrary belief distribution $p(x \mid \tau)$ over the state when following trajectory τ, the KL divergence reduces to

$$D_{\text{KL}}\left(\delta_g(x) \parallel p(x \mid \tau)\right) = \underset{\tau}{\text{argmin}} \int_{x \in \mathcal{X}} \delta_g(x) \log \frac{\delta_g(x)}{p(x \mid \tau)} dx \tag{19}$$

$$= \underset{\tau}{\text{argmin}} \int_{x \in \{g\}} \log \frac{1}{p(x \mid \tau)} dx \tag{20}$$

$$= \underset{\tau}{\text{argmin}} - \log p(x = g \mid \tau) \tag{21}$$

$$= \underset{\tau}{\text{argmax}}\, p(x = g \mid \tau) \tag{22}$$

which is maximizing the probability of reaching the point-based goal g following trajectory τ.

In the special case where the probability distribution over state is a Gaussian, we recover the same weighted Euclidean distance cost as above albeit weighted by the belief state precision instead of the goal distribution precision.

D. Chance-Constrained Goal Set

Consider a uniform distribution $\mathcal{U}_{\mathcal{G}}(x)$ over goal set $\mathcal{G}$ and an arbitrary distribution $p(x \mid \tau)$ over the terminal state after following trajectory τ. Minimizing the M-projection of the KL divergence between these distributions we get

$$\underset{\tau}{\text{argmin}}\, D_{\text{KL}}(\mathcal{U}_{\mathcal{G}}(x) \parallel p(x \mid \tau)) = \underset{\tau}{\text{argmin}} \int_{x \in \mathcal{X}} \mathcal{U}_{\mathcal{G}}(x) \log \frac{\mathcal{U}_{\mathcal{G}}(x)}{p(x \mid \tau)} dx \tag{23}$$

$$= \underset{\tau}{\text{argmin}} \int_{x \in \mathcal{G}} u_{\mathcal{G}} \log \frac{u_{\mathcal{G}}}{p(x \mid \tau)} dx \tag{24}$$

$$= \underset{\tau}{\text{argmin}} \int_{x \in \mathcal{G}} u_{\mathcal{G}} \left[\log u_{\mathcal{G}} - \log p(x \mid \tau)\right] dx \tag{25}$$

$$= \underset{\tau}{\text{argmin}} - \int_{x \in \mathcal{G}} u_{\mathcal{G}} \log p(x \mid \tau) dx + \int_{x \in \mathcal{G}} u_{\mathcal{G}} \log u dx \tag{26}$$

$$= \underset{\tau}{\text{argmin}} - u_{\mathcal{G}} \int_{x \in \mathcal{G}} \log p(x \mid \tau) dx + C \tag{27}$$

$$= \underset{\tau}{\text{argmax}} \int_{x \in \mathcal{G}} \log p(x \mid \tau) dx \tag{28}$$

$$= \underset{\tau}{\text{argmax}} \int_{x \in \mathcal{G}} p(x \mid \tau) dx \tag{29}$$

Equation 29 defines the probability of reaching any state in the goal set $\mathcal{G}$, a commonly used term for reaching a goal set in chance-constrained control (e.g. Equation (6) in [9]).

APPENDIX II
EXPERIMENT ENVIRONMENTS

A. Dubins Car 2D Navigation

We use the Dubins car model from [4] which is a simple vehicle model with non-holonomic constraints in the state space $\mathcal{X} = SE(2)$. The state $\boldsymbol{x} = (p_x, p_y, \phi)$ denotes the car's planar position (p_x, p_y) and orientation ϕ. The dynamics obey

$$\dot{p}_x = v \cos\phi, \;\; \dot{p}_y = v \sin\phi, \;\; \dot{\phi} = u \tag{30}$$

where $v \in [0, v_{max}]$ is a linear speed and $u \in [-\tan\psi_{max}, \tan\psi_{max}]$ is the turn rate for $\psi_{max} \in \left(0, \frac{\pi}{2}\right)$.

We use an arc primitive parameterization similar to [4] to generate trajectory samples for CEM. Actions v, u are applied at each timestep for duration τ such that the robot moves in a straight line with velocity v if $u = 0$ and arcs with radius v/u otherwise. A trajectory with m arc primitives has the form $(v_1, u_1, \tau_1, \ldots, v_m, u_m, \tau_m) \in \mathbb{R}^{3m}$, which are sampled during CEM optimization.

The state space under this parameterization evolves as

$$p_x(t) = p_x(t_i) + \frac{v}{u_{i+1}}(\sin(\phi_i + \Delta t_i u_{i+1}) - \sin(\phi_i) \tag{31}$$

$$p_y(t) = p_y(t_i) + \frac{v}{u_{i+1}}(\cos\phi_i - \cos(\phi_i + \Delta t_i u_{i+1})) \tag{32}$$

$$\phi(t) = \phi_i + \Delta t_i u_{i+1} \tag{33}$$

where $\Delta t_i = t - t_i$ and $\phi_i = \phi(t_i)$ for the i^{th} primitive. Note we add a small value $u' = 1e-5$ to each u_i to avoid division by zero, which simplifies the computation described in [4]. We extend the model in [4] to have stochastic dynamics by adding Gaussian noise $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \alpha \boldsymbol{I})$ to the state updates in Equations 31-33; we use a value of $\alpha = 0.02$.

B. 7-DOF Arm Environment

We use the Franka Emika Panda arm rendered in rviz. The state is the arm's 7-D joint configuration in radians, where we compute state updates simply as

$$\boldsymbol{q}_{t+1} = \boldsymbol{q}_t + \boldsymbol{w}; \quad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{0}, \alpha \boldsymbol{I}) \tag{34}$$

We use PyBullet for checking collisions between the robot and the environment.

We use the same KL divergence cost and collision cost as the Dubins car environment. We add one additional cost term for the arm that specifies the arm's end-effector should reach a desired position in its workspace

$$\sum_{i=t}^{t+H} \lambda_i \|\boldsymbol{x}_d - FK(\boldsymbol{q})\|_2^2 \tag{35}$$

where $\boldsymbol{x}_d$ is the desired end-effector position to be reached, $FK(\cdot)$ is the robot's forward kinematics function, and $\lambda_i = \frac{i-t}{H}$. We compute the forward kinematics using PyBullet.