SUPPLEMENTARY MATERIALS

A. Scale-Invariance and "Volume" Interpretation of α

We show that Eq. 4 results in the formulation being scale-invariant with respect to b. Consider the same behavior under two different units b_1 and b_2 with $b_1 = c \cdot b_2$. For example, b_1 can be the trajectory length in centimeters and b_2 is the same quantity but in meters, and $c = 100$. Thus, $p(c \cdot b_1) = p(b_2)$ and $b_1^* = c \cdot b_2^*$. To maintain the same α level in Eq. 4, we need to have $\sigma_1 = c \cdot \sigma_2$. This implies that

$$p(t, \tau | \hat{b}_1 = b_1^*) = \frac{\mathrm{No}(b_1^*; b(\tau, t), \sigma_1^2) p(\tau|t)\pi(t)}{p(\hat{b}_1 = b_1^*)} \quad (13)$$

$$= \frac{\mathrm{No}(b_2^*; b(\tau, t), \sigma_2^2) p(\tau|e)\pi(t)}{p(\hat{b}_2 = b_2^*)} = p(t|\hat{b}_2 = b_2^*) \quad (14)$$

because $\mathrm{No}(b_1^*; b(\tau, t), \sigma_1^2) = \mathrm{No}(b_2^*; b(\tau, t), \sigma_2^2)$ due to the same scaling of $b_1 \sim b_2$ and $\sigma_1 \sim \sigma_2$, and $p(\hat{b}_1 = b_1^*) = p(\hat{b}_2 = b_2^*)$ as they are the same event. We conclude the posterior distribution is scale-invariant with respect to $b(\tau, t)$.

To motivate the bound of $[b^* - \sqrt{3}\sigma, b^* + \sqrt{3}\sigma]$ in Eq. 4, we consider a uniform approximation to $\mathrm{No}(b^*, \sigma^2)$. To match the mean b^* and standard deviation σ, $\mathrm{Unif}(b^* - \sqrt{3}\sigma, b^* + \sqrt{3}\sigma)$ is needed. If we use this uniform distribution in Eq. 2, the posterior Eq. 3 can be instantiated by sampling from the prior and rejecting tasks for which the trajectory behavior $b(\tau, t)$ falls outside of this bound. Thus, Eq. 4 specifies that the "volume" of $(\alpha \cdot 100)\%$ under $p(t, \tau)$ is maintained.

The same invariance and "volume" interpretation holds for Eq. 8 as well. The former stems from the standardization on b performed in Eq. 6. The latter uses the same uniform approximation but the bound is one-sided since $\beta \in (0, 1)$ by nature of the sigmoid transformation.

B. Dynamical System Modulation

We review the DS formulation proposed by [14], and present our problem-specific adaptations for 2D Navigation (Supp. E) and 7DoF arm reaching (Supp. H). A reader familiar with DS motion controllers may skip this review.

Given a target $\mathbf{x}^*$ and the robot's current state $\mathbf{x}$, a linear controller $\mathbf{u}(x) = \mathbf{x}^* - \mathbf{x}$ will guarantee convergence of $\mathbf{x}$ to $\mathbf{x}^*$ if there are no obstacles. However, it can easily get stuck in the presence of obstacles. [14] proposes a method to calculate a modulation matrix $M(\mathbf{x})$ at every $\mathbf{x}$ such that if the new controller follows $\mathbf{u}_M(\mathbf{x}) = M(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x})$, then $\mathbf{x}$ still converges to $\mathbf{x}^*$ but never gets stuck, as long as $\mathbf{x}^*$ is in free space. In short, the objective of the DS modulation is to preserve the linear controller's convergence guarantee while also ensuring that the robot is never in collision.

The modulation matrix $M(\mathbf{x})$ is computed from a list of obstacles, each of which is represented by a Γ-function. For the i-th obstacle $\mathcal{O}_i$, its associated gamma function Γ_i must satisfy the following properties:

- $\Gamma_i(\mathbf{x}) \leq 1 \iff \mathbf{x} \in \mathcal{O}_i$,
- $\Gamma_i(\mathbf{x}) = 1 \iff \mathbf{x} \in \partial\mathcal{O}_i$,
- $\exists\, \mathbf{r}_i, \text{s.t.} \forall\, t_1 \geq t_2 \geq 0, \forall\, \mathbf{u}, \Gamma_i(\mathbf{r}_i + t_1\mathbf{u}) \geq \Gamma_i(\mathbf{r}_i + t_2\mathbf{u})$.

In words, the Γ-function value needs to be less than 1 when inside the obstacle, equal to 1 on the boundary, greater than 1 when outside. This function must also be monotonically increasing radially outward from a specific point $\mathbf{r}_i$. This point is dubbed the *reference point*. From this formulation, $\mathbf{r}_i \in \mathcal{O}_i$ and any ray from $\mathbf{r}_i$ intersects with the obstacle boundary $\partial\mathcal{O}_i$ exactly once. The latter property is also the definition that $\mathcal{O}_i$ is "star-shaped" (Fig. 11). For most common (2D) geometric shape such as rectangles, circles, ellipses, regular polygons and regular star polygons, $\mathbf{r}_i$ can be chosen as the geometric center.

We first consider the case of a single obstacle $\mathcal{O}$, represented by Γ with reference point $\mathbf{r}$. Use d to denote the dimension of the space. We define

$$M(\mathbf{x}) = E(\mathbf{x})D(\mathbf{x})E^{-1}(\mathbf{x}). \quad (15)$$

We have

$$E(x) = [\mathbf{s}(\mathbf{x}), \mathbf{e}_1(\mathbf{x}), ..., \mathbf{e}_{d-1}(\mathbf{x})], \quad (16)$$

where

$$\mathbf{s}(\mathbf{x}) = \frac{\mathbf{x} - \mathbf{r}}{||\mathbf{x} - \mathbf{r}||} \quad (17)$$

is the unit vector in the direction of $\mathbf{x}$ from $\mathbf{r}$, and $\mathbf{e}_1(\mathbf{x}), ...,$ $\mathbf{e}_{d-1}(\mathbf{x})$ form a $d-1$ orthonormal basis to the gradient of the Γ-function, $\nabla\Gamma(\mathbf{x})$ representing the normal to the obstacle surface. $D(\mathbf{x})$ is a diagonal matrix whose diagonal entries are $\lambda_s, \lambda_1, ..., \lambda_{d-1}$, with

$$\lambda_s = 1 - \frac{1}{\Gamma(\mathbf{x})}, \quad (18)$$

$$\lambda_1, ..., \lambda_{d-1} = 1 + \frac{1}{\Gamma(\mathbf{x})}. \quad (19)$$

each eigenvalue determines the scaling of each direction. Conceptually, as the robot approaches the obstacle, this modulation decreases the velocity for the component in the reference point direction (i.e. toward obstacles) while increases velocity for perpendicular components. The combined effect results in the robot being deflected away tangent to the obstacle surface.

With N obstacles, we compute the modulation matrix $M_i(\mathbf{x})$ for every obstacle using the procedure above and the individual controllers $\mathbf{u}_{M_i}(\mathbf{x}) = M_i(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x})$. The final modulation is the aggregate of all the individual modulations. However, a simple average is insufficient since closer obstacles should have higher influence to prevent collisions.

[14] proposed the following aggregation procedure. Let $\mathbf{u}_i$ denote the individual modulations, and $\mathbf{u}$ denote the final aggregate modulation. Let n_i to denote the norm of $\mathbf{u}_i$. Defines $\mathbf{u}$ to be

$$\mathbf{u} = n_a \mathbf{u}_a, \quad (20)$$

where n_a is the aggregate norm, and $\mathbf{u}_a$ is the aggregate direction.

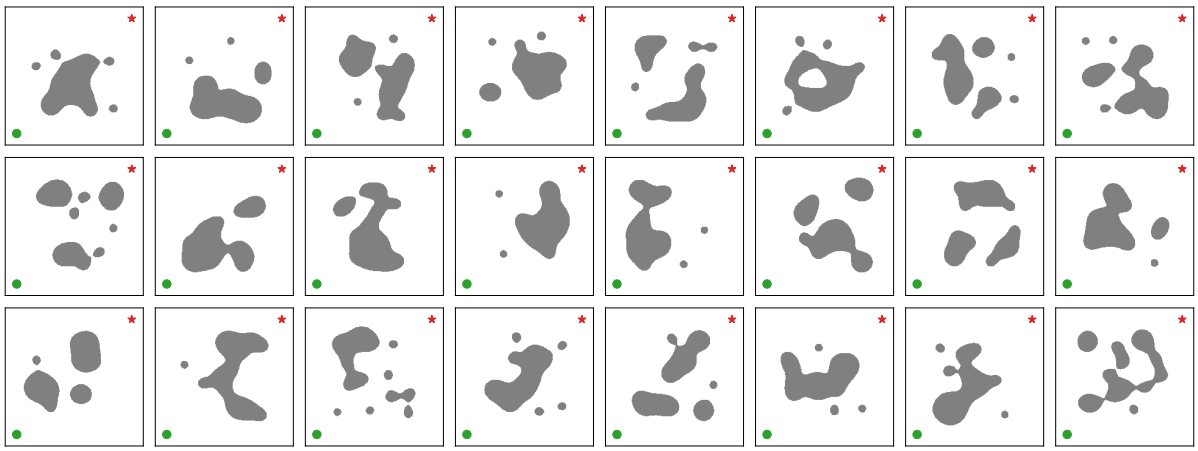

Fig. 8. An assortment of randomly generated RBF 2D environments, providing a sense of the diversity generated with this formulation. The green dots are the environment starting points and the red stars are navigation targets. We show DS modulation examples for the first three environments in Fig. 10.

The aggregate norm is computed as

$$n_a = \sum_{i=1}^{N} w_i n_i, \tag{21}$$

$$w_i = \frac{b_i}{\sum_{j=1}^{N} b_j}, \tag{22}$$

$$b_i = \prod_{1 \le j \le N, j \neq i} \Gamma_j(\mathbf{x}). \tag{23}$$

The above definition ensures that $\sum_{i=1}^{N} w_i = 1$, and $w_i \to 1$ when $\mathbf{x}$ approaches $\mathcal{O}_i$ (and only $\mathcal{O}_i$, which holds as long as obstacles are disjoint).

$\mathbf{u}_a$ is instead computed using what [14] calls "$\boldsymbol{\kappa}$-space interpolation." First, similar to the basis vector matrix $E(\mathbf{x})$ introduced above, we construct another such matrix, but with respect to the original controller $\mathbf{x}^* - \mathbf{x}$. We denote it as $R = [(\mathbf{x}^* - \mathbf{x})/||\mathbf{x}^* - \mathbf{x}||, \mathbf{e}_1, ..., \mathbf{e}_{d-1}]$, where $\mathbf{e}_1, ..., \mathbf{e}_{d-1}$ are again orthonomal vectors spanning the null space.

For each $\mathbf{u}_i$, we compute its coordinate in this new R-frame as $\hat{\mathbf{u}}_i = R^{-1}\mathbf{u}_i$. Its $\boldsymbol{\kappa}$-space representation is

$$\boldsymbol{\kappa}_i = \frac{\arccos(\hat{\mathbf{u}}_i^{(1)})}{\sum_{m=2}^{d} \hat{\mathbf{u}}_i^{(m)}} \left[\hat{\mathbf{u}}_i^{(2)}, ..., \hat{\mathbf{u}}_i^{(d)}\right]^T \in \mathbb{R}^{d-1}, \tag{24}$$

where the superscript (m) refers to the m-th entry. $\boldsymbol{\kappa}_i$ is a scaled version of the $\hat{\mathbf{u}}_i$ with the first entry removed. We perform the aggregation in this $\boldsymbol{\kappa}$-space using the weights w_i calculated above (25), transform it back to the R-frame (26), and finally transform it back to the original frame (27):

$$\boldsymbol{\kappa}_a = \sum_{i=1}^{N} w_i \boldsymbol{\kappa}_i \tag{25}$$

$$\hat{\mathbf{u}}_a = \left[\cos(||\boldsymbol{\kappa}_a||), \frac{\sin(||\boldsymbol{\kappa}_a||)}{||\boldsymbol{\kappa}_a||}\boldsymbol{\kappa}_a^T\right]^T \tag{26}$$

$$\mathbf{u}_a = R\hat{\mathbf{u}}_a. \tag{27}$$

As mentioned in Eq. 20, the final modulation is $\mathbf{u} = n_a \mathbf{u}_a$.

1) Tail-Effect: An artifact of the above formulation is the "tail-effect," where the robot is modulated to go around the obstacle even when it has passed by the obstacle and the remaining trajectory has no chance of collision under the non-modulated controller. This effect has been observed in [30] for a related but different type of modulation. Fig. 9 is reproduced from [30] (Fig. 7) showing the tail effect on the left and its removal on the right.

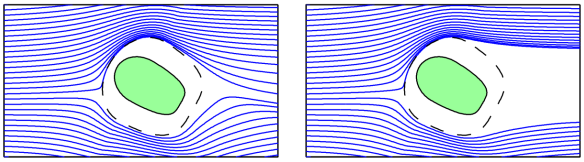

Fig. 9. Tail effect (left) and its removal (right), reproduced from Fig. 7 in [31]. The target is on the far right side.

This tail effect induces the placement of obstacles at the end of the "diagonal corridor" as seen in our straight-line deviation experiments (Fig. 4, left). We can modify the modulation to remove this effect.

C. RBF-Defined Environment Visualization

Fig. 8 depicts a randomly selected assortment of 2D environments. These environments demonstrate the flexibility and diversity of the RBF environment definition.

D. IL Controller for 2D Navigation

The imitation learning controller is a memoryless policy implemented as a fully connected neural network with two hidden layers of 200 neurons each and ReLU activations. The input is 18 dimensional, with two dimensions for the current (x, y) position of the robot, and 16 dimensions for a lidar sensor in 16 equally-spaced directions, with a maximum range of 1. The network predicts the heading angle θ, and the controller operates on the action of $[\Delta x, \Delta y] = [0.03\cos\theta, 0.03\sin\theta]$.

The network is trained on smoothed RRT trajectories. Specifically, we use the RRT controller to find and discretize

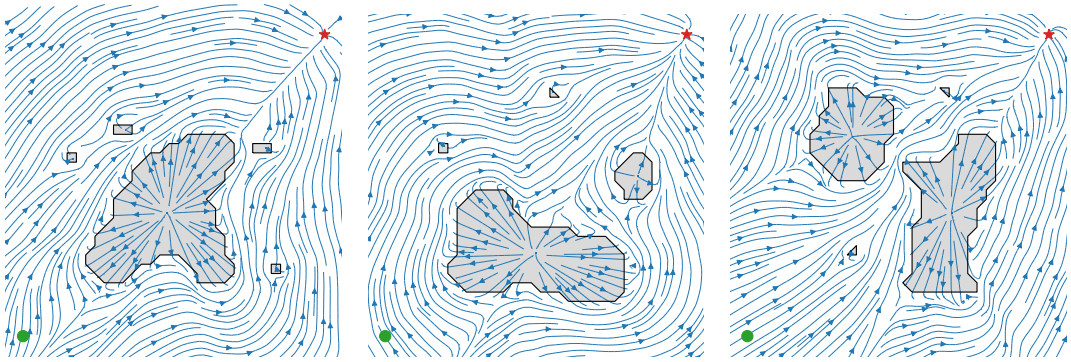

Fig. 10. Streamlines showing the modulation effect of the dynamical system for three 2D navigation tasks. The environments correspond to the first three examples of Fig. 8. Green dots are starting positions and red stars are navigation targets.

a trajectory. Then the smoothing procedure repeatedly replaces each point by the mid-point of its two neighbors, absent collisions. When this process converges, each point on the trajectory becomes one training data point.

Since only local observations are available and the policy is memoryless, the robot may get stuck in obstacles, which happens in approximately 10% of the runs. In addition, while the output target is continuous, a regression formulation with mean-squared error (MSE) loss is inappropriate, due to multimodality of the output. For example, when the robot is facing an obstacle, moving to either left or right would avoid it, but if both directions appear in the dataset, the MSE loss would drive the prediction to be the average, resulting in a head-on collision. This problem has been recognized in other robotic scenarios such as grasping [32] and autonomous driving [33]. We follow the latter to treat this problem as classification with 100 bins in the $[0, 2\pi]$ range.

E. DS Controller for 2D Navigation

For the DS controller, there are two technical challenges in using [14] on our RBF-defined environment. First, we need to identify and isolate each individual obstacle, and second, we need to define a Γ-function for each obstacle.

To find all obstacles, we discretize the environment into an occupancy grid of resolution 150×150 covering the area of $[-1.2, 1.2] \times [-1.2, 1.2]$. Then we find connected components using flood fill, and each connected component is taken to be an obstacle.

To define a Γ-function for each obstacle, we first choose the reference point as the center of mass of the connected component. Then we cast 50 rays in 50 equally spaced directions from the reference point and find the intersection point of each ray with the boundary of the connected component. Finally, we connected those intersections in sequence and get a polygon. In case of multiple intersection points, we take the farthest point as vertex of the polygon, essentially completing the non-star-shaped obstacle to be star-shaped, as shown in Fig. 11.

Given an arbitrary point $\mathbf{x}$, we define

$$\Gamma(\mathbf{x}) = \frac{||\mathbf{x} - \mathbf{r}||}{||\mathbf{i} - \mathbf{r}||}, \tag{28}$$

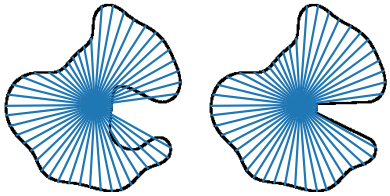

Fig. 11. Left: an obstacle which is not star-shaped. Some radial lines extending from the obstacle's reference point cross the boundary of the obstacle twice. Right: the same obstacle, modified to instead be star-shaped.

where $\mathbf{r}$ is the reference point and $\mathbf{i}$ is the intersection point with the polygon of the ray from $\mathbf{r}$ in $\mathbf{x}-\mathbf{r}$ direction. It is easy to see that this Γ definition satisfies all three requirements for Γ-functions listed in Supp. B.

Finally, to compensate for numerical errors in the process (e.g. approximating obstacles with polygons), we define the control inside obstacle to be the outward direction, which helps preventing the robot from getting stuck at obstacle boundaries in practice. Three examples of DS modulation of the 2D navigation environment are shown in Figure 10.

F. RRT Controller for 7DoF Arm Reaching

Since the target location is specified in the task space, we first find the target joint space configuration using inverse kinematics (IK). The initial configuration starts with the arm positioned down on the same side as the target. If the IK solution is in collision, we simulate the arm moving to it using position control, and redefine the final configuration at equilibrium as the target (i.e. its best effort reaching configuration). We solve the IK using Klamp't [34].

G. RL Controller for 7DoF Arm Reaching

The RL controller implements the proximal policy gradient (PPO) algorithm [25]. The state space is 22-dimensional and consists of the following:

- 7D joint configuration of the robot,
- 3D position of the end-effector,
- 3D roll-pitch-yaw of the end effector,
- 3D velocity of the end-effector,
- 3D position of the target,
- 3D relative position from the end-effector to the target.

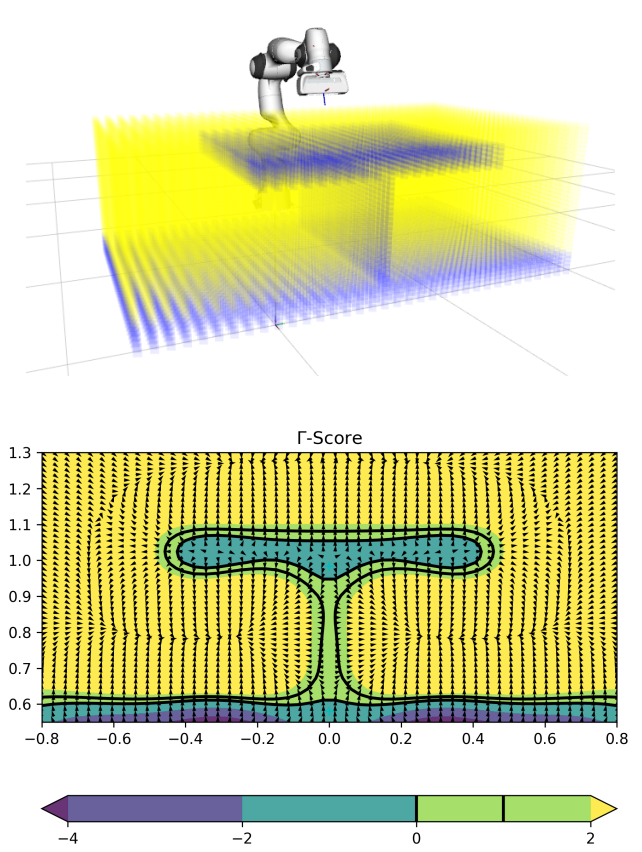

Fig. 12. Above: the division of 3D space as either containing an obstacle or free space. This data is used to train an SVM, which acts as an interpolator. The classification scores of the SVM are used as the Γ function for this 3D reaching task. Below: a 2D slice showing the smoothed Γ scores.

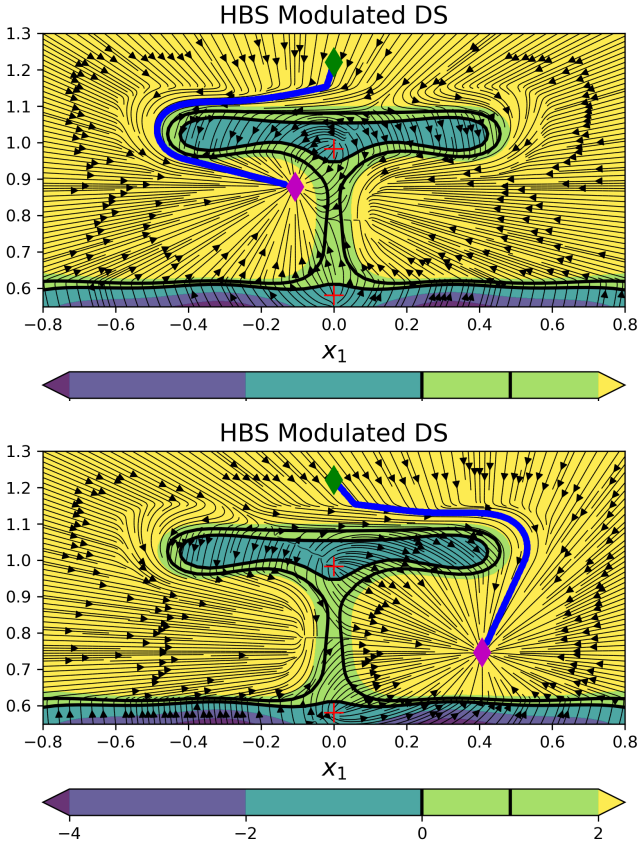

Fig. 13. Cross-sections showing streamlines of the dynamical system modulation effect for two distinct targets in the 3D reaching task. Red crosses indicate reference points. Green diamond is the initial position of the end-effector for all experiments.

The action is 7-dimensional for movement in each joint, which is capped at $[-0.05, 0.05]$.

Both the actor and the critic are implemented with fully connected networks with two hidden layers of 200 neurons each, and ReLU activations. The action is parametrized as Gaussian where the actor network predicts the mean, and 7 standalone parameters learns the log variance for each of the 7 action dimensions. At test time, the policy deterministically outputs the mean action given a state.

H. DS Controller for 7DoF Arm Reaching

For the DS controller in the 7DoF arm reaching task, we face the same challenges as in the 2D navigation task. Namely, defining an appropriate Γ-function for the obstacle configuration that holds the three properties introduced in [14] (listed in Supp. B). Additionally, as the DS modulation technique does not consider the robot's morphology, end-effector shape or workspace limits (as it only modulates a point-mass) we implement several adaptations to allow for this technique to be used properly with a 7DoF robot arm. Specifically, the modulated linear controller $\mathbf{u}_M(\mathbf{x}) = M(\mathbf{x}) \cdot \mathbf{u}(\mathbf{x})$, with a modulation matrix $M(\mathbf{x})$ computed as in [14] and described in Supp. B, controls for the 3D position of the tip of the end-effector. This 3D position is shown as the small reference frame at the edge of the rectangular surface of the end-effector in Fig. 12. The desired velocity of the end-effector tip, given by the modulated linear controller, is then tracked by the 7DoF arm via the position-level IK solver used in the RRT implementation described in Supp. F. Following we describe the details of these adaptations.

1) SVM-Defined Γ-function for Constrained Workspaces: The original approach for the DS modulation strategy [30] uses spherical or ellipsoidal geometric representations to define the Γ-functions pertaining to each obstacle. Such representations are also used in the improved approach that we adopt in this work [14]. In such works, when multiple convex obstacles are present in the environment, these are defined by ellipsoids that encapsulate the entire obstacle. On the other hand, if the obstacles are non-convex and star-shaped, a collection of ellipsoids with a common intersection is used to parametrize the Γ-functions. This approach works well in practice, however, representing star-shaped objects as collections of ellipsoids or even convex objects as ellipsoids might lead to undesirable behaviors. They can produce either overly conservative collision avoidance behaviors or an under-specified Γ-function that cuts corners and wrongfully leads to collisions at the edges of tables or other type of rectangular-shaped objects. Hence, the success of this DS modulation strategy relies heavily on the correct placement and fitting of the ellipsoids on the obstacles and also the

correct placement of the reference points $\mathbf{r}_i$ corresponding to the i-th obstacle $\mathcal{O}_i$. In the 2D navigation task, we solved this issue by representing the obstacles as polygonal-based Γ-functions (see Supp. E). This approach works well in 2D for disconnected obstacles, yet, it is not easily transferable to 3D. Furthermore, in the case of our proposed constrained table-top environment the obstacle configuration is not star-shaped, hence, a more general Γ-function is necessary.

To avoid the geometric error-prone Γ-functions, we take inspiration from prior work in data-driven self-collision avoidance boundary learning [27]. In this work, a single Γ-function is used to represent the free-space and collided regions of the robot's workspace in joint-space coordinates. Given a dataset of collided and non-collided robot configurations, a Γ-function of class $\mathcal{C}^1$ is learned by framing the problem as a binary classification solved via support vector machines (SVM). As shown in Fig. 12, we discretize the 3D workspace of the robot and generate a dataset of collided positions as (-1) class and the free-space as (+1) class. By using the radial basis function (RBF) kernel $K(\mathbf{x}_1, \mathbf{x}_2) = e^{-\gamma||\mathbf{x}_1 - \mathbf{x}_2||^2}$, where parameter γ defines kernel width, the SVM decision function $\Gamma(\mathbf{x})$ has the following form:

$$\begin{aligned}\Gamma(\mathbf{x}) &= \sum_{i=1}^{N_{sv}} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \\ &= \sum_{i=1}^{N_{sv}} \alpha_i y_i e^{-\gamma||\mathbf{x}-\mathbf{x}_i||^2} + b,\end{aligned} \tag{29}$$

and the equation for $\nabla\Gamma(\mathbf{x})$ is naturally derived as follows:

$$\begin{aligned}\nabla\Gamma(\mathbf{x}) &= \sum_{i=1}^{N_{sv}} \alpha_i y_i \frac{\partial K(\mathbf{x}, \mathbf{x}_i)}{\partial \mathbf{x}} \\ &= -\gamma \sum_{i=1}^{N_{sv}} \alpha_i y_i e^{-\gamma||\mathbf{x}-\mathbf{x}_i||^2} (\mathbf{x} - \mathbf{x}_i).\end{aligned} \tag{30}$$

In Eq. 29 and 30, $\mathbf{x}_i$ $(i = 1, ..., N_{sv})$ are the support vectors from the training dataset, y_i are corresponding collision labels (-1 if position is collided, $+1$ otherwise), $0 \leq \alpha_i \leq C$ are the weights for support vectors and $b \in \mathbb{R}$ is decision rule bias. Parameter $C \in \mathbb{R}$ is a penalty factor used to trade-off between errors minimization and margin maximization. We empirically set the hyper-parameters of the SVM to $C = 20$ and $\gamma = 20$. Parameters α_i and b and the support vectors $\mathbf{x}_i$ are estimated by solving the optimization problem for the soft-margin kernel SVM using the Python scikit-learn library.

2) 7DoF Arm Position Control with 3D Modulated DS: Given a desired modulated 3D velocity for the end-effector tip, $\dot{\mathbf{x}}_M = \mathbf{u}_M(\mathbf{x})$, we compute the next desired 3D position by numerical integration:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_M(\mathbf{x}_t)\Delta t \tag{31}$$

where $\mathbf{x}_t, \mathbf{x}_{t+1} \in \mathbb{R}^3$ are the current and next desired 3D position of the tip of the end-effector and $\Delta t = 0.03$ is the control loop time step. $\mathbf{x}_{t+1}$ is then the target in Cartesian world space coordinates that defines the objective of the position-based IK solver implemented in Klamp't [34].

I. Transition Kernel in MCMC Sampling

We used a truncated Gaussian transition kernel for all experiments. For the RBF-defined 2D environment, we initialize 15 obstacle points with coordinates sampled uniformly in $[-0.7, 0.7]$. The transition kernel operates independently on each obstacle coordinate: given the current value of x, the kernel samples a proposal from $\mathrm{No}(\mu = x, \sigma^2 = 0.1^2)$ truncated to $[-0.7, 0.7]$ (and also appropriately scaled). For the arm reaching task, the target is sampled uniformly from two disjoint boxes, with the left box at $[-0.5, -0.05] \times [-0.3, 0.2] \times [0.65, 1.0]$ and the right box at $[0.05, 0.5] \times [-0.3, 0.2] \times [0.65, 1.0]$. Again, we use the same transition kernel with $\sigma_x = 0.1, \sigma_y = 0.03, \sigma_z = 0.035$ in three directions. Again, the distribution is truncated to the valid target region ($x \in [-0.5, -0.05] \cup [0.05, 0.5], y \in [-0.3, 0.2], z \in [0.65, 1.0]$). In other words, the transition kernel implicitly allows for the jump across two box regions.

In addition, the stochastic RRT controller also requires a transition kernel. As discussed in Sec. V, we initialize its values on an as-needed basis. When necessary, we sample a configuration uniformly between the lower- and upper-limit (i.e. $[x_L, x_U)$. For each configuration, the same Gaussian kernel truncated to $[x_L, x_U]$, with $\sigma = 0.1(x_U - x_L)$.

J. Additional Results for 2D Navigation

TABLE I

MEANS REPORTED FOR 500 SAMPLES DRAWN FROM EACH OF THE PRIOR AND POSTERIOR FOR EACH 2D NAVIGATION CONTROLLER.

Control	Metric	Target	Prior Mean	Posterior Mean
DS	Length	0	203	166
	Avg. Jerk	0	1.84e−3	1.46e−3
	Straight	0	0.256	0.084
	Legibility	min	0.819	0.650
	Obstacle	0	0.309	0.229
	Obstacle	max	0.309	0.611
IL	Length	0	161	161
	Avg. Jerk	0	6.95e−4	3.19e−4
	Straight	0	0.378	0.301
	Legibility	min	0.877	0.784
	Obstacle	0	0.262	0.218
	Obstacle	max	0.262	0.387
RRT	Length	0	182	174
	Avg. Jerk	0	4.24e−4	2.79e−4
	Straight	0	0.470	0.162
	Legibility	min	0.798	0.669
	Obstacle	0	0.312	0.241
	Obstacle	max	0.312	0.442

Table I presents a quantitative overview of the performance of our sampling procedure, allowing us to compare behaviors between different controllers. For example, we find the DS controller has the highest variability on the obstacle clearance min and max tasks: it has the largest obstacle avoidance metric mean (0.611), and the second lowest obstacle clearance metric mean $(0, 229)$. Where behavior metrics are positive real numbers (e.g., trajectory length, obstacle clearance), a

target of 0 is functionally equivalent to using the maximal mode approach (with $b^* \to -\inf$) target.

In addition to this full summary of the quantitative results, we include visual overviews of the maximum obstacle clearance behavior and the minimum obstacle clearance behaviors for all three controllers in Figure 14. For the DS formulation, obstacle avoidance is best achieved when the environment clusters obstacles in the center. In contrast, for both RRT and IL, obstacle avoidance is best achieved when the obstacles are clustered in the upper left corner. For minimal obstacle clearance behaviors, obstacles are clustered near the start and end positions for all three controllers.

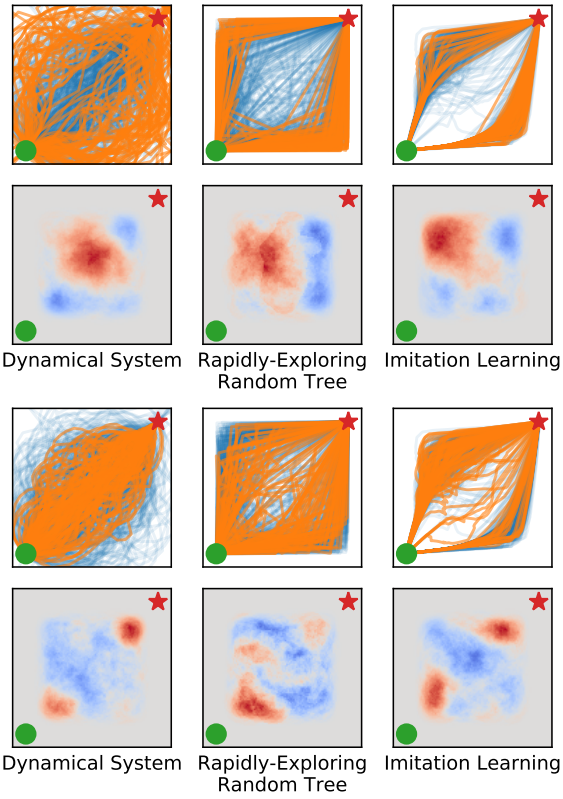

Fig. 14. First two rows: trajectories which maximize obstacle clearance from the prior (blue) and posterior (orange) for each controller, and the corresponding obstacle density plots (red indicates obstacles are more likely present). Below: the same, but for minimized obstacle clearance instead.

K. Additional Results for 7DoF Arm Reaching

Table II presents quantitative overviews of some behaviors for the 7DoF arm reaching task.

TABLE II

MEANS REPORTED FOR 500 SAMPLES DRAWN FROM EACH OF THE PRIOR AND POSTERIOR FOR EACH 7DOF ARM REACHER CONTROLLER.

Control	Metric	Target	Prior Mean	Posterior Mean
DS	Avg. Jerk	0	$2.31\mathrm{e}-4$	$5.98\mathrm{e}-5$
	Straight	0	0.980	0.913
	EE Dist	0	0.934	0.623
RL	Avg. Jerk	0	7.17e−3	5.50e−3
	Straight	0	0.858	0.762
	EE Dist	0	0.958	0.691
RRT	Avg. Jerk	0	1.87e−3	4.92e−4
	Straight	0	1.223	0.897
	EE Dist	0	3.741	1.192