Online Implicit Surfaces for Obstacle Modeling and Avoidance

Abhinav Kumar[1], Dmitry Berenson[1]

***Abstract*—Adapting model-based control to novel environments is challenging as dynamics models learned offline may not generalize to the obstacle configuration of the novel environment. We propose a method to identify and avoid obstacles online whose geometry is not known *a priori* without updating the offline, nominal, dynamics. Our method relies on a Gaussian process implicit surface (GPIS) to construct data-efficient obstacle representations using visual and inferrred contact data derived from observed states and dynamics predictions. This allows us to design a model predictive controller (MPC) using the uncertainty estimates provided by the GPIS to successfully navigate around obstacles to complete multiple manipulation tasks. By modeling the environment instead of directly adapting the dynamics, our method is able to solve both low-dimensional peg-in-hole tasks and high-dimensional rope and cable manipulation tasks. This enables our method to succeed in 30/30 trials vs 15/30 for a baseline on a simulated rope manipulation task while requiring 63% fewer control steps to succeed.**

I. Introduction

In unstructured or cluttered environments where full geometric models of obstacles are not available *a priori*, special care must be taken to avoid obstacle collisions that could prevent task completion. The challenge is heightened when manipulating objects with high-dimensional states, for example rope. In this work, we develop a method to identify and avoid *a priori* unknown obstacles online using visual and contact information.

We use a Gaussian process implicit surface (GPIS) [1] to model obstacles. GPIS uses a Gaussian Process (GP) to learn a 0-level-set surface that we use to model obstacle geometry. Our method learns a GPIS using visual information from a fixed view of the scene as well as contact information inferred from tracking the state of the manipulated object. With access to a nominal dynamics model, we use tracked state information to learn where nominal dynamics are incorrect online. This enables obstacle modeling without specialized tactile sensing, which may not be available along the surface of a manipulated object, or full visibility of the obstacle.

In addition, using a GP provides us access to uncertainty estimates over the model's prediction. The GP predicts a Gaussian distribution for a given input and the variance of that distribution can be used to estimate the uncertainty. The uncertainty provides a signal that can be used to tune the sensitivity of the controller to the GPIS's predictions as well as aid in escaping what would otherwise be local minima by seeking out unvisited regions of the state space.

[1] Robotics Department, University of Michigan, Ann Arbor, USA {abhin, dmitryb}@umich.edu

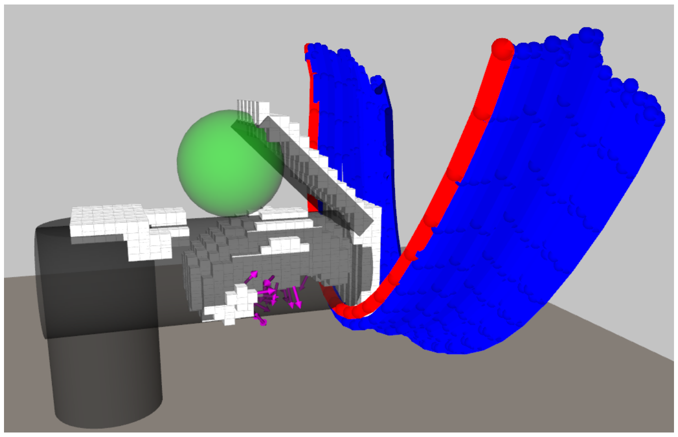
Fig. 1. Our method learns a continuous representation of the obstacle geometry, shown here as a voxel grid with surface normals approximated by the pink gradients. The blue MPC rollout shows the planned path out of the previously unknown obstacle as it moves toward the green goal. Our method reasons about occluded regions and is able to estimate the locations of obstacles. Robot not visualized.

Prior work has focused on directly adapting dynamics models to new environments or learning neural networks that model where dynamics are inaccurate [2]–[4]. These methods can be used to reason about obstacles. However, our GPIS representation has properties that can more directly be used to inform planning, mainly the uncertainty of its predictions.

GPIS has previously been used to model environments and object geometries [5]–[12]. These works leverage uncertainty to help construct more accurate geometric models. We use the uncertainty to escape local minima of the controller and inform our obstacle identification to jointly tackle the problems of identifying and avoiding obstacles. These works assume access to rich perception signals, including visual data with dynamic viewpoints or tactile sensors. We do not assume that our viewpoint of the system can change over time nor do we have access to tactile data when grasped objects make contact with obstacles, instead relying on a combination of limited visual data with contacts estimated through state tracking and knowledge of nominal dynamics.

We show that our method results in higher success rates and shorter episode lengths than baselines. We show that existing methods that reason about local minima of MPC controllers to detect obstacles such as [13] are insufficient for consistently solving the rope and cable manipulation tasks and planning methods that do not reason online about the environment fail when obstacles are occluded.

Specifically, our contributions are a method for learning an obstacle representation online using a fusion of visual data and contact data inferred through state tracking and an MPC controller that leverages GPIS uncertainty estimates to inform obstacle avoidance.

II. Problem Statement

Let $\mathbf{u} \in \mathcal{U}$ represent the control signal and $\mathbf{X} \in \mathcal{X}$ represent the state of a grasped object. We assume $\mathbf{X} = (\mathbf{x}^1...\mathbf{x}^n)$, meaning $\mathbf{X}$ can be represented as an ordered set of n components where $\mathbf{x}^i \in \mathbb{R}^3$. This representation is useful for high-dimensional systems like deformable objects, which can be represented as a collection of particles or points of interest. For example, a rope can be represented as a set of ordered points. An example where $n = 1$ is a peg grasped by a robot in a peg-in-hole task. This flexible representation enables reasoning about collisions between different components of manipulated objects and the environment.

Given an initial state $\mathbf{X}_0$ and a goal set G, which we assume to be reachable, we seek to find a trajectory τ using model predictive control (MPC) that reaches G with a minimal number of control steps. G specifies goal locations for a subset of components of $\mathbf{X}$. G_i is the goal location for a specific component i and not all components may have goal configurations. An example would be where there is a goal location for the center of a rope but not for other points along the object. A trajectory τ has a horizon T, a sequence of controls $\tau_{\mathbf{u}} = \{\mathbf{u}_0...\mathbf{u}_{T-1}\}$, and a sequence of states $\tau_{\mathbf{X}} = \{\mathbf{X}_0...\mathbf{X}_T\}$.

We assume access to a depth image D and corresponding point cloud P of the environment collected without occlusion from the robot or manipulated object. Pre-generated visual data prevent visibility issues online due to robot occlusion. We assume that only the robot and manipulated object state change over time, meaning the environment is static.

We assume access to a function $d_x(\mathbf{x}_1, \mathbf{x}_2)$ that provides a distance between state components.

We assume access to a dynamics model $f(\mathbf{X}, \mathbf{u})$ that predicts the next state of the system given the current state and some control. f will be applied to a novel environment with obstacles that f may not be able to model.

This work addresses modeling obstacles online and guiding the controller to regions where the prediction error of f is relatively low, thus allowing task completion.

III. Method

Our method can be split into two parts: learning obstacle representations online and using that information for control.

A. *Online Obstacle representation*

We learn a GPIS online during task execution. GPIS uses a Gaussian process (GP) to fit a 0-level-set surface given exterior, surface, and interior points. The GPIS is defined as:

$$\texttt{GPIS} : \mathbb{R}^3 \to \mathbb{R}; \texttt{GPIS}(\mathbf{x}) \begin{cases} < 0 & \text{if } \mathbf{x} \text{ is interior} \\ = 0 & \text{if } \mathbf{x} \text{ is on the surface} \\ > 0 & \text{if } \mathbf{x} \text{ is exterior} \end{cases} \tag{1}$$

We generate labels from state observations, using dynamics, and use visual data to clean these labels. This enables obstacle detection in visible and occluded regions of the environment.

1) Dynamics-Based Label Generation: For an observed transition $(\mathbf{X}_t, \mathbf{u}_t, \mathbf{X}_{t+1})$, we identify regions of state space where f is inaccurate by comparing the realized next state $\mathbf{X}_{t+1}$ to the state predicted by the dynamics $\hat{\mathbf{X}}_{t+1}$. We generate labels $\mathbf{Y}_{t+1}, \hat{\mathbf{Y}}_{t+1} \in \mathbb{R}^n$, corresponding to $\mathbf{X}_{t+1}$ and $\hat{\mathbf{X}}_{t+1}$, using Equations (2) and (3) for each state component.

$$\mathbf{Y}^i_{t+1} = \min\left(\gamma \frac{d_{\mathbf{x}}(\mathbf{X}^i_t, \mathbf{X}^i_{t+1})}{d_{\mathbf{x}}(\mathbf{X}^i_t, \hat{\mathbf{X}}^i_{t+1})}, 1\right) \tag{2}$$

$$\hat{\mathbf{Y}}^i_{t+1} = 2\mathbf{Y}^i_{t+1} - 1 \tag{3}$$

We use $\gamma \in [1, \infty)$ to account for inaccuracies in f in freespace, where higher values of γ lead to a higher tolerance of dynamics error. This is useful when working with learned dynamics models or inaccurate simulators. At each timestep, we add data from $\mathbf{X}_{t+1}$ and $\hat{\mathbf{X}}_{t+1}$

Calculating the labels in this way uses knowledge of nominal dynamics to estimate unreachable regions. Visited state components $\mathbf{X}^i_t$ have a label between 0 and 1 and lower values correspond to less accurate dynamics. $\hat{\mathbf{X}}^i_t$ is interior when $\hat{\mathbf{Y}}^i_t \leq 0$. γ controls the associated dynamics error threshold. Adding both $\mathbf{Y}^i_t$ and $\hat{\mathbf{Y}}^i_t$ to the GPIS allows it to interpolate a 0-level-set surface between positive and negative labels.

When $\hat{\mathbf{X}}_{t+1}$ is added to the GPIS, it is possible to place interior points in freespace, thereby incorrectly blocking paths to the goal. To compensate for this, we include a parameter δ which interpolates between $\mathbf{X}_t$ and $\hat{\mathbf{X}}_{t+1}$. We then add components from the interpolation, $\hat{\mathbf{X}}^{\delta}_{t+1}$, to the GPIS. In scenarios with tight corridors or thin obstacles, a lower value of δ can be helpful.

2) Visual Label Cleaning: Only using dynamics to generate labels can lead to false positives as one part of an object being in contact with an obstacle can cause the entire object to stop moving. To help address this, we use visual data to clean the labels. Specifically, we determine if components are visible and, if so, whether they are in contact with the environment.

We construct visual labels by projecting component coordinates into image coordinates. Given the intrinsic and extrinsic camera parameters, we can recover pixel coordinates (u, v) and depth z for a state component. Letting $D(u, v)$ be the depth value in the depth image D at (u, v), a state component is visible if $z < D(u, v)$. Visible components are given a label of 1. Components of $\mathbf{X}$ that are within a distance r_c of P are marked as in contact, meaning a label of 0. r_c is a parameter whose value is informed by the resolution of the point cloud and the geometry of the grasped object. We do not add the dynamics predictions of visible freespace components to the GPIS to avoid incorrectly adding interior points to the GPIS.

We add data to the GPIS at each timestep and use gradient descent to tune the GP kernel parameters. We seed the GPIS at initialization with G with label of 1 since we assume goal configurations are reachable.

3) Visual Mean Prior: In addition to using visual information to update labels, we also use it to set a prior mean function for the GPIS. We use our visual observations to construct a more informative prior than assuming that unvisited states are in freespace. We construct a voxel grid from the initial point cloud observation of the scene where occupied voxels have a value of -1, labeling them as interior points. We calculate our prior mean for a query point by indexing into this voxel grid.

4) Maintaining Tractability: Gaussian processes struggle with large amounts of data due to the computational complexity of computing posterior predictions. To address this issue, we limit the size of the online data set to a value D_{max} and remove excess data points. We remove data whose label values are closest to the mean label value of the data set to preserve diversity of the data.

B. Control

Our MPC cost function is $J(\tau) = J_g(\tau) + \alpha J_o(\tau) + J_c(\tau) + \beta J_e(\tau)$ where the different terms are: a goal directed cost J_g, an obstacle repulsion cost J_o, a collision cost J_c, and an exploration cost J_e. $\alpha, \beta \in \mathbb{R}$ are used to weigh the costs.

1) Goal Cost: The goal cost in (4) drives the controller toward the goal. $\mathbb{1}_{g_t}$ is 1 when the goal distance is less than r_g for all state components with defined goals and is weighted by a parameter $\eta \in \mathbb{R}$. r_g is the distance from the goal that would indicate task success. $\mathbb{1}_{g_t}$ helps the controller converge to the goal by creating a deeper basin in the cost function near the goal that the controller can exploit.

$$J_g(\tau) = \sum_{t=1}^{T} \left(-\eta \cdot \mathbb{1}_{g_t} + \sum_{i \in G} d(G_i, \mathbf{X}_t^i) \right) \tag{4}$$

2) Obstacle Repulsion Cost: We construct a cost in (5) that repels the controller from the surface predicted by the GPIS. By making posterior predictions on the samples in the GPIS's data set, we can identify a set of obstacle points $\mathcal{O}$.

$$J_o(\tau) = \frac{1}{|\mathcal{O}|} \sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{o \in \mathcal{O}} \frac{\texttt{align}(\mathbf{x}_{t-1}^i, \mathbf{x}_t^i, o)}{d_{\mathbf{x}}(\mathbf{x}_t^i, o)^2} \tag{5}$$

We use the uncertainty of the GPIS prediction to construct $\mathcal{O}$. For a component in the GP's data set, let μ be the posterior mean of the GP at that point and let σ be the posterior standard deviation. $\mu + Z\sigma$, where μ is the posterior mean, σ is the posterior standard deviation, and Z is the Z-score of a standard Normal distribution corresponding to a given confidence level $\zeta \in (0, 1)$ computes an upper bound on the GP's prediction. Points where this upper bound is less than or equal to 0 are added to $\mathcal{O}$. By incorporating the uncertainty in this way, we can choose to be more or less conservative about obstacle avoidance. Higher values of ζ are useful in situations with tight corridors or thin obstacles.

We penalize motion toward detected obstacle points by comparing the trajectory with the GPIS gradient. We define $\texttt{align}(\mathbf{x}_t, \mathbf{x}_{t-1}, o) = \max(cos(\mathbf{x}_t - \mathbf{x}_{t-1}, -\nabla\texttt{GPIS}(o)), 0)$, where cos is the cosine similarity and $-\nabla\texttt{GPIS}(o)$ is the negative gradient of the GPIS at an obstacle point $o \in \mathcal{O}$.

3) Exploration Cost: This cost $J_e(\tau) = -\sum_{t=1}^{T} {\sigma^2}_t^s$ uses the variance of the GPIS at a state $\mathbf{X}$, where ${\sigma^2}_t \in \mathbb{R}^n$ is the posterior variance for the n components. As in prior work, we use the uncertainty to gain information by exploring new regions of state space. We also find this form of exploration helps escape from what would otherwise be local minima.

We choose one component $s \in [1, n]$ every T_e timesteps whose corresponding variance we optimize. We choose the component with the minimum posterior mean prediction from the GPIS. This is done to focus exploration on the part of the object that is most likely to be stuck.

4) Collision Cost: We define a collision cost in (6). This cost identifies transitions that would intersect with the surface predicted by the GPIS. This is done by calculating the same predicted upper bound as for J_o for states along the rollout.

$$J_c(\tau) = \sum_{t=1}^{T} \sum_{i=1}^{n} C \cdot \mathbb{1}_{\texttt{GPIS}(\mathbf{x}_t^i) + Z\sigma_t \leq 0} \tag{6}$$

IV. Results

We evaluate our method on peg-in-hole and rope manipulation tasks, demonstrating the method's utility across varied state dimensionalities. We use model predictive path integral control (MPPI) [14] for our MPC controller. We execute 1 step of the plan, replanning at each timestep. We use a Matern kernel for the GPIS, implemented through [15]. Parameter values can be found in Table II in Appendix A.

A. Peg-in-Hole

We use the peg-in-hole tasks defined in [13] in which a grasped peg simulated in PyBullet [16] is navigated to a hole. A success is placing the peg within 2 cm of the hole within 500 control steps. The state is (x, y, z, R_x, R_y), where (x, y, z) is the $\mathbb{R}^3$ position of the end-effector and (R_x, R_y) are reaction forces on the end-effector. The control signal is $(\Delta x, \Delta y)$. We use the $\mathbb{R}^3$ position for the GPIS. For these tasks, $n = 1$.

We compare our method to TAMPC [13] and use their dynamics model learned wihout the presence of obstacles for f. We do not use any visual data for our method to provide a comparison to TAMPC, which does not use visual input. We also perform an ablation by removing uncertainty information. We remove the exploration cost and do not calculate upper bounds for the collision and obstacle repulsion costs.

We evaluate for 30 trials. We achieve higher success rates than TAMPC and our ablation on these tasks with a shorter episode length, as shown in Table I. We believe this is due to the learned GPIS providing a dense representation of the environment, allowing for collision checking as well as providing an uncertainty signal that we can use for exploration. The learned surface also approximates the true obstacle configuration in the regions visited by the robot, as seen in Fig. 2.

B. Simulated Rope Manipulation

In this task, a two-armed, 16-dof robot removes a rope from under a hook. The hook has a barrier that occludes part of the top of the obstacle, as shown in Fig. 3. A success is defined

Environment	Method	Success	Control Steps (Given Success)
Peg-U	Ours	**.93**	$\mathbf{95.4 \pm 16.9}$
	TAMPC	.80	232.7 ± 48.0
	Unc. Ablation	.5	158.1 ± 67.3
Peg-I	Ours	**.97**	$\mathbf{150.1 \pm 18.3}$
	TAMPC	.67	215.7 ± 23.3
	Unc. Ablation	0	-
Peg-T	Ours	**.97**	$\mathbf{83.3 \pm 21.2}$
	TAMPC	.80	152.9 ± 28.4
	Unc. Ablation	.1	452.7 ± 10.8
Sim. Rope Manipulation	Ours	**1**	$\mathbf{128.4 \pm 18.9}$
	Unc. Ablation	.87	264.9 ± 44.6
	TAMPC	.5	345.5 ± 38.4
	Partial SDF	0	-

TABLE I
SUCCESS RATES AND 95% CONFIDENCE INTERVALS FOR CONTROL STEPS. CONTROL STEP STATISTICS ARE CALCULATED FOR SUCCESSFUL TRIALS.

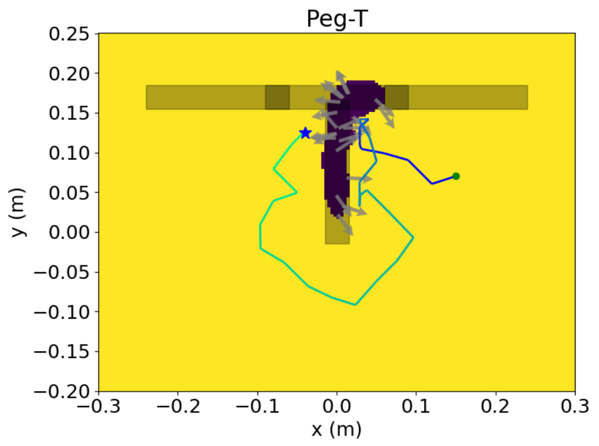

Fig. 2. Trajectory and surface for the Peg-T task. The modeled obstacle with confidence adjustment is in purple. The arrows represent GPIS obstacle points and point in the direction of the gradient. Trajectory starts at the green circle and ends at the blue star.

as placing the center of the rope in a sphere of radius 5cm above the hook within 500 steps.

We learn f offline using dynamics data generated with Mujoco [17]. In simulation, we attach the rope to floating grippers and sample random trajectories. We represent the rope as 25 articulated links and track the $\mathbb{R}^3$ position for each link. The set of 25 positions is the set of state components. The control is $[\Delta p_l, \Delta p_r] \in \mathbb{R}^6$, where Δp_l is the $\mathbb{R}^3$ change in the left gripper's position and Δp_r is the $\mathbb{R}^3$ change in the right gripper's position. We fit a 3-layer multi-layer perceptron (MLP) to this data. The MLP's hidden layer is of size 512.

Our MPPI controller outputs changes in gripper positions. We use the Jacobian psuedoinverse method [18] in the DeepMind Control Suite [19] to calculate the new joint positions. We use CDCPD2 [20] to estimate the rope state to fit our partial observability assumption. CDCPD2 includes regularization terms that promote smoothness of the estimate and prevent large deviations in the estimate between timesteps, leading to reasonable estimates for occluded portions of the rope.

We compare our method to TAMPC, the ablation of the uncertainty, and a baseline which directly uses the partial visual information without any online adaptation. For the TAMPC state distance function used to determine if the controller is

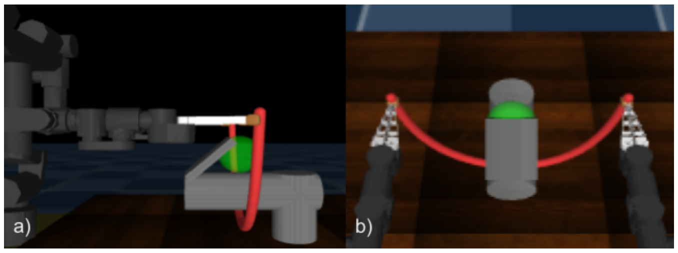

Fig. 3. Simulated rope task. **a)** The robot moves the center of the rope to the green goal region. The barrier occludes part of the obstacle. **b)** Camera angle

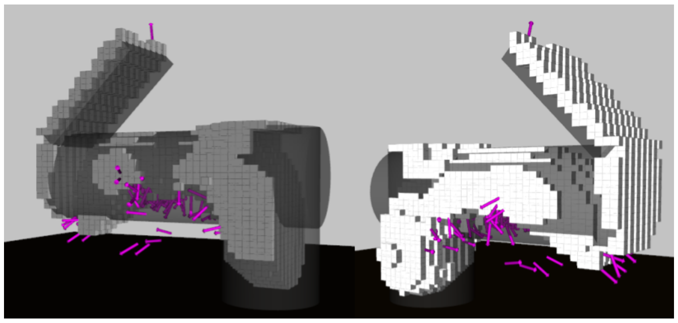

Fig. 4. Multiple views of the same voxelized GPIS. The surface corresponding to the table is not visualized. The arrows show the gradient of the surface.

stuck in a local minima, we consider the $\mathbb{R}^3$ position of the center of the rope to provide a more focused distance than a distance in the original $\mathbb{R}^{75}$ state space of the rope. We do not train a residual dynamics model online for TAMPC as the online data is insufficient for training a useful model for the high-dimensional state-action space. Our other baseline computes a signed distance field (SDF) of the environment generated from a partial view of the environment. This baseline uses MPPI with a horizon of 150 and 3000 samples.

Our results in Table I show that our method achieves higher success rates and shorter episode lengths over 30 trials. We also recover an approximation of the obstacle geometry in Fig. 4. We believe TAMPC's trap representation provides a sparser signal to the controller and struggles to cover the space of possible local minima induced by the hook. The non-adaptive baseline cannot reason about the occluded part of the obstacle, leading it to collide with the obstacle.

V. CONCLUSION

We propose a method for modeling *a priori* unknown obstacles to enable manipulation in novel environments. We achieve higher success rates and lower trajectory lengths across multiple tasks, including high-dimensional deformable object manipulation tasks. Our method leverages a novel fusion of visual and inferred contact information to model obstacles using a Gaussian process implicit surface, enabling data efficient obstacle modeling. We propose a novel MPC cost function that leverages GPIS gradients to guide the controller away from obstacles. Future work could include extending the GPIS to reason about environments where nominal dynamics are inaccurate for reasons other than obstacles.

References

[1] O. Williams and A. Fitzgibbon, “Gaussian process implicit surfaces,” in *Gaussian Processes in Practice*, 2006.

[2] P. Mitrano, A. LaGrassa, O. Kroemer, and D. Berenson, “Focused adaptation of dynamics models for deformable object manipulation,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5931–5937.

[3] C. Wang, Y. Zhang, X. Zhang, Z. Wu, X. Zhu, S. Jin, T. Tang, and M. Tomizuka, “Offline-online learning of deformation model for cable manipulation with graph neural networks,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5544–5551, 2022.

[4] A. L. LaGrassa and O. Kroemer, “Learning model preconditions for planning with multiple models,” in *Conference on Robot Learning*. PMLR, 2022, pp. 491–500.

[5] B. Lee, C. Zhang, Z. Huang, and D. D. Lee, “Online continuous mapping using gaussian process implicit surfaces,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6884–6890.

[6] G. Z. Gandler, C. H. Ek, M. Björkman, R. Stolkin, and Y. Bekiroglu, “Object shape estimation and modeling, based on sparse gaussian process implicit surfaces, combining visual data and tactile exploration,” *Robotics and Autonomous Systems*, vol. 126, p. 103433, 2020.

[7] S. Caccamo, Y. Bekiroglu, C. H. Ek, and D. Kragic, “Active exploration using gaussian random fields and gaussian process implicit surfaces,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 582–589.

[8] L. Liu, S. Fryc, L. Wu, T. L. Vu, G. Paul, and T. Vidal-Calleja, “Active and interactive mapping with dynamic gaussian process implicit surfaces for mobile manipulators,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3679–3686, 2021.

[9] S. Ottenhaus, D. Renninghoff, R. Grimm, F. Ferreira, and T. Asfour, “Visuo-haptic grasping of unknown objects based on gaussian process implicit surfaces and deep learning,” in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019, pp. 402–409.

[10] S. Ottenhaus, M. Miller, D. Schiebener, N. Vahrenkamp, and T. Asfour, “Local implicit surface estimation for haptic exploration,” in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 850–856.

[11] S. Dragiev, M. Toussaint, and M. Gienger, “Gaussian process implicit surfaces for shape estimation and grasping,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2845–2850.

[12] L. Wu, K. M. B. Lee, C. Le Gentil, and T. Vidal-Calleja, “Log-gpis-mop: A unified representation for mapping, odometry, and planning,” *IEEE Transactions on Robotics*, 2023.

[13] S. Zhong, Z. Zhang, N. Fazeli, and D. Berenson, “Tampc: A controller for escaping traps in novel environments,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1447–1454, 2021.

[14] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, “Information theoretic mpc for model-based reinforcement learning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.

[15] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, “Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration,” in *Advances in Neural Information Processing Systems*, 2018.

[16] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” http://pybullet.org, 2016–2021.

[17] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[18] S. R. Buss, “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods.”

[19] Y. Tassa, S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. P. Lillicrap, and N. Heess, “dm_control: Software and tasks for continuous control,” *CoRR*, vol. abs/2006.12983, 2020. [Online]. Available: https://arxiv.org/abs/2006.12983

[20] Y. Wang, D. McConachie, and D. Berenson, “Tracking partially-occluded deformable objects while enforcing geometric constraints,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 14 199–14 205.

Appendix A

	Peg-in-Hole	Sim. Rope
λ MPPI temperature	.01	.1
K MPPI samples	500	300
T MPPI horizon	T: 10, U: 15, I: 20	25
Σ MPPI noise	$\mathbf{diag}[.2_{\times 2}]$	$\mathbf{diag}[.001_{\times 6}]$
ν	1.5	.5
γ	1	1.5
δ	U: .25, I, T: 1	1
D_{max}	3000	3000
ζ	I, T:.85, U: .9	.75
α	.003	.0008
β	.6	.4
η	1	.1
C	1000	10
T_e	-	5
r_g	.02	.05
r_c	-	.01

TABLE II
Parameters