RUMI: Rummaging Using Mutual Information

Sheng Zhong[1], Nima Fazeli[1], and Dmitry Berenson[1]

***Abstract*—This paper presents Rummaging Using Mutual Information (RUMI), a method for online generation of robot action sequences to gather information about the pose of a known movable object in visually-occluded environments. Focusing on contact-rich rummaging, our approach leverages mutual information between the object pose distribution and robot trajectory for action planning. From an observed partial point cloud, RUMI deduces the compatible object pose distribution and approximates the mutual information of it with workspace occupancy in real time. Based on this, we develop an information gain cost function and a reachability cost function to keep the object within the robot's reach. These are integrated into a model predictive control (MPC) framework with a stochastic dynamics model, updating the pose distribution in a closed loop. Key contributions include a new belief framework for object pose estimation, an efficient information gain computation strategy, and a robust MPC-based control scheme. RUMI demonstrates superior performance in both simulated and real tasks compared to baseline methods.**

I. Introduction

Active exploration, the process of autonomously planning actions to gather more information about a target quantity, is a core problem in robotics, particularly when dealing with unknown environments [1]. This problem encompasses a range of scenarios, differentiated by the type of robot (e.g., mobile vs. stationary), the primary sensor modality (often vision), and the specific quantity to be estimated.

As robotics applications have transitioned from known, structured environments like factories to the unknown, dynamic environments of homes, new challenges have emerged. One critical application area is object manipulation, where visual perception is often hindered by occlusions caused by both the environment and the objects themselves [50]. To address these challenges, we focus on actively exploring to estimate the pose of a movable object with a known shape through contact-rich interactions, commonly referred to as rummaging.

Occlusions of the target object, both from itself and from other objects, motivate the need to use contact to determine the object's pose. Our prior work has investigated how to track the position of contact points during rummaging with an unknown number of objects [49], and how to estimate the plausible set of object poses given observed contact and free space points [50]. However, the problem of how to plan information-gathering trajectories to estimate a movable object's pose is still under-explored. A primary challenge is the object's mobility, coupled with the requirement for contact-based information collection.

This work was supported in part by the Office of Naval Research Grant N00014-24-1-2036 and NSF grants IIS-2113401 and IIS-2220876.
[1]Department of Robotics, University of Michigan, MI 48109, USA {zhsh, nfz, dmitryb}@umich.edu

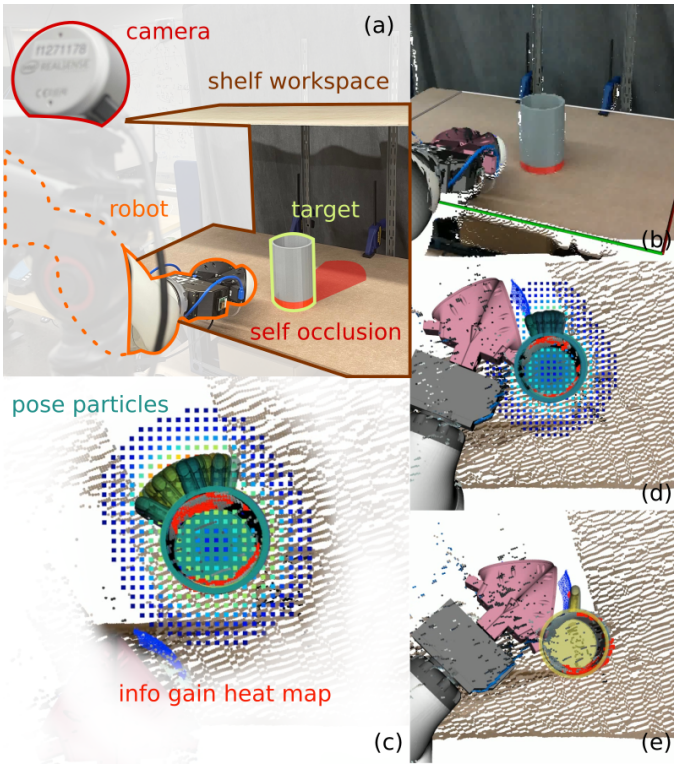


Fig. 1: (a) A real-world active exploration experiment where the goal is to estimate the pose of a movable mug. The mug pose is ambiguous due to self occlusion. (b) The initial point cloud view of the scene from the camera perspective. (c) Top-down view: RUMI maintains a belief over the mug's pose using a particle filter, where the pose particles are shown as overlaid objects with their relative likelihood indicated by color ranging from blue to yellow. Observed surface points are in red. From the pose particles and observations, RUMI generates an information gain field to plan over, shown as a heat map. Note the ambiguity and resulting information gain is concentrated on where the handle could be. (d) Even without making contact with the handle, the robot sweeps out free space eliminating pose particles with handles on the left side and constraining the pose. (e) Finally, making contact produces an accurate pose estimate.

Without careful planning, making contact can inadvertently push the object out of the robot's workspace, as evidenced in our experiments.

Active exploration is often framed from an information-theoretic perspective, where the quantity to be estimated is treated as a random variable, and actions are selected to minimize its uncertainty. This approach can be computationally expensive, necessitating a trade-off between accuracy

and speed or limiting the exploration to a single next best action. Additionally, some methods restrict the action space to movements along the object's surface [43], [11]. While this restriction simplifies the problem, it also limits the robot's capabilities. Instead, we aim to enable robots to make and break contact dynamically throughout the rummaging process, enhancing their exploratory capabilities.

To address the above challenges, we present Rummaging Using Mutual Information (**RUMI**), an active exploration method. Specifically, our contributions include:

1) a framework for creating and updating a belief over poses given observed point clouds, augmented with volumetric semantics such as whether each point is in free space or on the surface of the object, based on the discrepancy formulated in CHSEL [50]
2) a measure of information gain based on the mutual information between the object pose and volumetric semantics at the positions that the robot trajectory will cover, and show that it can be efficiently computed in parallel for dense workspace points in real time
3) a closed loop MPC planning framework using cost functions based on the information gain and maintaining object reachability, and a stochastic object dynamics model

In our experiments, we show that RUMI is the only method to achieve consistent success in simulated and real robot rummaging tasks across various objects.

II. Related Work

In a broad sense, we focus on the problem of actively exploring an unknown environment to reduce the uncertainty of some quantity. We specifically consider the distinction of active perception, or sensor path planning [37], [4], [1], in which a robot explores an environment without interacting and changing it, with interactive perception [2], in which the robot must change the environment to explore it. This paper focuses on the interactive perception problem of estimating the pose of a movable rigid object with a known shape using a robot arm. The environments we are interested in include cabinets and other tight spaces which limit the ability of the robot arm to grasp top-down. Prior work considering uncertainty in similar environments [38] assumed a static environment while we allow for movable objects.

Purely vision-based approaches in these environments suffer from occlusion, a restricted field of view, and the close range of the robot wrist to objects during the course of rummaging (making it difficult to use a wrist-camera). Additionally, we consider tasks that require physically moving other objects in order to access the target object in Section V-J.

In general, active exploration is the iterative process of:

1) forming a belief over state given observations
2) computing expected information gain over a workspace
3) planning an action sequence
4) executing some of the action sequence and collecting observations

A. Representing Belief

Representations suitable for active exploration have been studied extensively. In many cases, parametric filters like the extended Kalman Filter (EKF) [39], [24] may be used when the posterior of the quantity of measure should be approximately Gaussian. Otherwise, non-parametric methods like particle filters [9], [18] are often used. Occupancy grids have also been popular, e.g. used in the simultaneous localization and mapping (SLAM) variant of active exploration [31], [45], [7]. In particular, when assuming each grid cell is independent, information gain based on the entropy of all the cells may be efficiently computed on an occupancy grid. We make a similar assumption that enables efficient computation of our information gain.

Recently, Gaussian processes (GPs) [16] have also been used for estimating object shape. GP implicit surfaces (GPIS) have shown strong representation power [10], [11]. GPIS uses a GP to output a field in which the 0-level set represents the surface of the object. In our method, we do not need the full representation power of a GP since we have a known object shape. Instead, we use a particle filter to represent the pose distribution, and present a novel way to evaluate the particle probabilities given an observed point cloud.

B. Information Gain

The information gain can be formulated in many ways, often depending on the belief representation. For GPIS the variance of the GP [11], or the differential entropy of the GP for adding a new data point [12] can be evaluated directly and used. However, despite work on geometric shape priors for GPIS [27], there remains no satisfactory way to condition a GP on a known shape with unknown pose. We implement a GPIS baseline and condition it on the shape by augmenting the input data. Mutual information between observations and the estimated quantity is also common [16], [26], which measures the reduction in uncertainty of the estimated quantity given the observations. Thus, we formulate our information gain function based on the mutual information between the object pose and the occupancy at points a robot trajectory would sweep out.

C. Planning

Searching for an optimally-informative trajectory is usually computationally intensive. GP-based methods in particular are limited by inference times that grow rapidly with increasing number of data points, often addressed by using sparse GPs or downsampling to trade off accuracy [41]. Some methods greedily selects the optimal next configuration, and additionally constrain the action space to slide along the surface of the object [43], [11]. Our formulation of the information gain allows us to efficiently evaluate it for many query trajectories in parallel, enabling us to use longer-horizon planning methods such as sampling-based model predictive control in a closed loop. We consider difficult tasks which necessitate long horizon planning.

Active exploration problems also differs by sensing modality. In the context of object shape and pose estimation, the

most common modality is visual perception, with the common framing of the problem as finding the next best view [19]. Tactile approaches have also demonstrated success [47], [11], as well as hybrid approaches [36], [40]. Tightly coupled with sensing modality is the distinction of whether the robot is passively observing the environment or actively interacting with and changing the environment as in the interactive perception problem [2]. RUMI is a hybrid approach for interactive perception, primarily relying on contact-rich interactions using tactile sensors, but also leveraging visual perception to initialize pose estimates. Unlike most other methods for object pose or shape estimation, we do not assume the object is stationary, which accounts for a large part of the difficulty. We additionally consider an unknown number of unknown objects cluttered around the target object in Section V-J. The closest method to ours is Act-VH [36], which trains an implicit surface neural network to output hypothesis voxel grids of seen objects given a partially observed point cloud and selects the best point to probe next. One major weakness of this method is the need to either retrain their network on all candidate objects whenever there is a new target object, or to train a network per object and assume object identity is known. Our method can be applied to new known objects without any training. Additionally, their object is in between the robot and the camera, meaning that the visually-occluded region is highly reachable, bypassing a major challenge that we address. Lastly, we consider the information gain from full robot trajectories rather than a single next point to probe.

III. Problem Statement

Let $\mathbf{q} \in \mathbb{R}^{N_q}$ denote the robot configuration, and $\mathbf{u} \in \mathbb{R}^{N_u}$ denote control. We study a single robot exploring an unmodeled environment, using limited visual perception and contact-heavy rummaging to estimate the pose of a single movable rigid target object of known shape. A rigid object's configuration is defined by its pose, a transform $\mathbf{T} \in \text{SE(3)}$. Every $\mathbf{T}$ can be identified with a $\mathbb{R}^{4\times4}$ homogeneous transformation matrix, and for convenience, we use $\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x}$ to denote the homogeneous transform of point $\mathbf{x} \in \mathbb{R}^3$ from world frame coordinates to the object frame of $\mathbf{T}$ (homogeneous coordinates have 1 appended). There is an underlying dynamics function $f : \mathbb{R}^{N_q} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_q}$ that we do not know, but are given the free space dynamics function $f_f : \mathbb{R}^{N_q} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_q}$. The difference in dynamics is primarily due to contact between the robot and the target object. We are interested in generating a fixed length trajectory of T actions, $\mathbf{u}_1, ..., \mathbf{u}_T$ to actively explore and estimate the target object's pose.

Specifically, we have the target object's precomputed object frame signed distance function (SDF) derived from its 3D model, $\texttt{sdf} : \mathbb{R}^3 \to \mathbb{R}$. After each action, sensors observe a set of points at time $t : \mathcal{X}'_t = \{(\mathbf{x}_1, s_1), ..., (\mathbf{x}_N, s_N)\}_t$ with world positions $\mathbf{x}_n \in \mathbb{R}^3$ and semantics s_n (described below). For convenience, we call each pair $(\mathbf{x}, s)$ a *geometric feature*. Let $\mathcal{X}_t$ denote the accumulated set of geometric features up to and including time t. Sensors may include but are not limited to robot proprioception, end-effector mounted tactile sensors, and external cameras, but all must produce outputs convertible to geometric features.

We treat the pose of the target object as a random variable and define $p(\mathbf{T}|\mathcal{X}_t)$ as the posterior probability distribution over poses given $\mathcal{X}_t$. Observation noise, object symmetry, and the partial nature of $\mathcal{X}_t$ results in pose uncertainty.

Let $\mathbf{T}^*$ be the true object transform, then the observed semantics are

$$s_n = \begin{cases} \texttt{free} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) > 0 \\ \texttt{occupied} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) < 0 \\ \texttt{surface} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) = 0 \end{cases}$$

For a workspace point $\mathbf{x}$ that we have not observed, its semantics is a discrete random variable $S_\mathbf{x}$ with the shorthand $p(S_\mathbf{x}) = p(s|\mathbf{x})$. We are given a sensor model $p(S_\mathbf{x}|\mathbf{T}) = p(S_\mathbf{x}|\texttt{sdf}(\mathbf{T}\mathbf{x}))$ such as in Fig. 2 that gives the probability of observing each s value given a SDF value. The sensor model does not consider uncertainty over the position, and we assume we are given exact positions with only uncertainty over semantics $S_\mathbf{x}$.

Given a prior pose distribution $p(\mathbf{T})$, for example knowing the object is upright but having a uniform distribution on its orientation and position over the surface of the workspace, and starting at $\mathbf{q}_1$, our goal is to estimate the pose of the object by maximizing the expected information gain after T actions:

$$\begin{aligned} \underset{\mathbf{u}_1,...,\mathbf{u}_T}{\arg\max} \quad & \mathbb{E}_{\mathcal{X}_T}[D_{KL}(p(\mathbf{T}|\mathcal{X}_T)||p(\mathbf{T}))] \\ \text{s.t.} \quad & \mathbf{q}_{t+1} = f(\mathbf{q}_t, \mathbf{u}_t),\ t = 1, ..., T \end{aligned} \tag{1}$$

The expectation is over the semantics of each position in $\mathcal{X}_T$. Note that this is equivalent to the mutual information between $\mathbf{T}$ and $\mathcal{X}_T$, $I(\mathbf{T}; \mathcal{X}_T)$ [33].

We consider environments where there is initial occlusion of the target object by the environment, other objects, or the object itself. We have the model of the static environment, but do not know how many objects there are. The challenge of this problem comes from the need for contact-based perception due to limited sensing capabilities, coupled with the fact that the target object is movable. Moreover, an ineffective action sequence can result in undesirable contacts, potentially pushing the object out of the robot's reachable workspace. We assume the robot impedance controller would not produce sufficient forces as to violate the rigid-body assumption.

We evaluate the quality of the estimated pose distribution by evaluating the likelihood of the ground truth pose $\mathcal{L}(\mathbf{T}^*|\mathcal{X}_t)$, or equivalently its *negative log likelihood* (NLL). Low NLL indicates both certainty and correctness of the pose distribution. We do so by sampling a set of surface points in the object frame and transforming them by $\mathbf{T}^*$ to produce world positions $\mathbf{X}$. We then evaluate the NLL of all of the points having `surface` semantics:

$$nll(\mathcal{X}) = -\log p(\bigcap_{\mathbf{x}\in\mathbf{X}} S_\mathbf{x} = \texttt{surface}|\mathcal{X}) \tag{2}$$

We use this metric as well as computational efficiency to evaluate our method against baselines and ablations.

IV. Method

Our high level approach to addressing the problem in Eq. 1 is depicted in Fig. 3. We represent the pose posterior $p(\mathbf{T}|\mathcal{X})$

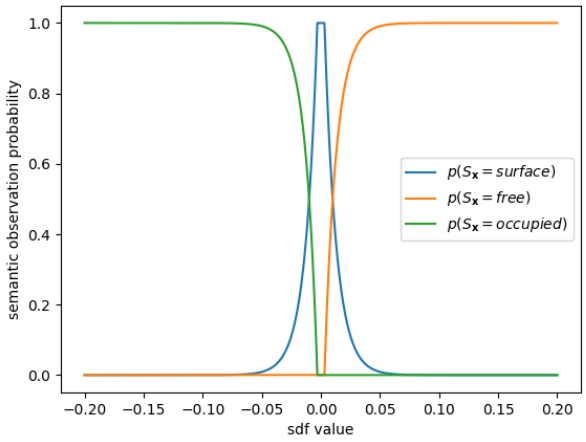


Fig. 2: Example sensor model that gives a probability of observing each semantics class given a SDF value.

with a particle filter and describe how to evaluate $p(\mathbf{T}|\mathcal{X})$. Next, we present a tractable surrogate for information gain that we develop into a cost function for model predictive control (MPC). To discourage trajectories that move the target object out of the robot's reachable area, we develop an additional reachability cost function. Furthermore, to estimate the displacement of the target object given an action trajectory, we implement a stochastic dynamics model $\hat{f}$. We use the cost functions and the dynamics function inside MPC, which executes in a closed loop for T steps. During this process, we detail how to merge current observations with previous ones and update the pose posterior $p(\mathbf{T}|\mathcal{X})$.

A. Representing Pose Posterior

We maintain a belief over the pose posterior $p(\mathbf{T}|\mathcal{X}_t)$ using a particle filter, where each particle is a pose. We have P particles $\mathbf{T}_{1..P}$, with weights $w_{1..P}$ such that $\sum_{i=1}^{P} w_i = 1$. Our choice of a particle filter over alternative representations is motivated by the potential multi-modality of the posterior and the ability to process each particle in parallel.

A major obstacle to the tractability of solving Eq. 1 is the information correlation between geometric features. Observing one decreases the information gain from others in a non-trivial manner, and it is a common long-standing assumption to consider the information gain from each independently [6], [42]. Thus, we assume the conditional mutual independence of $S_\mathbf{x}$ for all query positions $\mathbf{x}$ given observed $\mathcal{X}_t$.

Critical to our method is a way to evaluate the posterior $p(\mathbf{T}|\mathcal{X})$. Our prior work CHSEL [50] formulated a differentiable cost function $\hat{C}(\mathcal{X}, \mathbf{T})$ that evaluates the discrepancy between $\mathcal{X}$ and $\mathbf{T}$. It bears similarity to hydroelastic, or pressure field contact modelling [13], [28], except in addition to the pressure field penalizing object penetration, there are pressure fields that penalize semantics violation, such as observed free space geometric features being inside objects.

We simplify the third semantics class from CHSEL, which represented known SDF of any value. We restrict it to $s = 0$, which refers to surface points. The cost is formulated by first partitioning the observed $\mathcal{X}$ into $\mathcal{X}_f = \{(\mathbf{x}, s) \mid s = \texttt{free}\}$, $\mathcal{X}_o = \{(\mathbf{x}, s) \mid s = \texttt{occupied}\}$, and $\mathcal{X}_s = \{(\mathbf{x}, s) \mid s = \texttt{surface}\}$.

$$\hat{C}(\mathcal{X}, \mathbf{T}) = \sum_{\mathbf{x}, s \in \mathcal{X}_f} \hat{c}_f(\tilde{\mathbf{x}}) + \sum_{\mathbf{x}, s \in \mathcal{X}_o} \hat{c}_o(\tilde{\mathbf{x}}) + \sum_{\mathbf{x}, s \in \mathcal{X}_s} \hat{c}_k(\tilde{\mathbf{x}}) \tag{3}$$

$$\hat{c}_f(\tilde{\mathbf{x}}) = C \max(0, \alpha - \texttt{sdf}(\tilde{\mathbf{x}})) \tag{4}$$

$$\hat{c}_o(\tilde{\mathbf{x}}) = C \max(0, \alpha + \texttt{sdf}(\tilde{\mathbf{x}})) \tag{5}$$

$$\hat{c}_k(\tilde{\mathbf{x}}) = |\texttt{sdf}(\tilde{\mathbf{x}})| \tag{6}$$

where $C > 0$ is a scaling parameter and $\alpha > 0$ allows for small degrees of violation due to uncertainty in the positions.

Their gradients are defined as

$$\nabla \hat{c}_f(\tilde{\mathbf{x}}) = C \max(0, \alpha - \texttt{sdf}(\tilde{\mathbf{x}}))(-\nabla \texttt{sdf}(\tilde{\mathbf{x}})) \tag{7}$$

$$\nabla \hat{c}_o(\tilde{\mathbf{x}}) = C \max(0, \alpha + \texttt{sdf}(\tilde{\mathbf{x}})) \nabla \texttt{sdf}(\tilde{\mathbf{x}}) \tag{8}$$

$$\nabla \hat{c}_k(\tilde{\mathbf{x}}) = \texttt{sdf}(\tilde{\mathbf{x}}) \nabla \texttt{sdf}(\tilde{\mathbf{x}}) \tag{9}$$

where $\nabla \texttt{sdf}(\tilde{\mathbf{x}})$ is the object SDF gradient with respect to an object-frame position $\tilde{\mathbf{x}}$ and normalized such that $||\nabla \texttt{sdf}(\tilde{\mathbf{x}})||_2 = 1$.

Similar to energy-based methods, we use the Boltzmann distribution [15], [44] to interpret Eq. 3 as the posterior pose probability:

$$p(\mathbf{T}|\mathcal{X}) = \eta e^{-\lambda \hat{C}(\mathcal{X}, \mathbf{T})} \tag{10}$$

where $\lambda > 0$ selects how peaky the distribution should be and η is the normalization constant such that $\int \eta e^{-\lambda \hat{C}(\mathcal{X}, \mathbf{T})} d\mathbf{T} = 1$.

We observe that the cost in Eq. 3 is additive in the sense

$$\hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \mathbf{T}) = \hat{C}(\mathcal{X}, \mathbf{T}) + \hat{C}((\mathbf{x}, s), \mathbf{T}) \tag{11}$$

This is an important property that enables us to efficiently evaluate information gain of all workspace positions in parallel.

B. Mutual Information Surrogate

Our conditional mutual independence assumption of $p(S_\mathbf{x}|\mathcal{X})$ lets us consider the information gain from knowing the semantics at a single new position, which we denote the *information gain field* $\tilde{I}(\mathbf{x}|\mathcal{X})$. This is much simpler than considering the information gain of a robot trajectory directly because there is no time component or correlation between the semantics of neighbouring geometric features. Suppose we have observed $\mathcal{X}$ and want to evaluate the information gain from observing some new geometric feature $(\mathbf{x}, s)$. Note that here we are querying a specific given value of $\mathbf{x}$, but $S_\mathbf{x}$ is still a random variable, so the expectation is over $p(S_\mathbf{x}|\mathcal{X})$:

$$I_f(\mathbf{x}|\mathcal{X}) = \mathop{\mathbb{E}}_{s \sim p(S_\mathbf{x}|\mathcal{X})} [D_{KL}(p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s)) || p(\mathbf{T}|\mathcal{X}))] \tag{12}$$

$$= \mathop{\mathbb{E}}_{s} [\mathop{\mathbb{E}}_{\mathbf{T} \sim p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))} [\log \frac{p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))}{p(\mathbf{T}|\mathcal{X})}]] \tag{13}$$

The forward KL divergence results in an expectation over $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$. Since we need to evaluate the information gain for many positions in the workspace, this becomes intractable.

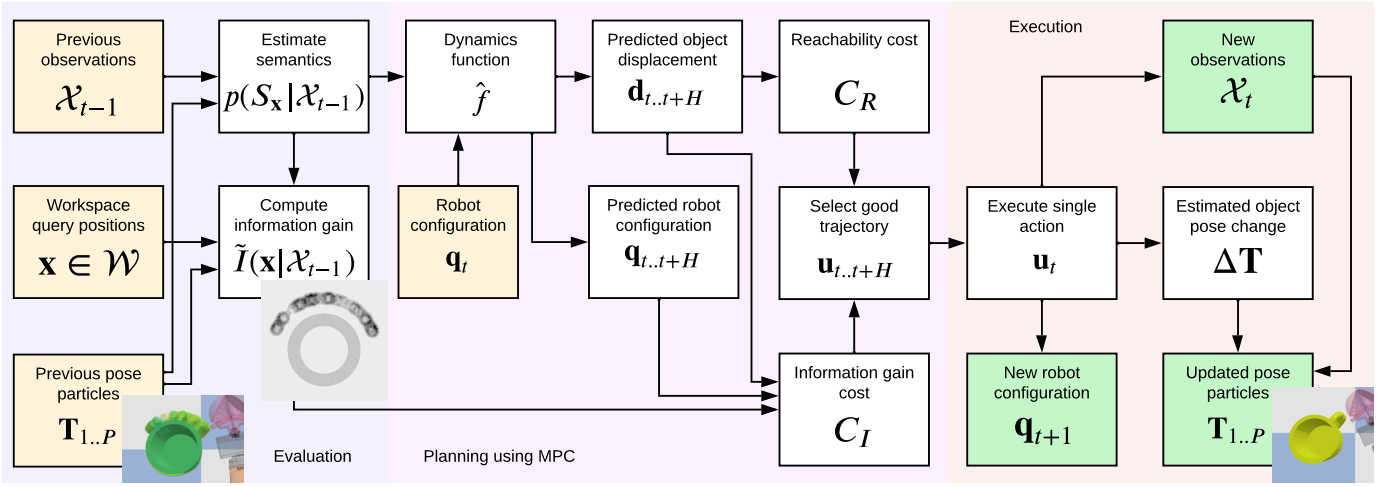


Fig. 3: Flow chart showing one time step of RUMI's approach for solving Eq. 1. Beige blocks are inputs to this time step while green ones are outputs of this time step. The process is also into evaluating current information gain, planning into the future using the information gain, and executing one step of the plan and updating observations.

To address this challenge, we use the reverse KL divergence, since the expectation is then over $p(\mathbf{T}|\mathcal{X})$ for all queried positions. In general, KL divergence is not symmetric. However, when two distributions are close together the KL divergence is approximately symmetric [48], [20]. In our case the KL divergence is between $p(\mathbf{T}|\mathcal{X})$ and $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$ with all having SE(3) support, avoiding infinite divergences. As we increase $|\mathcal{X}|$ during exploration, we expect the two distributions to become closer and the reverse KL to better approximate the forward KL divergence.

Intuitively, a geometric feature has high reverse KL divergence if it has high $p(\mathbf{T}|\mathcal{X})$ and low $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$. These correspond to geometric features that would invalidate currently high-probability poses i.e. these are positions we would like to explore.

Using reverse KL, We now have

$$I_r(\mathbf{x}|\mathcal{X}) = \mathop{\mathbb{E}}_{s}[D_{KL}(p(\mathbf{T}|\mathcal{X})||p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s)))] \tag{14}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}\sim p(\mathbf{T}|\mathcal{X})}[\log \frac{p(\mathbf{T}|\mathcal{X})}{p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))}]] \tag{15}$$

Substituting Eq. 10 in

$$I_r(\mathbf{x}|\mathcal{X}) = \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\log \frac{p(\mathbf{T}|\mathcal{X})}{p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))}]] \tag{16}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\log \frac{\eta_1 e^{-\lambda \hat{C}(\mathcal{X}, \mathbf{T})}}{\eta_2 e^{-\lambda \hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \mathbf{T})}}]] \tag{17}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\log \frac{e^{-\lambda \hat{C}(\mathcal{X}, \mathbf{T})}}{e^{-\lambda \hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \mathbf{T})}}]] + \log \frac{\eta_1}{\eta_2} \tag{18}$$

$$= \lambda \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[-\hat{C}(\mathcal{X}, \mathbf{T}) + \hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \mathbf{T}))] + \log \frac{\eta_1}{\eta_2} \tag{19}$$

where η_1 and η_2 are the normalizing constants for $p(\mathbf{T}|\mathcal{X})$ and $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$, respectively. First we simplify using the additive property of $\hat{C}$ (Eq. 11) then consider the normalizing constants,

$$I_r(\mathbf{x}|\mathcal{X}) = \lambda \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\hat{C}((\mathbf{x}, s), \mathbf{T})]] + \log \frac{\eta_1}{\eta_2} \tag{20}$$

We note that η_2 depends on the querying position $\mathbf{x}$ because each $\mathbf{x}$ induces a different $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$. This normalizing constant is intractable to compute because it involves an integral over $\mathbf{T}$, so we instead optimize the approximation

$$\tilde{I}(\mathbf{x}|\mathcal{X}) = \lambda \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\hat{C}((\mathbf{x}, s), \mathbf{T})]] \tag{21}$$

$$= \lambda \sum_{s} p(S_{\mathbf{x}} = s|\mathcal{X}) \mathop{\mathbb{E}}_{\mathbf{T}}[\hat{C}((\mathbf{x}, s), \mathbf{T})] \tag{22}$$

Selecting λ too high leads to the pose particle weights dominated by a few, causing particle degeneracy.

We approximate the expectation over the posterior by taking the weighted sum over the pose particles

$$\tilde{I}(\mathbf{x}|\mathcal{X}) \approx \lambda \sum_{s} \sum_{i=1}^{P} p(S_{\mathbf{x}} = s|\mathcal{X}) w_i \hat{C}((\mathbf{x}, s), \mathbf{T}_i) \tag{23}$$

Finally, we consider how we can approximate the conditional semantics distribution $p(S_{\mathbf{x}}|\mathcal{X})$ which is the last term required for fully computing $\tilde{I}(\mathbf{x}|\mathcal{X})$. We use the law of total probability

$$p(S_{\mathbf{x}}|\mathcal{X}) = \int p(S_{\mathbf{x}}|\mathbf{T}, \mathcal{X}) p(\mathbf{T}|\mathcal{X}) d\mathbf{T} \tag{24}$$

Here again we approximate the expectation over the posterior by taking the weighted sum over the pose particles

$$p(S_{\mathbf{x}}|\mathcal{X}) \approx \sum_{i=1}^{P} w_i p(S_{\mathbf{x}}|\mathbf{T}_i, \mathcal{X}) \tag{25}$$

we assume the conditional independence of $S_{\mathbf{x}}$ and $\mathcal{X}$ when

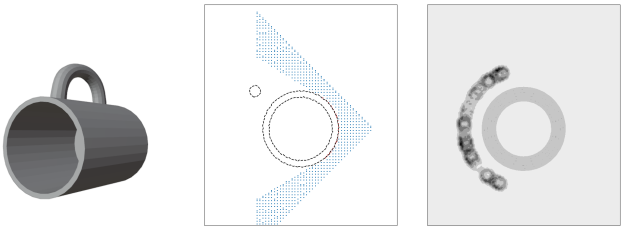


Fig. 4: (left) Example mug object with (middle) $\mathcal{X}$ rendered from a pinhole camera on one side, not seeing where the handle is. The ground truth object surface is outlined in dotted black, observed surface geometric features are in red, and observed free space is in blue. (right) The information gain field estimated with $P = 100$ is darker where there is more information, where the handles could be.

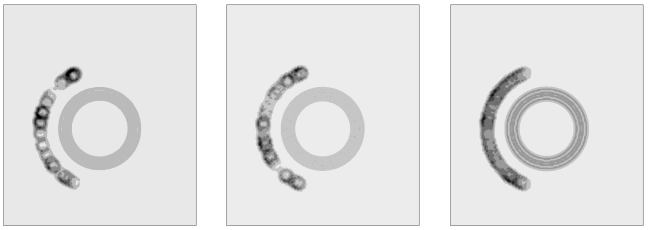


Fig. 5: Computed information gain field $\tilde{I}(\mathbf{x}|\mathcal{X})$ from Fig. 4 with (left) $P = 30$, (middle) $P = 100$, and (right) $P = 1000$.

given $\mathbf{T}$, so

$$p(S_{\mathbf{x}}|\mathcal{X}) \approx \sum_{i=1}^{P} w_i p(S_{\mathbf{x}}|\mathbf{T}_i)) \tag{26}$$

$$= \sum_{i=1}^{P} w_i p(S_{\mathbf{x}}|\texttt{sdf}(\mathbf{T}_i\mathbf{x})) \tag{27}$$

where $p(S_{\mathbf{x}}|\texttt{sdf}(\mathbf{T}_i\mathbf{x}))$ is given by the sensor model.

Note that all the terms in Eq. 23 only query $\mathbf{x}$ and $\mathbf{T}_{1..P}$, without needing to directly consider $\mathcal{X} \cup (\mathbf{x}, s)$. This enables us to evaluate $\tilde{I}(\mathbf{x}|\mathcal{X})$ for all positions inside a workspace $\mathbf{x} \in \mathcal{W} \subset \mathbb{R}^3$ in parallel.

C. Illustrative Example

To develop intuition, we consider a mug as the target object, depicted in Fig. 4 (left). Initially, a camera observes one side of the mug, narrowing down its position. However, since it cannot observe the handle and there is partial rotational symmetry, there is uncertainty in the orientation of the object. Fig. 4 (right) is the computed information gain $\tilde{I}(\mathbf{x}|\mathcal{X})$ over the entire workspace, showing that most of the information gain is concentrated where the handle could be.

Intuitively, we expect a smooth dark band where the handles could be but observe unevenness. This is due to the approximation error of $p(\mathbf{T}|\mathcal{X})$ being represented by finitely many pose particles and the approximation of I_r with $\tilde{I}$. This is illustrated by Fig. 5. With larger P, the trade off for gaining a more accurate approximation of $p(\mathbf{T}|\mathcal{X})$ is increased memory usage. Since we process the particles and query positions in parallel, memory becomes the bottleneck and they have to be processed in batches, turning the memory trade off into a runtime one.

D. Posterior Update Process

So far, we have developed the information gain field given some observed $\mathcal{X}$ at one time step. We now describe the active rummaging process in Algorithm 1 to update the posterior.

Before any actions, we are given the pose prior $p(\mathbf{T})$ from which we sample P initial poses $\mathbf{T}_{0,1..P}$. Note that given a fixed set of geometric features $\mathcal{X}$, the posterior probability of poses can be compared using Eq. 10. With the relative posterior probability and samples from the prior, we can theoretically draw samples from the posterior using techniques such as Markov Chain Monte Carlo (MCMC) [14], [8]. However, MCMC tend to struggle with the high dimensionality of poses ($\mathbf{T} \in$ SE(3)). With the interpretation of Eq. 3 as the posterior (Eq. 10), optimization of Eq. 3 on prior pose particles can naturally be interpreted as approximately sampling from the posterior. Thus we apply CHSEL (Algorithm 1 from [50]) to produce the initial pose particles in Algorithm 1 line 2.

Algorithm 1: Particle filter posterior update

Given: P number of particles,
$\mathbf{T}_{0,1..P}$ initial poses,
$\mathbf{q}_1$ initial robot configuration,
σ_t translation noise,
σ_R rotation noise,
l_r resample discrepancy threshold,
$\mathcal{W}$ workspace set of query positions

1 $\mathcal{X}_0 \leftarrow$ sensors observe at $\mathbf{q}_1$
2 $\mathbf{T}_{1..P} \leftarrow \texttt{CHSEL}(\mathbf{T}_{0,1..P}, \mathcal{X}_0)$
3 $w_{1..P} \leftarrow \texttt{WeighParticles}(\mathbf{T}_{1..P}, \mathcal{X}_0)$
4 **for** $t \leftarrow 1$ **to** T **do**
5 compute $p(S_{\mathbf{x}}|\mathcal{X}_{t-1})$ using Eq. 27 and $\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1})$ using Eq. 23 for $\mathbf{x} \in \mathcal{W}$ and cache in voxel grids
6 $\mathbf{u}_t \leftarrow \texttt{Plan}(\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1}), p(S_{\mathbf{x}}|\mathcal{X}_{t-1}), \mathbf{q}_t)$
7 robot executes action $\mathbf{u}_t$ to arrive at $\mathbf{q}_{t+1}$
8 $\mathcal{X}'_t \leftarrow$ sensors observe at $\mathbf{q}_{t+1}$
9 $\Delta\mathbf{T}_r \leftarrow$ sensors observe change in robot end-effector pose while in contact
10 $\Delta\mathbf{T}, \Delta\mathbf{T}_w \leftarrow \texttt{ObjMove}(\mathcal{X}_{t-1}, \mathcal{X}'_t, \mathbf{T}_{1..P}, \Delta\mathbf{T}_r)$
11 **if** ΔT *not* **0 then**
// predict step
12 **for** $i \leftarrow 1$ **to** P **do**
13 $\Delta\mathbf{T}_\sigma \leftarrow \texttt{PerturbTransform}(\sigma_t, \sigma_R)$
$\mathbf{T}_i \leftarrow \Delta\mathbf{T}_\sigma \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$
14 $\mathcal{X}_t \leftarrow \texttt{MergeObs}(\mathcal{X}_{t-1}, \mathcal{X}'_t, \mathbf{T}_{1..P}, \Delta\mathbf{T}_w)$
// update step, even when not in contact
15 $l_{1..P} \leftarrow \hat{C}(\mathcal{X}, \mathbf{T}_{1..P})$
16 **if** $\max(l_{1..P}) > l_r$ **then**
17 $\mathbf{T}_{1..P} \leftarrow \texttt{Resample}(\mathbf{T}_{1..P}, w_{1..P}, \mathcal{X}_t)$
18 $w_{1..P} \leftarrow 1/P$
19 **else**
20 $w_{1..P} \leftarrow \texttt{WeighParticles}(\mathbf{T}_{1..P}, \mathcal{X}_t)$

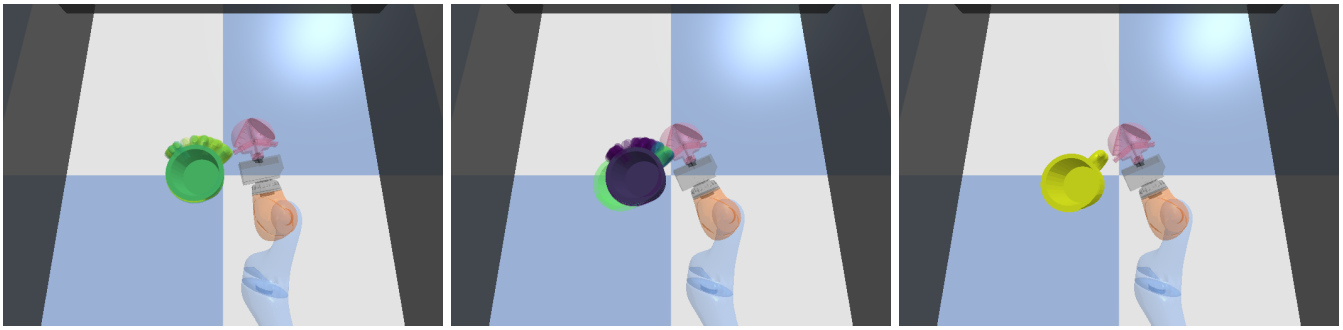


Fig. 6: Resampling during a pybullet simulated mug task with partial initial observation like in Fig. 4. (left) Before contact pose particles, and (middle) pose particles after contact with many receiving low likelihood indicated by the dark color due to discrepancy with the newly observed surface geometric features. This leads to resampling, and (right) resampled particles that are all high likelihood and centered around the ground truth pose.

Algorithm 2: `WeighParticles`

Given: $\mathbf{T}_{1..P}$ pose particles,
$\mathcal{X}$ set of observed geometric features,
λ peakiness

1 $l_{1..P} \leftarrow \hat{C}(\mathcal{X}, \mathbf{T}_{1..P})$
2 $w_{1..P} \leftarrow e^{-\lambda l_{1..P}}$
3 $w_{1..P} \leftarrow w_{1..P} / \sum_{i=1}^{P} w_i$ `// normalize sum to 1`

CHSEL performs Quality Diversity (QD) optimization [34] on Eq. 3 to find poses that have low discrepancy while maintaining diversity across some measure of pose space. We use the orientation component of $\mathbf{T}$, or just the yaw when restricting the pose search space to SE(2) as the measure.

We then assign weights to each pose particle as described in Algorithm 2. These weights represent the relative posterior probability of each particle. We normalize the weights so that $\sum_{i=1}^{P} w_i = 1$. Normalizing is important so that the use of weights in approximating expectations over the pose posterior in Eq. 23 and Eq. 25 remain valid. A side benefit of normalization is that we can omit the normalizing constant η from Eq. 10 in Algorithm 2 line 2.

Then for each time step t, we first compute $p(S_{\mathbf{x}}|\mathcal{X}_{t-1})$ and $\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1})$ using Eq. 27 and Eq. 23, respectively, for $\mathbf{x} \in \mathcal{W}$ and cache the results in voxel grids. These voxel grids allow linear interpolation querying and return 0 for $\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1})$ and `free` for $p(S_{\mathbf{x}}|\mathcal{X}_{t-1})$ when $\mathbf{x}$ is outside $\mathcal{W}$. They are used to plan a robot trajectory, as described in Subsection IV-E. The robot executes the first action in the planned trajectory and sensors observe both a new set of geometric features $\mathcal{X}'_t$ and the change in robot end effector pose while in contact $\Delta\mathbf{T}_r$.

In Algorithm 1 line 10 we estimate the change in pose $\Delta\mathbf{T}$ of the target object given $\mathcal{X}_{t-1}$, $\mathcal{X}'_t$, $\mathbf{T}_{1..P}$, and $\Delta\mathbf{T}_r$. Some end-effectors can either enforce sticking contact [17] or measure slip (such as in [30], [35]) to estimate $\Delta\mathbf{T}$ directly. Not all robots have these sensors, so we present an optimization based method in Algorithm 5. The main idea is to find a $\Delta\mathbf{T}$ that transforms $\mathcal{X}_{t-1}$ such that it is consistent with the most recently observed $\mathcal{X}'_t$. Our prior is that contact was sticking; that is $\Delta\mathbf{T} = \Delta\mathbf{T}_r$ in Algorithm 5 line 1. We select a representative pose particle $\mathbf{T}_i$ with the lowest discrepancy to apply $\Delta\mathbf{T}$ to. For N_o optimization steps, we evaluate $\hat{C}$ on $\mathcal{X}'$ and the hypothesis new pose $\Delta\mathbf{T} \cdot \mathbf{T}_i$. $\hat{C}$ is differentiable with respect to $\Delta\mathbf{T} \cdot \mathbf{T}_i$, and we back propagate gradients to $\Delta\mathbf{T}$ and perform stochastic gradient descent (SGD). We then produce the world frame change in pose $\Delta\mathbf{T}_w = \mathbf{T}_i^{-1} \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$ that can be applied to world frame positions.

Typically in particle filters we update the posterior via alternating prediction (via forward dynamics) and correction (from sensor data) steps. If the object did not move, then we also predict the pose particles remain stationary. If the object did move ($\Delta\mathbf{T}$ not $\mathbf{0}$) then our forward dynamics predicts movement $\mathbf{T}_i \leftarrow \Delta\mathbf{T}_\sigma \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$, where $\Delta\mathbf{T}_\sigma$ is a transform perturbation sampled with the process in Algorithm 3 that adds diversity to the particles.

Before we can perform the correction step, we first merge the previous observations $\mathcal{X}_{t-1}$ with the current observations $\mathcal{X}'_t$, as described in Algorithm 6. The object geometric features are transformed by $\Delta\mathbf{T}_w$ while the free geometric features remain stationary. However, the move might have invalidated some previous free ones and so we check whether $\texttt{sdf}(\mathbf{T}_i\mathbf{x}) > 0,\ \forall i = 1..P$ for each $(\mathbf{x}, \texttt{free}) \in \mathcal{X}$ in Algorithm 6 line 3. We then take the union of the transformed $\mathcal{X}_o$, validated $\mathcal{X}'_f$, and newly observed $\mathcal{X}'$. To avoid duplicate data, we voxel downsample by creating voxel grids, one per semantics value, that spans the range of the positions with resolution r_d. We assign the voxel grids with the positions then extract the center of voxel cells that received any assignment as new positions. We denote this downsampling process as $D : \mathbb{R}^3 \times \mathbb{R}^+ \to \mathbb{R}^3$.

With updated observations $\mathcal{X}_t$, we can update the weights of the pose particles. Importantly, we update even when not making contact because observing $s = \texttt{free}$ geometric features provides information about where the object is not. This process is described in Algorithm 2, where Eq. 3 is applied to get discrepancies $l_{1..P}$. We then apply Eq. 10 to convert it to an unnormalized probability. For numerical stability, we subtract the minimum l from all of them to get relative discrepancy. This is without loss of generality since the normalization forces the weights to sum to 1.

In addition to the update step, we resample the pose particles to avoid degeneracy and maintain diversity as is typical of particle filters. Many heuristics exist for deciding when to

Algorithm 3: `PerturbTransform`

Given: σ_t translation noise, σ_R rotation noise
Output: $\Delta\mathbf{T}_\sigma$ delta transformation

// sample process noise
$\boldsymbol{\Delta t} \sim \mathcal{N}(\mathbf{0}, \mathbf{diag}([\sigma_t, \sigma_t, \sigma_t])$
$\theta \sim \mathcal{N}(0, \sigma_R)$
$\mathbf{e} \sim U(\{\mathbf{x} \mid ||\mathbf{x}||_2 = 1, \mathbf{x} \in \mathbb{R}^3\})$
$\boldsymbol{\Delta R} \leftarrow e^{\theta \mathbf{e}}$ // axis angle to matrix
$\Delta\mathbf{T}_\sigma \leftarrow \begin{bmatrix} \boldsymbol{\Delta t} & \boldsymbol{\Delta R} \\ \mathbf{0} & 1 \end{bmatrix}$

Algorithm 4: `Resample`

Given: $\mathbf{T}_{1..P}$ pose particles,
$w_{1..P}$ particle weights,
$\mathcal{X}$ set of observed geometric features,
σ_t translation noise,
σ_R rotation noise
N_o resample optimization steps

// sampling importance resampling
$\mathbf{T}_{1..P} \leftarrow \texttt{ImportanceResample}(\mathbf{T}_{1..P}, w_{1..P})$
for $i \leftarrow 1$ **to** P **do**
  $\Delta\mathbf{T}_\sigma \leftarrow \texttt{PerturbTransform}(\sigma_t, \sigma_R)$
  $\mathbf{T}_i \leftarrow \Delta\mathbf{T}_\sigma \cdot \mathbf{T}_i$
for $j \leftarrow 1$ **to** N_o **do**
  differentiate $\hat{C}(\mathcal{X}, \mathbf{T}_{1..P})$ to get $\mathbf{T}_{1..P}$ gradients
  SGD to optimize $\mathbf{T}_{1..P}$

Algorithm 5: `ObjMove`

Given: $\mathcal{X}$ set of previously observed geometric features
$\mathcal{X}'$ set of new observed geometric features
$\mathbf{T}_{1..P}$ pose particles,
$\Delta\mathbf{T}_r$ change in end-effector pose during contact
N_o number of optimization steps

$\Delta\mathbf{T} \leftarrow \Delta\mathbf{T}_r$ // sticking contact prior
$l_{1..P} \leftarrow \hat{C}(\mathcal{X}, \mathbf{T}_{1..P})$
$i \leftarrow \arg\min l_{1..P}$
for $j \leftarrow 1$ **to** N_o **do**
  // hypothesis moved object pose
  differentiate $\hat{C}(\mathcal{X}', \Delta\mathbf{T} \cdot \mathbf{T}_i)$ to get $\Delta\mathbf{T}$ gradients
  SGD to optimize $\Delta\mathbf{T}$
$\Delta\mathbf{T}_w \leftarrow \mathbf{T}_i^{-1} \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$

Algorithm 6: `MergeObs`

Given: $\mathcal{X}$ set of previously observed geometric features,
$\mathcal{X}'$ set of new observed geometric features,
$\mathbf{T}_{1..P}$ pose particles,
$\Delta\mathbf{T}_w$ world frame change in object pose
r_d downsample resolution

$\mathcal{X}_o \leftarrow \{(\Delta\mathbf{T}_w\mathbf{x}, s) \mid (\mathbf{x}, s) \in \mathcal{X}, s \neq \texttt{free}\}$
$\mathcal{X}_f \leftarrow \{(\mathbf{x}, s) \mid (\mathbf{x}, s) \in \mathcal{X}, s = \texttt{free}\}$ // stationary
// remove all that may be occupied now
$\mathcal{X}'_f \leftarrow \{(\mathbf{x}, s) \mid (\mathbf{x}, s) \in \mathcal{X}_f, \texttt{sdf}(\mathbf{T}_i\mathbf{x}) > 0,\ \forall i = 1..P\}$
$\mathcal{X} \leftarrow \mathcal{X}_o \cup \mathcal{X}'_f \cup \mathcal{X}'$
voxel downsample $\mathcal{X}$ with resolution r_d

resample [25] based mostly on removing low weight particles. However, the particle weights only represent their relative probability with respect to other particles, and we have a more direct signal in the discrepancy $l_{1..P} = \hat{C}(\mathcal{X}, \mathbf{T}_{1..P})$ to evaluate when the pose particles have low likelihood. We use this in Algorithm 1 line 16 by comparing the maximum discrepancy of the particles to a threshold $l_r > 0$. For better robustness against outlier pose samples, a percentile of the discrepancy instead of the max can be used. This process is visualized in Fig. 6, where a contact made with the handle at the back of the mug forces a resample due to the previous pose particles' discrepancy with the observed surface geometric features.

Finally, the resampling process is described in Algorithm 4. We first perform the well known sampling importance resampling [25], then like in the prediction step we perturb the pose particles to generate diversity. We then ensure the pose particles have high probability by performing SGD on $\hat{C}(\mathcal{X}, \mathbf{T}_{1..P})$.

E. Planning Problem

We use model predictive path integral (MPPI) control [46] to plan a H horizon length trajectory and execute the first step of it in Algorithm 1 line 6. H may be less than T due to computation limitations. Without loss of generality, consider $t = 1$ at planning time for notation simplification. MPPI samples many Gaussian action perturbations around a nominal action trajectory to produce $\mathbf{u}_{1..H}$, rolls out the robot configuration from $\mathbf{q}_0$ to get $\mathbf{q}_{1..H}$ with a dynamics function, and evaluates each configuration trajectory with a cost function to weigh how the action trajectories should be combined. We initialize the nominal trajectory with noise, warm start it by running MPPI without actually executing the planned trajectory for several iterations. Then when executing $\mathbf{u}_1$, we use $\mathbf{u}_{2..H}, \mathbf{0}$ as the nominal trajectory for the next step. By convention, MPPI minimizes cost, and so we present costs where lower values are better.

F. Information Gain Cost

We assume we have the robot model such that we can map $h(\mathbf{q}) \to \mathcal{R}$ where $\mathcal{R}$ is the set of world coordinate positions inside or on the surface of the robot. Note that when observing $\mathcal{X}'_t$ in Algorithm 1 line 8, $\{(\mathbf{x}, \texttt{free}) \mid \mathbf{x} \in h(\mathbf{q}_{t+1})\}$ should at least be in $\mathcal{X}'_t$ since the object cannot be inside the robot. Additionally, we assume we can identify $h_I(\mathbf{q}) \subset h(\mathbf{q})$ that selects the points of the robot that can observe information through contact. For example, the wrist of the end effector may be much less effective at reliably localizing contact than the tactile sensor. We only consider h_I for gathering information but the full h for the dynamics model.

For a rolled-out configuration trajectory $\mathbf{q}_{1..H}$ we define the

information gain cost

$$C'_I(\mathbf{q}_{1..H}) = \sum_{\mathbf{x} \in D(\bigcup_{i=1}^{H} h_I(\mathbf{q}_i), r_d)} -\tilde{I}(\mathbf{x}|\mathcal{X}) \quad (28)$$

which is the information gain field at every robot interior point in the rolled out trajectory, downsampled to avoid double-counting.

This cost function develops naturally from $\tilde{I}(\mathbf{x}|\mathcal{X})$, however it does not take into account that the object can move, and in doing so, can change $\tilde{I}(\mathbf{x}|\mathcal{X})$. Consider a trajectory where a robot moves into contact with the target object then continues in a straight line with the target object remaining in sticking contact. While it would traverse the workspace and gather high C'_I as a result, relative to the object it has not moved after coming into contact, and so should collect no new information. Indeed, $\tilde{I}(\mathbf{x}|\mathcal{X})$ is better seen as an object frame field, as only motion relative to the object should collect information.

To address this, we introduce predicted object displacement $\mathbf{d} \in \mathbb{R}^3$, and define the adjusted information gain cost

$$C_I(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) = \sum_{\mathbf{x} \in D(\bigcup_{i=1}^{H} [h_I(\mathbf{q}_i) - \mathbf{d}_i], r_d)} -\tilde{I}(\mathbf{x}|\mathcal{X}) \quad (29)$$

where $h(\mathbf{q}_i) - \mathbf{d}_i \ \forall i = 1..H$ transforms the world query positions to be in the displaced object frame.

G. Dynamics Model

We predict the displacement $\mathbf{d}$ in our dynamics model $\hat{f}$ in addition to $\mathbf{q}$. We assume the difference of the true dynamics f from the given free space dynamics f_f is only due to making contact with the target object, and use the precomputed $p(S_\mathbf{x}|\mathcal{X})$ voxel grid to predict when that occurs. One step of $\hat{f}$ is described in Algorithm 7 and below:

First we apply free space dynamics to get candidate configuration $\mathbf{q}'$. We then sample if this configuration leads to contact by considering the least likely to be `free` position $\mathbf{x}_i$ from $h(\mathbf{q}') - \mathbf{d}_t$ in Algorithm 7 line 3. We randomly sample from the categorical distribution $s \sim p(S_{\mathbf{x}_i}|\mathcal{X})$. If we sample $s = \texttt{free}$, then the candidate configuration is used as the next one and the object is not displaced. Otherwise, we need to consider if it is a pushing contact. We compute this action's displacement $\mathbf{d}'$ by considering the change in position from where $\mathbf{x}_i$ was before the action. Then, we estimate the surface normal $\hat{\mathbf{n}}$ at this point in line 11 by taking the weighted sum of the SDF gradient of the contact position transformed by each of the pose particles. If the angle between $\hat{\mathbf{n}}$ and $-\mathbf{d}'$ is less than some threshold θ_p based on an estimation of the friction cone between the robot and the object, then it is considered pushing. If it is a pushing contact, we increase object displacement and move the robot normally. Otherwise, the robot is predicted to remain in its previous configuration to discourage non-pushing contacts, and no further object displacement is produced.

Note that $\mathbf{q}_{t+1}, \mathbf{d}_{t+1} \sim \hat{f}(\mathbf{q}_t, \mathbf{d}_t; ...)$ is stochastic since we sample contacts. To reduce variance, for a single action trajectory $\mathbf{u}_{1..H}$ we roll out multiple configuration trajectories by applying $\hat{f}$ on copies of the starting configuration $\mathbf{q}_1$ and $\mathbf{u}_{1..H}$. The cost of $\mathbf{u}_{1..H}$ is the average cost across the multiple $\mathbf{q}_{1..H+1}$. Practically, if the object has thin walls relative to the distance a single action could move the robot, as in the case of mugs, each action could be divided up and applied sequentially to avoid dynamics predicting the robot penetrating the object walls.

Algorithm 7: $\hat{f}$ — approximate robot and object displacement dynamics

Given: $\mathbf{q}_t$ current robot configuration,
$\mathbf{u}_t$ action,
$\mathbf{d}_t$ current object displacement,
$\mathbf{T}_{1..P}$ pose particles, $w_{1..P}$ particle weights
$p(S_\mathbf{x}|\mathcal{X})$ semantics probability voxel grid,
f_f given free space dynamics,
$\nabla\texttt{sdf}$ object frame SDF gradient,
h robot interior points model,
θ_p pushing angle threshold
Output: $\mathbf{q}_{t+1}$ new robot configuration,
$\mathbf{d}_{t+1}$ new object displacement

1 $\mathbf{q}' \leftarrow f_f(\mathbf{q}_t, \mathbf{u}_t)$ // candidate config
2 $\mathcal{R} \leftarrow h(\mathbf{q}') - \mathbf{d}_t$
// find most likely contact
3 $i \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{R}} p(S_{\mathbf{x}_i} = \texttt{free}|\mathcal{X})$
4 $s \sim p(S_{\mathbf{x}_i}|\mathcal{X})$ // sample semantics
5 **if** $s = \textit{free}$ **then**
6 $\quad \mathbf{q}_{t+1} \leftarrow \mathbf{q}'$
7 $\quad \mathbf{d}_{t+1} \leftarrow \mathbf{d}_t$
8 **else**
$\quad$ // determine displacement
9 $\quad \mathcal{R}_b \leftarrow h(\mathbf{q}_t)$
10 $\quad \mathbf{d}' \leftarrow \mathbf{x}_i - \mathcal{R}_b[i]$// corresponding ith position
$\quad$ // estimate object surface normal
11 $\quad \hat{\mathbf{n}} \leftarrow \sum_{j=1}^{P} w_j \nabla\texttt{sdf}(\mathbf{T}_j \mathbf{x}_i)$
12 $\quad$ **if** *angle between* $\hat{\boldsymbol{n}}$ *and* $-\boldsymbol{d}' < \theta_p$ **then**
$\quad\quad$ // pushing or not
13 $\quad\quad \mathbf{q}_{t+1} \leftarrow \mathbf{q}'$
14 $\quad\quad \mathbf{d}_{t+1} \leftarrow \mathbf{d}_t + \mathbf{d}'$
15 $\quad$ **else**
$\quad\quad$ // discourage non-pushing contact
16 $\quad\quad \mathbf{q}_{t+1} \leftarrow \mathbf{q}_t$
17 $\quad\quad \mathbf{d}_{t+1} \leftarrow \mathbf{d}_t$

H. Reachability Cost

For manipulator arms with immobile bases, it is important to explicitly penalize when actions could move the object outside of its reachable region. Under just the information gain cost from Eq. 29, an action trajectory pushing the object out of reach will evaluate to have equal or better cost than a trajectory doing nothing. If the object is at the edge of the robot's reachability, such as a mug with sides that are within reach but the occluded handle at the back being out of reach, sampling a H step trajectory that first displaces the mug then collects the high information gain at the back of the mug is very unlikely. H may also be too short to allow such a trajectory to exist.

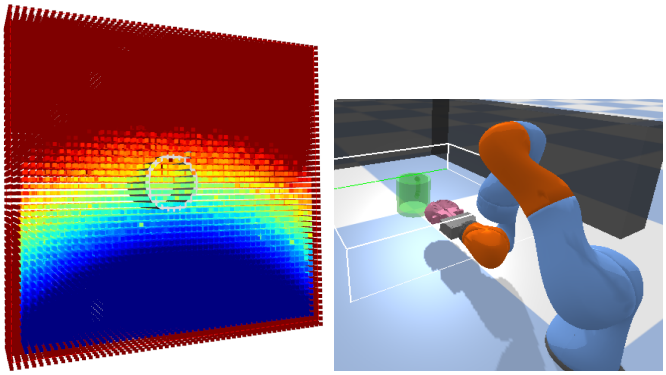


Fig. 7: (left) Reachability of workspace positions for the (right) simulated KUKA arm and workspace. Red corresponds to $r(\mathbf{x}) = 0$ and dark blue corresponds to $r(\mathbf{x}) = 1$. The workspace in sim is drawn as a box around the object.

To address this, we introduce *reachability* $r(\mathbf{x}) \in [0, 1]$ and the reachability cost $C_R(\mathbf{d}_{1..H})$ which encodes the desired behaviour of pushing object frame points $\mathbf{x}$ with high $\tilde{I}(\mathbf{x}|\mathcal{X})$ to where they are reachable.

Reachability $r(\mathbf{x})$ represents the capability of the robot to gather information at $\mathbf{x}$, similar to checking $\exists \mathbf{q}$ s.t. $\mathbf{x} \in h_I(\mathbf{q})$. This can be approximated by performing inverse kinematics (IK) with $\mathbf{x}$ set as the goal position relative to the robot end effector frame. We also consider how robust $\mathbf{x}$ is to reach with different configurations, and evaluate the average IK performance with a fixed set of goal orientations $R_i \in R \subset \mathrm{SO}(3)$. Let $e_{\mathbf{x}}(\mathbf{x}, R_i)$ and $e_R(\mathbf{x}, R_i)$ be the position and rotation errors from running IK with the goal set to $(\mathbf{x}, R_i)$. We weigh e_R against $e_{\mathbf{x}}$ with $\alpha_R \geq 0$ and define an error tolerance threshold e_m such that any error at or above this value receives $r(\mathbf{x}) = 0$. Thus we define

$$r(\mathbf{x}) = \frac{1}{e_m} \max(0, e_m - \frac{1}{|R|} \sum_{R_i \in R} [e_{\mathbf{x}}(\mathbf{x}, R_i) + \alpha_R e_R(\mathbf{x}, R_i)]) \tag{30}$$

We precompute this for $\mathbf{x} \in \mathcal{W}$ and store the results in a voxel grid that allows linear interpolation. This only has to be done once per robot and workspace combination. See Fig. 7 for an example visualization of $r(\mathcal{W})$.

The reachability cost $C_R(\mathbf{d}_{1..H})$ is then the total reachable information within the workspace after displacement. We compute it according to Algorithm 8. First we compute $\bar{\tilde{I}}(\mathbf{x})$, the average information gain at every displaced workspace position over the planning horizon. Note that $\tilde{I}(\mathbf{x}|\mathcal{X})$ can be interpreted as the object frame information gain field at the time of planning, and so stationary workspace positions are effectively displaced by $-\mathbf{d}_{1..H}$ during planning. The reachable information is just the product $\bar{\tilde{I}}(\mathbf{x})r(\mathbf{x})$ which we sum across all the workspace points. This is then compared against the total information in the workspace to produce a negative ratio $C_R \in [-1, 0]$. Because C_I and C_R are in different units, having C_R be a ratio allows easier tuning of the total trajectory cost:

$$C(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) = \beta_I C_I(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) + \beta_R C_R(\mathbf{d}_{1..H}) \tag{31}$$

Algorithm 8: C_R reachability cost

Given: $\mathbf{d}_{1..H}$ object displacement trajectory,
$\mathcal{W}$ workspace,
$\tilde{I}(\mathbf{x}|\mathcal{X})$ information gain voxel grid,

$\bar{\tilde{I}}(\mathbf{x}) \leftarrow \frac{1}{H} \sum_t^H \tilde{I}(\mathcal{W} - \mathbf{d}_t|\mathcal{X})$ `// average info`
$RI \leftarrow \sum_{\mathbf{x} \in \mathcal{W}} \bar{\tilde{I}}(\mathbf{x}) r(\mathbf{x})$ `// reachable info`
$RI_m \leftarrow \sum_{\mathbf{x} \in \mathcal{W}} \tilde{I}(\mathbf{x}|\mathcal{X})$ `// max info possible`
$C_R(\mathbf{d}_{1..H}) \leftarrow -RI/RI_m$

I. Kernel Interpolated MPPI

The total cost from Eq. 31 does not include any explicit smoothing terms. To improve the smoothness of produced trajectories, we perform interpolation similar to [32]. The idea is to sample $H_v < H$ control points $\mathbf{v}_i \in \mathbb{R}^{N_u}$, then use a kernel $K : \mathbb{R}^{N_u} \times \mathbb{R}^{N_u} \to \mathbb{R}$ to interpolate the $\mathbf{u}_{1..H}$ in between $\mathbf{v}_{1..H_v}$. We call this method Kernel Interpolated MPPI (KMPPI). This is more general than the B-spline interpolation of [32] since it can be accomplished by using a B-spline kernel.

Let $\mathbf{H_u} = [0, 1, ..., H-1]$ denote the time coordinate of each $\mathbf{u}$ along the trajectory. We assume $\mathbf{v}_{1..H_v}$ are evenly spread out along the trajectory, and since there are H_v of them, subsequent ones increase their time coordinate by $\frac{(H-1)}{(H_v-1)}$ to give $\mathbf{H_v} = [0, \frac{(H-1)}{(H_v-1)}, \frac{2(H-1)}{(H_v-1)}, ..., H-1]$. The even assignment of $\mathbf{H_v}$ is not necessary; any can be given as long as the first term is 0 and the last term is $H-1$. Given a control sequence $\mathbf{v}_{1..H_v}$ we then convert it to $\mathbf{u}_{1..H}$

$$\mathbf{u}_{1..H} = K(\mathbf{H_u}, \mathbf{H_v}) K(\mathbf{H_v}, \mathbf{H_v})^{-1} \mathbf{v}_{1..H_v} \tag{32}$$

This allows smoothing in the action space, rather than in the robot configuration space, and we observe that it works well on our tasks. See Fig. 8 for a qualitative evaluation of the smoothing property on a toy 2D problem.

With the interpolated $\mathbf{u}_{1..H}$, KMPPI's subsequent steps are the same as MPPI's in that it generates configuration rollouts by applying the dynamics function $\mathbf{q}_{t+1}, \mathbf{d}_{t+1} \sim \hat{f}(\mathbf{q}_t, \mathbf{d}_t; ...), t = 1..H$, evaluates the cost of each $\mathbf{u}_{1..H}$ with Eq. 31, then combines the trajectory samples with a softmax based on the cost.

J. Termination Condition

In actual execution, we do not have access to $\mathbf{T}^*$ to evaluate $nll(\mathcal{X})$ and need another signal to terminate execution. We use the convergence of the pose particles, with the hypothesis that pose particles likely only converge when $p(\mathbf{T}^*|\mathcal{X})$ is high, i.e. the pose particles do not randomly converge to an incorrect estimate. We evaluate convergence using the average square root pairwise Chamfer distance between the pose particles (APC). Similar to the $nll(\mathcal{X})$ evaluation, we evaluate this on a sampled set of object frame surface positions $\tilde{x} \in \tilde{\mathbf{X}}$.

$$APC(\mathbf{T}_{1..P}) = \frac{1}{P^2 |\tilde{\mathbf{X}}|} \sum_{i=1}^{P} \sum_{j=1}^{P} \sum_{\tilde{x} \in \tilde{\mathbf{X}}} |\mathtt{sdf}(\mathbf{T}_i^{-1} \mathbf{T}_j \tilde{x})| \tag{33}$$

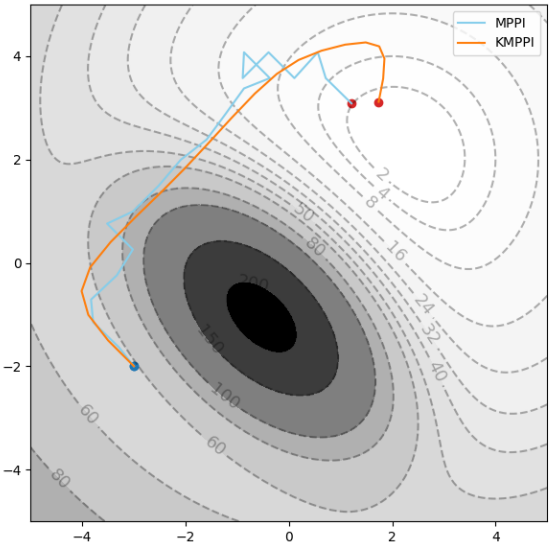


Fig. 8: Planned trajectories on a toy 2D linear integrator environment. The cost contour map is represented, with the cost of the trajectory being the accumulated cost experienced at each state. There is additionally a quadratic action penalty at each step $\mathbf{u}^T 0.1\mathbf{I}\mathbf{u}$. Dark blue is the starting state. Each trajectory has $H = 20$, with our KMPPI having $H_v = 5$ and using the radial basis function kernel with scale 2.

We terminate execution when $APC(\mathbf{T}_{1..P}) < \beta_t l_c$, where l_c is the diagonal length of the object's bounding box, and β_t is a ratio that selects for a desired level of pose particle convergence. A lower value means rummaging will continue for longer, but may produce a more accurate pose estimate.

V. Experiments

In this section, we first describe our simulated and real robot environments. We then detail the experiments to estimate the pose of a movable target object. We introduce our baselines and ablations and how we quantitatively evaluate the methods on the experiment. We then show how RUMI could be applied when there are an unknown number of unknown movable objects cluttered around our target object. Lastly, we present results that show RUMI is the only method to perform consistently well across all the experiments.

A. Sim Environment

Common to all the experiments, we have a single movable object on a flat surface starting within reach of a single 7DoF KUKA LBR iiwa arm with two soft-bubble tactile sensors [21]. This is modelled in sim in Fig. 7. Due to the complexity of modelling deformable objects, we model the soft-bubble tactile sensors as rigid bodies and observe the surface points of any object penetrating them after each simulation step. We also include a fixed external depth camera to reduce the initial exploration required, but we also show that our method works without a good initial view of the object in some experiments.

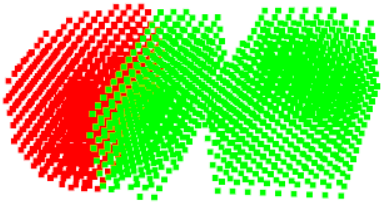


Fig. 9: Visualization of interior robot points $h(\mathbf{q})$ as green and red points, and the information gathering subset $h_I(\mathbf{q})$ as just the red points for the robot gripper mounted with two soft-bubble tactile sensors seen in Fig. 1.

For observing $s = \texttt{surface}$ points at $\mathbf{q}_t$, we select $\{\mathbf{x} | \ \mathbf{x} \in h_I(\mathbf{q}_t), |\texttt{sdf}(\mathbf{T}^*\mathbf{x})| < 3mm\}$. This simulates some observation noise which we show that RUMI is robust to, despite assuming no noise in the observed positions. We assume we can only gather contact information from the front of the gripper, where the two soft-bubble tactile sensors are mounted. In planning, this is the difference between $h(\mathbf{q})$ and $h_I(\mathbf{q})$ for our end effector shown in Fig. 9.

The $\mathcal{X}'$ provided by the depth camera includes $s = \texttt{free}$ points generated by tracing rays from the camera to 95% of each pixel's detected depth, and $s = \texttt{surface}$ from segmented object surfaces. See Fig. 4 (middle), and Fig. 11 for example observation point clouds. We only use vision to provide the initial $\mathcal{X}_0$ to demonstrate the viability of tactile based rummaging.

To highlight the difference between other components of all methods, we directly observe $\Delta\mathbf{T}$ in Algorithm 1 line 9. Note that this also applies to all baselines and ablations, and so does not provide an unfair advantage to RUMI. This is equivalent to assuming we can accurately measure slip between the end effector and object.

B. Sim Tasks

In simulation, we experiment on 3 different objects: a mug, a YCB [5] power drill, and a YCB cracker box, each with 3 different initial poses depicted in Fig. 10. For each, we perform 10 runs of $T = 40$ steps, using a different fixed random seed for each run that is shared across baselines and ablations. We terminated tasks early if the pose particles converged as measured by $APC(\mathbf{T}_{1..P}) < 0.03 l_c$, where l_c is the diagonal distance of each object's bounding box.

In each experiment, the robot's end effector is position and yaw controlled, with the action space either being $\mathbf{u} = [dx, dy, d\theta]$ (planar) or $\mathbf{u} = [dx, dy, dz, d\theta]$ (3D) with ranges from [-1,1] for each dimension. The action spaces are scaled to allow the use of consistent KMPPI parameters across experiments. We scale these to physical units by translating a control value of 1 to $80mm$ or 0.5 radians, carried out in many mini steps. We perform inverse kinematics to convert these to joint position commands. We used regular grids with resolutions (grid square side length) r_w as the workspaces. Note that other

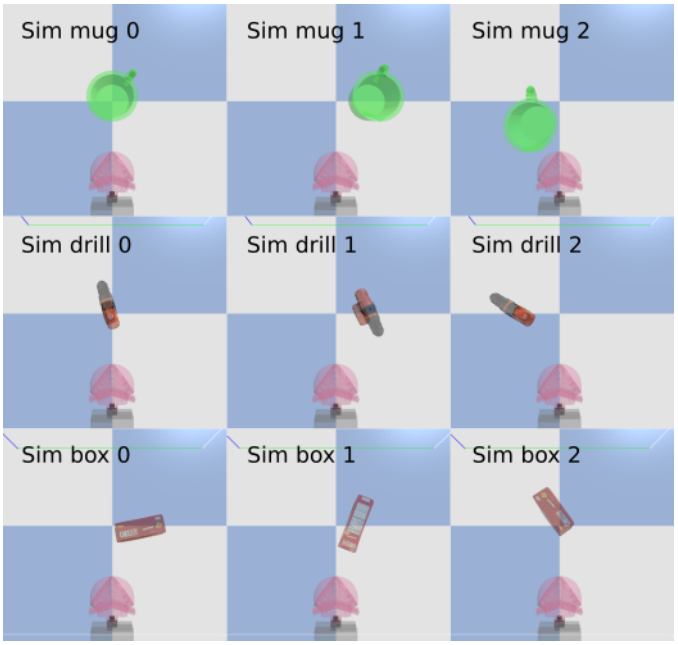


Fig. 10: Starting poses for labelled sim tasks.

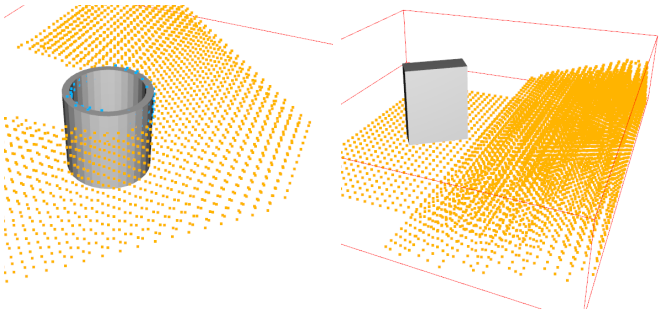


Fig. 11: Comparison of the rendered $\mathcal{X}_0$ given to (left) the sim mug 0 task and (right) sim box 2 task. $s = \texttt{free}$ points are in orange, and $s = \texttt{surface}$ points are in blue. There are no initially observed surface points on the box.

sets of worldspace points that are not necessarily regular grids could be used. We used $\mathcal{W} = [0, 0.8] \times [-0.4, 0.4]$ in meters for planar action spaces, and $\mathcal{W} = [0, 0.8] \times [-0.4, 0.4] \times [0, 0.2]$ for 3D action spaces. r_w for each task can be found in Tab. I.

C. Pose Prior

We used different $p(\mathbf{T})$ for each task as each had different assumptions to illustrate separate aspects of exploration. For the mug and power drill, we assume the object stays upright and search for their pose in SE(2) instead of SE(3). The mug tasks evaluates how well $\tilde{I}(\mathbf{x}|\mathcal{X})$ conforms to our intuition, since we expect the most information to be where the handle could be. The sim drill task evaluates how well our planner extends to objects with complex geometry. The sim box task tests how well the pose particles can represent full SE(3) and the necessary 3D exploration to identify which side of the box is lying against the floor. Additionally, for the drill and box tasks, we increase the difficulty in terms of environmental occlusions by placing the camera at an angle such that it cannot directly observe the object. The camera configurations and the initial object pose are depicted in Fig. 11. For SE(3) pose search in the sim box experiments, we add free points where the floor is to avoid pose estimates that penetrate the floor. The different task setups are summarized in Tab. I.

To sample $\mathbf{T}_{0,1..P} \sim p(\mathbf{T})$ on mug tasks since we can initially observe the side of the object unlike the drill and box tasks, we first estimate the position of the center of the mug, then $\mathbf{T}_{0,1..P}$ is sampled with uniformly random yaw at the same center. For the other tasks, we sample $\mathbf{T}_{0,1..P}$ with random positions sampled from $\mathcal{N}(0, 0.05) \times \mathcal{N}(0, 0.05) \times 0$, and uniformly random yaw for the drill and uniformly over SO(3) for the box.

D. Real Environment

The real robot setup is seen in Fig. 1 and Fig. 10. It uses the same robot as in simulation. The soft-bubble sensors are compliant to contact and have a depth camera inside to estimate dense contact patches. Similar to prior work [50], we consider points on the soft bubble surface with deformation beyond $4mm$ and being in the top 10^{th} percentile of all deformations to be in contact. We apply a mean filter to remove noise. We use a RealSense L515 lidar camera as the fixed external camera. For evaluating ground truth object pose, we have a RealSense D435 camera mounted looking top-down on the workspace.

The mug had distinct colors from the shelf and so we segmented it with a color filter. To improve segmentation, we used a robot self-filter and an edge filter to remove unreliable points, and used a temporal filter to only accept surface points that persists over a 0.4s window. See Fig. 1 for example observation point clouds. We re-observe the scene after each action. Due to self-occlusion and object symmetry, visual observations do not uniquely identify object pose. Same as for the simulated box, the real box task has occluded vision that prevented direct observation of it, seen in the top of Fig. 12.

For the real mug task, we do not assume we can accurately measure slip between the end effector and object. Instead, we estimate $\Delta\mathbf{T}$ with Algorithm 5 for all methods. For the real box task, we observe the change in object pose from the ground truth since we cannot directly observe the object to estimate $\Delta\mathbf{T}$ with Algorithm 5.

E. Real Task

We estimate the pose of a real mug and box starting in a single configuration depicted in Fig. 12 and execute $T = 15$ steps of each method. The robot's action space is seen in Tab. I, and a control value of 1 corresponds to $50mm$ or 0.4 radians. The workspace was $\mathcal{W} = [0.55, 1.1] \times [-0.33, 0.33] \times [0.23, 0.47]$ in meters (for planar action space, a fixed height of 0.305m was used). The $\mathbf{T}_{0,1..P}$ initialization process is similar to sim for each corresponding task, with $\mathbf{T}_{1..P}$ after sampling from CHSEL in Algorithm 1 line 2 shown at the bottom of Fig. 12. Note that the box's initial $\mathbf{T}_{1..P}$ covers the workspace since vision was occluded. We terminated tasks when the pose particles converged as measured by $APC(\mathbf{T}_{1..P}) < 0.05 l_c$, where l_c is the diagonal distance of each object's bounding box.

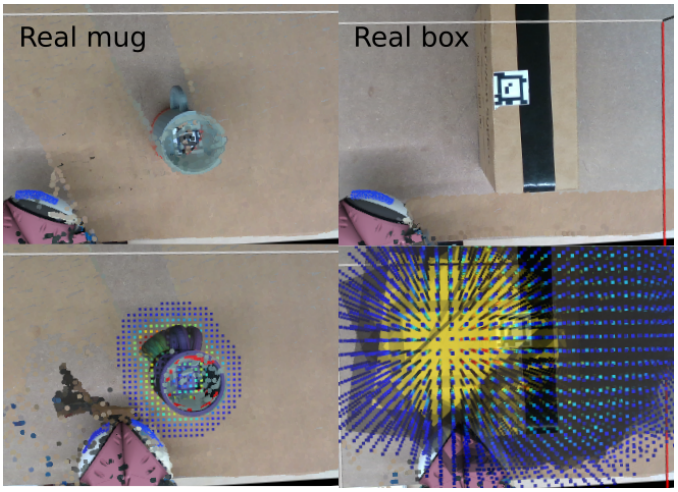


Fig. 12: Initial configuration of the real mug and box tasks (top) and example initialized pose particles (bottom). In the box task, the workspace is occluded and the box cannot be directly observed. Point cloud observations from an external side view (accessible to the robot) are overlaid on a top-down raw camera view (inaccessible to the robot).

Object	action space	pose search space	resolution r_w (m)
Sim mug	planar	SE(2)	0.01
Sim drill	3D	SE(2)	0.02
Sim box	3D	SE(3)	0.02
Real mug	planar	SE(2)	0.01
Real box	3D	SE(2)	0.02

TABLE I: Task setup for different objects.

F. Sensor Model

We use the sensor model depicted in Fig. 2. Let $v = \texttt{sdf}(\mathbf{Tx})$ be the SDF value of a given query position $\mathbf{x}$. To represent bias towards over-reporting contact in our sensors, we use a tolerance of $\zeta = 0.003m$ and let $\tilde{v} = \textbf{sign}(v) \cdot \max(0, |v| - \zeta)$, where $\textbf{sign}(v) = 1$ if $v > 0$ else -1. Then $p(S_\mathbf{x}|\mathbf{T}) = p(S_\mathbf{x}|\texttt{sdf}(\mathbf{Tx})) = p(S_\mathbf{x}|v)$ is defined by

$$p(S_\mathbf{x}|v) = \begin{cases} \max(0, 1 - e^{-\alpha\tilde{v}}) & \texttt{free} \\ \max(0, 1 - e^{\alpha\tilde{v}}) & \texttt{occupied} \\ e^{-\alpha|\tilde{v}|} & \texttt{surface} \end{cases}$$

with $\alpha = 100$ where v is in meters. This model represents some of the ambiguities of detecting contact with the soft-bubble and similar tactile sensors. Due to the compliance of the membrane, even when a point is in free space, contact elsewhere could make it appear like this point is also in contact. Similarly, contact could also be missed, particularly around the edges of the soft-bubble. This sensor model performed well enough both in sim and on the real task that no calibration to the real soft-bubbles was needed.

G. KMPPI Parameters

We used a planning horizon of $H = 15$ and $H_v = 8$ number of control points. This is lower than the number of sim steps $T = 40$ because increasing horizon resulted in poorer-quality trajectories. This is due to the cost from Eq. 31 being a terminal cost for the whole trajectory, without distinguishing between steps inside the trajectory. We used the radial basis function (RBF) kernel with a scale of 2.

We planned using 500 action trajectory samples, each rolled out 5 times with $\hat{f}$ due to its stochastic nature. Additionally, to avoid contacts that penetrate the object, we split each action up into 4 sequentially applied actions that are 4 times lower in magnitude. We then use the average trajectory cost across the 5 rollouts. We replanned after executing 3 actions, or when the robot detects it is in contact.

For the inner MPPI parameters, we used $\lambda = 0.01$ for the temperature parameter from [46], with $\mathbf{0}$ noise mean and $1.5\mathbf{I}$ as the noise covariance.

H. Evaluation

We evaluate using the key metric of NLL defined in Eq. 2. Note that low NLL indicates both certainty and correctness of the pose distribution. We sample 500 positions $\tilde{x} \in \tilde{\mathbf{X}}$ uniformly on the surface of the object, and transform them to world positions with $\mathbf{T}^*$, the ground truth pose, to produce $\mathbf{X}$. We then evaluate the negative log likelihood of $\mathbf{x} \in \mathbf{X}$ being $\texttt{surface}$ points from Eq. 2. Because we assume $S_\mathbf{x}$ is conditionally mutually independent to every other $S_\mathbf{x}$ given $\mathcal{X}$, we can simplify Eq. 2

$$nll(\mathcal{X}) = -\log p(\bigcap_{\mathbf{x}\in\mathbf{X}} S_\mathbf{x} = \texttt{surface}|\mathcal{X}) \tag{34}$$

$$= -\sum_{\mathbf{x}\in\mathbf{X}} \log p(S_\mathbf{x} = \texttt{surface}|\mathcal{X}) \tag{35}$$

We substitute Eq. 27 in for $p(S_\mathbf{x}|\mathcal{X})$ to approximate nll with our pose particles

$$nll(\mathcal{X}) \approx -\sum_{\mathbf{x}\in\mathbf{X}} \log \sum_{i=1}^{P} w_i p(S_\mathbf{x} = \texttt{surface}|\texttt{sdf}(\mathbf{T}_i\mathbf{x})) \tag{36}$$

For the sim tasks, we have the ground truth object pose $\mathbf{T}^*$, while for the real tasks, we observe $\mathbf{T}^*$ from a camera mounted above the workspace. We evaluated nll after each step as an effective exploration rate. Additionally, we specify a nll threshold below which we qualitatively observe to be a good enough quality to be considered a success, seen in Tab. III. A run is counted a success if it achieves a minimum nll below the threshold at any step. This is typically, but not always, the last step. This is because, due to observation noise and moving the object outside of the observed region, the pose estimates could become less certain.

We also use the same $\tilde{x} \in \tilde{\mathbf{X}}$ to evaluate APC from Eq. 33. We used $\beta_t = 0.03$ for the sim tasks, meaning we terminated exploration when the average square root chamfer distance between all pairs of $\mathbf{T}_{1..P}$ is less than 3% of the object's bounding box diagonal length. For the real experiment we used $\beta_t = 0.05$.

We also investigated our hypothesis of $APC(\mathbf{T}_{1..P})$ as a good proxy for $nll(\mathcal{X})$ since it can be computed without privileged information. We did so by computing the linear correlation between the two across all the tasks. The runs from all methods were used. This is shown in Tab. II and Fig. 13 for

Task	sim mug			sim drill			sim box		
	0	1	2	0	1	2	0	1	2
cor(nll, APC)	0.93	0.82	0.84	0.74	0.94	0.94	0.58	0.80	0.49

TABLE II: Linear correlation between $nll(\mathcal{X})$ and $APC(\mathbf{T}_{1..P})$ across all sim tasks. Runs from all methods were considered in its calculation.

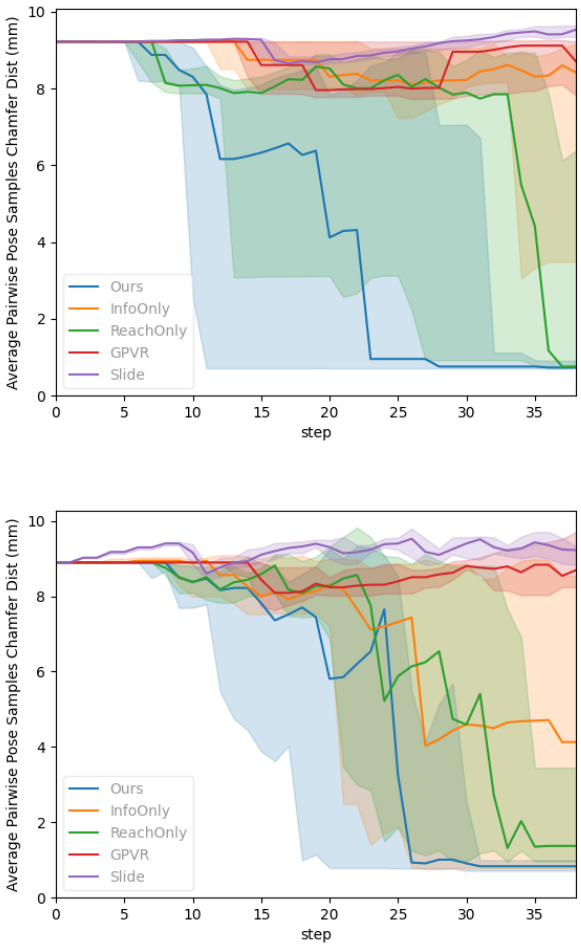


Fig. 13: Convergence of pose particles measured by $APC(\mathbf{T}_{1..P})$ for the sim mug 0 and sim mug 1 tasks. The median over 10 runs is plotted, with the 25^{th} to 75^{th} percentile shaded. There is strong correlation with the $nll(\mathcal{X})$ in the top left and top middle of Fig. 18.

the sim mug 0 and sim mug 1 tasks, which can be compared to the $nll(\mathcal{X})$ shown in the top left and top middle of Fig. 18. We see that there is an especially strong positive correlation for SE(2) particles of the sim mug and sim drill tasks, averaging to a correlation of **0.87**. The correlation for the SE(3) sim box tasks is not as strong.

I. Baselines and Ablations

Our full method parameters are summarized in Tab. IV. These parameters were used for all simulated and real tasks (except for β_t in deciding when to terminate), demonstrating the robustness of RUMI. For downsampling the observations

Object	success nll threshold
Sim mug	20
Sim drill	100
Sim box	150
Real mug	35
Real box	250

TABLE III: Maximum nll threshold for success for each task.

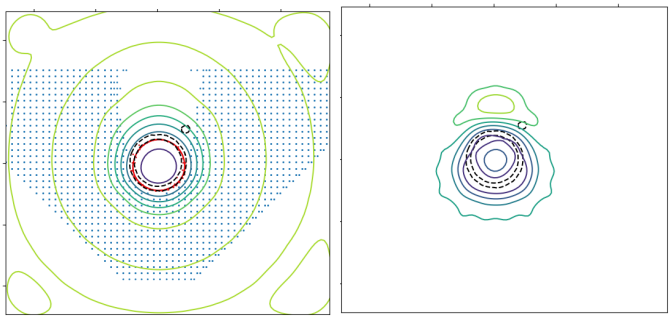


Fig. 14: (left) Gaussian Process fit to initial observations $\mathcal{X}_0$ of the sim mug 0 task. Free space observed points are shown as blue dots, surface observed points as red dots, and the ground truth object surface as black dotted lines. The GP output is overlayed as a contour map, with the red line indicating the 0-level set, corresponding to where the GPIS surface is. (right) $\mathbf{var}(\mathbf{x}|\mathbf{X})$ contour map for the same GP, with green indicating higher variance.

in Algorithm. 6, we used different resolutions for the free space ($r_{d,f}$) and surface ($r_{d,s}$) points; we did not observe any occupied points. The baselines also required the creation and update of the $p(\mathbf{T}|\mathbf{X})$ pose particles, and we use the same parameters to do so.

We present two ablations to our full method, **InfoOnly** which sets C_R to 0 and **ReachOnly** which sets C_I to 0. They share all other parameters with the full method and evaluate the usefulness of each individual cost.

For baselines, we first present the **Slide** heuristic inspired by [11]. This method has two modes of operation - if it is currently in contact, then it moves tangentially to the estimated surface normal to slide along it. It moves parallel to the shelf, and for each run randomly decides at the start of the run whether to slide clockwise or counterclockwise around contact. If it is not in contact, then it moves towards the estimated center of the object. Estimating the object center requires our pose particles, so we still update $p(\mathbf{T}|\mathcal{X})$ using Algorithm 1.

We also consider a Gaussian Process Implicit Surface baseline (GPIS) [3], [11], [22] that uses the variance of the GP as the exploration signal that we call GP Variance Reduction (**GPVR**). The GP is fit on $\{(\mathbf{x}, 0)|\ (\mathbf{x}, s) \in \mathcal{X}_t, s = \texttt{surface}\} \cup \{(\mathbf{x}, 1)|\ (\mathbf{x}, s) \in \mathcal{X}_t,\ s = \texttt{free}\}$. As typical for GPIS, surface points are labelled 0 and free points are labelled 1. It is refit on $\mathcal{X}_t$ for 50 optimization steps after every robot execution step. We use the $\nu = 1.5$ Matern kernel as recommended by [22]. See Fig. 14 for a visualization of the fitted GP as well as its variance $\mathbf{var}(\mathbf{x}|\mathcal{X})$ on the sim mug 0 task given $\mathcal{X}_0$.

For GPVR to be competitive, we had to make several

Parameter	value
P number of pose particles	100
λ peakiness	2
l_r discrepancy resample threshold	5
C CHSEL freespace discrepancy scale	10
H planning horizon	15
θ_p pushing angle threshold	45 degrees
C_I information gain cost scale	1
C_R reachability cost scale	200
e_m reachability IK error threshold	0.4
α_R reachability IK rotation error scale	0.1
$r_{d,f}$ downsample resolution free space	10mm
$r_{d,s}$ downsample resolution surface	2mm
σ_t pose translation noise	10mm
σ_R pose rotation noise	0
β_t chamfer distance convergence ratio	0.03 (0.05 for real)
N_o number of optimization steps	10

TABLE IV: Our method parameters across the different tasks.

modifications. Firstly, we needed to encode object shape as that is given information to RUMI. This is non-trivial to do by modifying the kernel, so we instead augmented the input data with $\{(\mathbf{x}, 1) |\ \mathbf{x} \in \mathcal{W},\ p(S_{\mathbf{x}} = \texttt{free}|\mathcal{X}_t) > 0.99\}$. We voxel downsampled all free points with a resolution 7 times r_w from Tab. I to avoid extremely slow inference and enforce consistent data density. Again, this baseline requires the computation and maintenance of $p(\mathbf{T}|\mathcal{X})$ with the pose particles to enable the estimation of $p(S_{\mathbf{x}} = \texttt{free}|\mathcal{X}_t)$. Without the above data augmentation, GPVR explores the unobserved corners of the workspace, despite seeing parts of the object elsewhere. Secondly, we needed to plan further than just the next step. Otherwise, because we start in and are surrounded by free space, the method goes in initially random directions. Instead of the greedy policy of maximizing the GP variance $\mathbf{var}(\mathbf{x})$ at the next position from [11], we formulated a cost function based on variance reduction for use as a running cost inside KMPPI.

$$C_{GP}(\mathbf{q}_t, \mathbf{d}_t) = \sum_{\mathbf{x} \in D(h_I(\mathbf{q}_t) - \mathbf{d}_t, r_d)} -\beta^{t-1}\, \mathbf{var}(\mathbf{x}|\mathcal{X}) \tag{37}$$

with a discount factor $\beta = 0.99$ to prioritize early rewards. Empirically, this worked better with $\mathbf{var}(\mathbf{x}|\mathcal{X})$ than voxelizing the entire trajectory as in Eq. 29. Before each planning step we precomputed $\mathbf{var}(\mathbf{x}|\mathcal{X})\ \forall \mathbf{x} \in \mathcal{W}$ to store in a voxel grid for faster repeated lookup. We normalized $\mathbf{var}(\mathbf{x}|\mathcal{X})$ such that $\max_{\mathbf{x} \in \mathcal{W}} \mathbf{var}(\mathbf{x}|\mathcal{X}) = 1$.

See Fig. 17 for a comparison of $\tilde{I}(\mathbf{x}|\mathcal{X})$ against $\mathbf{var}(\mathbf{x}|\mathcal{X})$ to be planned over in a similar manner. From the figure, we see that $\mathbf{var}(\mathbf{x}|\mathcal{X})$ is low at where the handle could be. This is because $\mathcal{X}$ includes the inside back of the mug, and the Matern kernel does not directly encode object shape but is just based on the Euclidean distance between points. It cannot separate the certainty of the back surface of the mug from the uncertainty of where the handle is, because it does not know that a handle exists. Instead, $\mathbf{var}(\mathbf{x}|\mathcal{X})$ is highest farther behind the mug, where we have observed no data due to occlusion. This is contrasted with $\tilde{I}(\mathbf{x}|\mathcal{X})$, which is highest where the handle could be because those regions are where the pose particles disagree the most.

J. Rummaging in Clutter Experiment

To evaluate the applicability of RUMI in more practical settings, we consider sim tasks that have an unknown number of unknown movable objects cluttered around the target object depicted in Fig. 15.

To handle the presence of other objects, we first cluster the initial camera observation's surface points using HDBSCAN [29] into K *object clusters*. K is not specified but instead determined by HDBSCAN. Let $\mathcal{X}_k, k = 1..K$ denote each object cluster, with every point having `surface` semantics. As short hand, $\mathcal{X}_k$ will be treated as a set of points or as a set of $(\mathbf{x}, \texttt{surface})$ pairs depending on context. Note the `free` points $\mathcal{X}_f$ are shared by every object. We then apply CHSEL to each $\mathcal{X}_k \cup \mathcal{X}_f$ with a batch of 100 initial transforms centered on the centroid of each $\mathcal{X}_k$ with uniform orientation. We evaluate the discrepancy (Eq. 3) on all registered transforms, assigning the $\mathcal{X}_k$ with the lowest median discrepancy as the target $k = \bar{k}$ to perform RUMI on, meaning $\mathcal{X} = \mathcal{X}_{\bar{k}} \cup \mathcal{X}_f$. This process is depicted in Fig. 16.

When making contact during execution, we assign each contact point the object cluster with the closest centroid. For contact with the target, Algorithm 1 is applied normally. For contact with a non-target ($k \neq \bar{k}$), the contact points are added to $\mathcal{X}_k$ and $\mathcal{X}_k$ is transformed by the measured change in end effector pose while in contact, essentially assuming sticking contact.

For planning, we found it advantageous to consider object cluster dynamics and introduce an *obstruction* cost function to penalize moving non-target $\mathcal{X}_k$ to high information gain regions. The non-target object would be obstructing the robot from observing those points and gaining information. Dynamics is similar to Algorithm 7, but we replace lines 3-5 with checking for $\mathbf{x} \in \mathcal{X}_k$ that has the lowest robot `sdf` value (most inside the robot). Thus, we estimate the displacement of all objects $\mathbf{d}_{1..K}$. The obstruction cost is similar to the information gain cost of Eq. 29:

$$C_O(\mathbf{d}_{1..K,1..H}) = \sum_{\mathbf{x} \in D(\bigcup_{i=1}^{H} \bigcup_{k=1,k\neq\bar{k}}^{K} [\mathcal{X}_k + \mathbf{d}_{k,i} - \mathbf{d}_{\bar{k},i}], r_d)} \tilde{I}(\mathbf{x}|\mathcal{X}) \tag{38}$$

where $\mathcal{X}_k$ is transformed to be the displaced object cluster points in the target object frame. Conveniently, C_O and C_I both evaluate information gain in the workspace, with the voxel downsampling D ensuring comparable density of query points, meaning the cost scales can be more easily tuned with respect to each other.

K. Results

The simulated task results are in Fig. 18 and the real task results are in Fig. 20. The number of successful trials out of 10 for each task is compared in Tab. V. Additionally, the median over the cumulative $nll(\mathcal{X})$ of each run are in Tab. VI. For the sim and real tasks, cumulative $nll(\mathcal{X})$ over time is a good indicator of exploration speed; however, for the box and sim drill tasks, cumulative $nll(\mathcal{X})$ is dominated by the initial search for the first surface points of the object since they do not start with the object in view. Thus, for those tasks it is

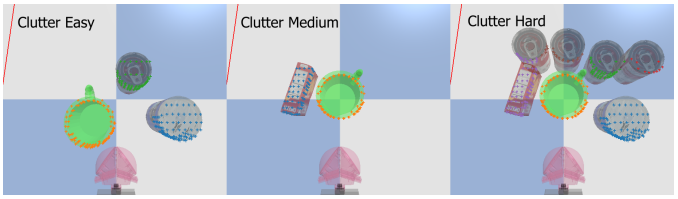


Fig. 15: Starting configuration of clutter experiments with clustered surface points rendered in different colors. All objects are movable and the mug is the target object. Clutter Medium may appear simpler than Clutter Easy, but due to the closer proximity of the objects, it becomes a more difficult task.

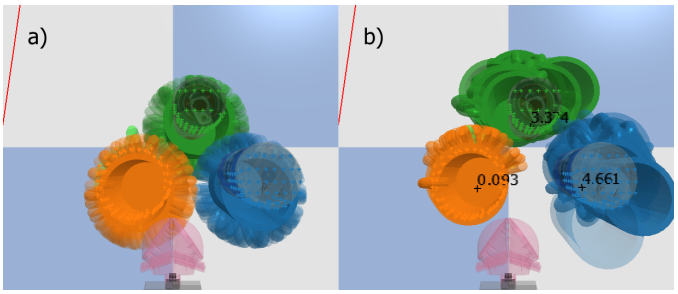


Fig. 16: Target object cluster selection process on Clutter Easy with a) 100 initial transforms centered on the centroid of each $\mathcal{X}_k$ and b) those transforms after applying CHSEL and the resulting median discrepancy of each cluster.

more a measure of how quickly the different methods make first contact with the object.

We observe that RUMI is the only method to achieve consistently good performance, if not the most number of successes, across all the sim and real tasks. On the sim mug tasks, it also had the lowest cumulative $nll(\mathcal{X})$, meaning it was the most efficient. The ablations show that both C_I and C_R are important for this task, although individually they can also perform well on certain tasks. For example on the sim mug task, ReachOnly achieved a high number of successes by itself. This was likely due to the handle being close to where the robot needed to push from to increase reachability. However even in this case, adding C_I improves efficiency because the mug could be pushed into more reachable regions without contacting the handle. This explains the occasional failures of the ReachOnly method on the mug tasks. On the drill tasks, pushing the object to be more reachable did not reliably lead to contact that was informative about the pose, and it did much worse than our full method and the InfoOnly baseline.

A common failure case for all methods was pushing the object to be outside the robot's reachable region. The performance gain of the full method against the InfoOnly ablation can be mostly attributed to preventing this. As long as the object was kept within reach and contacts kept being made with the object at different locations, the pose estimation was gradually improved. This is illustrated in ReachOnly's performance on the sim box tasks, where it is one of the slowest methods to reduce nll, but was still able to achieve a relatively high number of successful trials.

The Slide baseline exhibited behavior that in some ways was the opposite of ReachOnly's. It always pushed the object away from the robot, and it became a race of it gathering enough pose-identifying information from those contacts before the object moved out of reach. On the real robot, sometimes it did not register that a contact was made and would continue pushing forward. This strategy's success was highly configuration-dependent, seen in Tab. V, where it can either achieve reliable success (since there is only randomness in the sliding direction), or no success. This strategy however does often lead to it being the quickest method to make contact with the object, giving it low cumulative $nll(\mathcal{X})$ for the sim drill and box tasks.

The GPVR baseline's performance can be compared against the InfoOnly ablation's, as neither have an explicit cost for avoiding the object from being pushed out. As seen in Fig. 17, the highest $\mathbf{var}(\mathbf{x}|\mathcal{X})$, even when given points augmented using shape information, does not match where intuitively information about the shape might be held. A similar problem was present in the drill tasks, where GPVR does very poorly because the task requires making multiple contacts close together, such as on either side of the drill head. Upon making contact with one side, the proximity of observed surface points

TABLE V: Number of successful trials after 10 runs of active rummaging for pose estimation in different tasks in Fig. 10 and Fig. 12. Success is defined as the run achieving a minimum nll below the threshold defined in Tab. III. The most and 1 success below the most successes in each section are bolded.

Task	Ours	InfoOnly	ReachOnly	GPVR	Slide
sim mug 0	**9**	3	7	2	0
sim mug 1	**9**	5	**8**	0	0
sim mug 2	**10**	6	**10**	7	0
mug total	**28**	14	25	9	0
sim drill 0	**10**	**9**	5	1	6
sim drill 1	**9**	**9**	7	0	**10**
sim drill 2	**7**	**7**	**7**	0	0
drill total	**26**	**25**	19	1	16
sim box 0	**9**	**9**	**8**	4	0
sim box 1	**10**	**9**	**10**	4	**10**
sim box 2	**6**	**6**	4	1	1
box total	**25**	**24**	22	9	11
real mug	**7**	0	4	0	1
real box	**7**	3	0	2	3

TABLE VI: Median cumulative $nll(\mathcal{X})$ for 10 runs of active rummaging for pose estimation in different tasks in Fig. 10 and Fig. 12. The best and any 5% within the best are bolded.

Task	Ours	InfoOnly	ReachOnly	GPVR	Slide
sim mug 0	**925**	1447	1691	1691	2478
sim mug 1	**1447**	1598	1591	2252	2610
sim mug 2	1013	**827**	1104	1239	2467
mug total	**3385**	4444	4386	5182	7555
sim drill 0	**7975**	10425	15401	22725	**8369**
sim drill 1	13750	12473	21627	22835	**6049**
sim drill 2	**24479**	**23892**	27471	33419	69321
drill total	**46204**	**46790**	64499	78979	83739
sim box 0	12744	12811	**10718**	24276	16527
sim box 1	8109	7412	6596	10183	**4601**
sim box 2	**23923**	24675	43254	31196	**23157**
box total	**44776**	**44898**	60568	65655	**44285**
real mug	**640**	1330	946	1145	4652
real box	9177	8341	8762	9994	**7717**

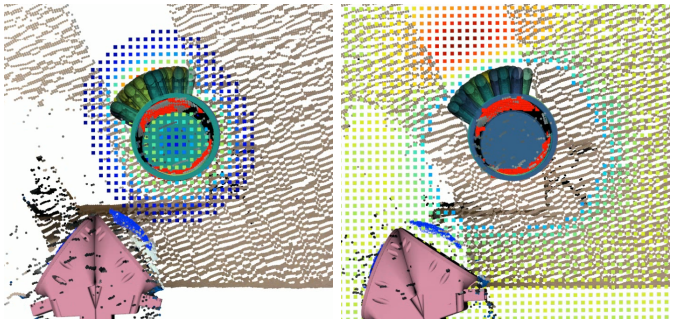


Fig. 17: Comparison of the fields to plan over evaluated at each $\mathbf{x} \in \mathcal{W}$ for (left) our method using $\tilde{I}(\mathbf{x}|\mathcal{X})$ against the (right) GPVR baseline using $\mathbf{var}(\mathbf{x}|\mathcal{X})$. $\mathbf{x}$ with too low $\tilde{I}$ or **var** are omitted.

lowers the GP variance around it, placing high cost on visiting the other side or the front of the drill, which was necessary to estimate its pose. This suggests that augmenting points is not a satisfactory way of conditioning on known object shape.

The clutter task results are in Fig. 19 where we see that our method performs the best on all tasks. Different scales for C_O were considered, with scale=1 relative to information gain performing the best. Having scale=0 led to pushing the other objects into the target object, while scale=5 led to avoiding pushing the non-target objects at all, making it impossible to do well on Clutter Hard.

L. Runtime Comparison

We also recorded the average computation time per step in the sim mug task and sim box task to highlight RUMI's computational efficiency in Tab. VII. Caching $p(S_{\mathbf{x}}|\mathcal{X})$ and $\tilde{I}(\mathbf{x}|\mathcal{X})$, and the dynamics were processes shared by all methods. All methods were implemented in PyTorch and accelerated by running on a modern computer with a NVIDIA RTX 4090 GPU. Computing $p(S_{\mathbf{x}}|\mathcal{X})$ and $\tilde{I}(\mathbf{x}|\mathcal{X})$ for $\mathbf{x} \in \mathcal{W}$ took 0.061s per step, while evaluating our cost inside the MPC took 0.178s for the sim mug. The time was dominated by evaluating $\hat{f}$ because it is stochastic and so benefited from sampling multiple state rollouts, in addition to dividing each step into 4 sequentially applied mini steps to avoid over-penetration. The GP fitting process also included caching $\mathbf{var}(\mathbf{x}|\mathcal{X})$ $\forall \mathbf{x} \in \mathcal{W}$ in a voxel grid to speed up inference inside the cost.

We considered how well the methods scale to the full 3D sim box task. The main challenge was the increased $\mathcal{W}$ size, with approximately 2.24 times more total points. Caching $p(S_{\mathbf{x}}|\mathcal{X})$ and $\tilde{I}(\mathbf{x}|\mathcal{X})$ slowed down to 0.387s, or an increase of 6.3 times, while our cost evaluation increased around 3 times to 0.556s. The GPVR cost run time scaled well because we were down sampling the workspace by 7 times the resolution for fitting the GP's free space.

Reducing the step size to no longer require dynamics mini steps, or using an alternative dynamics function would effectively improve the whole method's efficiency. Currently, RUMI can be run at around 1Hz, which was more than sufficient for quasi-static rummaging.

VI. Discussion and Future Work

A. Object Clutter

Qualitatively, the robot pushed multiple obstructing non-target objects to clear a path to the back of the mug for Clutter Hard, necessary for this task. This is a true interactive perception problem, rather than one arising from the lack of reliable and mobile vision. The good performance of RUMI on clutter tasks suggests potential for future practical applications. Only the target object mesh was known, with the number, placement, and shapes of the clutter objects being unknown. Critical to this success was the accurate initial clustering of the camera observation into object clusters, at least in separating the target from other objects, which may be more difficult in real environments with more noise. This required the objects to start sufficiently separated from each other, as well as for the objects to geometrically be sufficiently different from the target object. Non-geometric features such as color could be incorporated in this initial clustering step in future work.

Future work could also explore the total information gain provided by all object clusters; however, an obstacle to that is the fact that the information gain is unnormalized, so $\tilde{I}(\mathbf{x}|\mathcal{X}_k \cup \mathcal{X}_f)$ cannot be trivially summed for $k = 1..K$.

B. Limitations

In addition to requiring the target object to be geometrically distinct from the rest of the environment, another limitation is the need to avoid object toppling while exploring the environment. This places restrictions on object geometry and requires slow robot movement. Since after the initial visual observation we assume no reliable vision, we would have no way to sense and respond to object toppling. Incorporating mobile vision could be explored to respond to mitigate this.

C. Effects of Approximations

We used several approximations in the formulation of our method. Although it is difficult to quantitatively evaluate the effect of each, we provide a qualitative discussion to provide more insight into their effects. First, the use of reverse KL instead of forward KL in Eq. 14. We note that the difference is only significant in regions where I_r and I_f are high, and then only in relative magnitude. They predominantly agree categorically on which regions have near 0 information gain or significant information gain, so we expect this approximation to not affect exploration performance. Secondly, there is the

TABLE VII: Run time for different processes of our method and the GPVR baseline per execution step of the planar sim mug 0 and the 3D sim box 0 task across 10 runs. Standard deviation is in parenthesis.

process	average time per step (s)	
	sim mug	sim box
cache $p(S_{\mathbf{x}}|\mathcal{X})$, $\tilde{I}(\mathbf{x}|\mathcal{X})$	0.061 (0.001)	0.387 (0.002)
dynamics	0.724 (0.012)	1.242 (0.054)
our cost lookup	0.178 (0.002)	0.556 (0.021)
GP fit	1.870 (0.066)	2.541 (0.061)
GP cost lookup	0.116 (0.002)	0.184 (0.004)

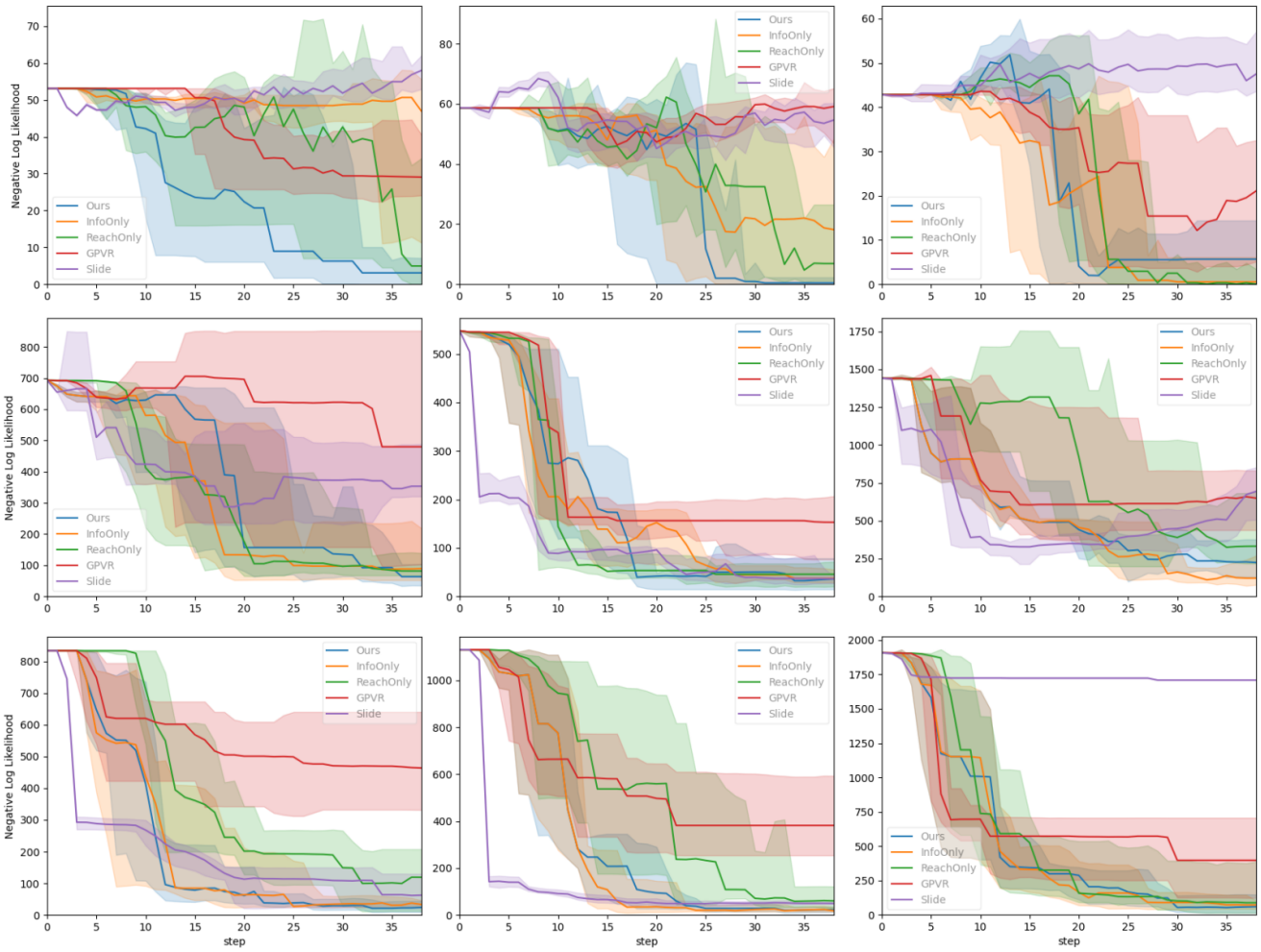


Fig. 18: $nll(\mathcal{X})$ after each execution step for simulation tasks depicted in Fig. 10. The median over 10 runs is plotted, with the 25^{th} to 75^{th} percentile shaded. From top to bottom we have the sim mug, sim drill, and sim box tasks. From left to right we have configuration 0, 1, and 2.

dropping of normalizing constants in Eq. 20. Their contribution is controlled by λ; with a lower λ the approximation accuracy goes down because the ignored term becomes relatively larger, but having too high λ leads to particle degeneracy in the particle filter approximation of the pose posterior. Finally, there is the particle filter approximation itself. From Fig. 5 we qualitatively see the effect of increasing P, which is to reduce the variance of the approximation. Beyond $P = 100$, we found little practical downstream benefit in exploration performance.

D. Unknown Object Shape

The last point of improvement is to relax our knowledge of the object from having its SDF to just having a class label. One naive approach is to use a template SDF for each object class and absorb the SDF uncertainty into the sensor model $\mathbf{p}(S_{\mathbf{x}}|\texttt{sdf}(\mathbf{Tx}))$. However, this fails with object classes that have high geometric variation. One possible approach would be to extend the pose posterior particle filter to also represent object shape, such that each particle is both a pose and a shape. The shape could be parameterized by recent advances in 3D representations such as the Deformed Implicit Field [23] that allows shape editing by constraining on surface points.

VII. Conclusion

We presented RUMI, an active exploration method based on the mutual information between a movable target object's uncertain pose and the robot trajectory. It maintains an explicit belief over the object pose using a particle filter, updating it with observed point clouds augmented with semantics, such as whether a point is in free space or on the object surface. Given object SDF, we formulated an information gain cost function evaluating the expected KL divergence between the pose distribution before and after executing a robot trajectory. In addition, we implemented a reachability cost function and showed that it was important to prevent pushing the object outside the robot's reachable region. Through comparison with baselines in real and simulated experiments, we showed that RUMI could effectively and efficiently condition on object shape to explore and estimate object pose. Additionally, we demonstrated potential for practical applications with tasks where an unknown number of unknown movable objects were cluttered around the target object.

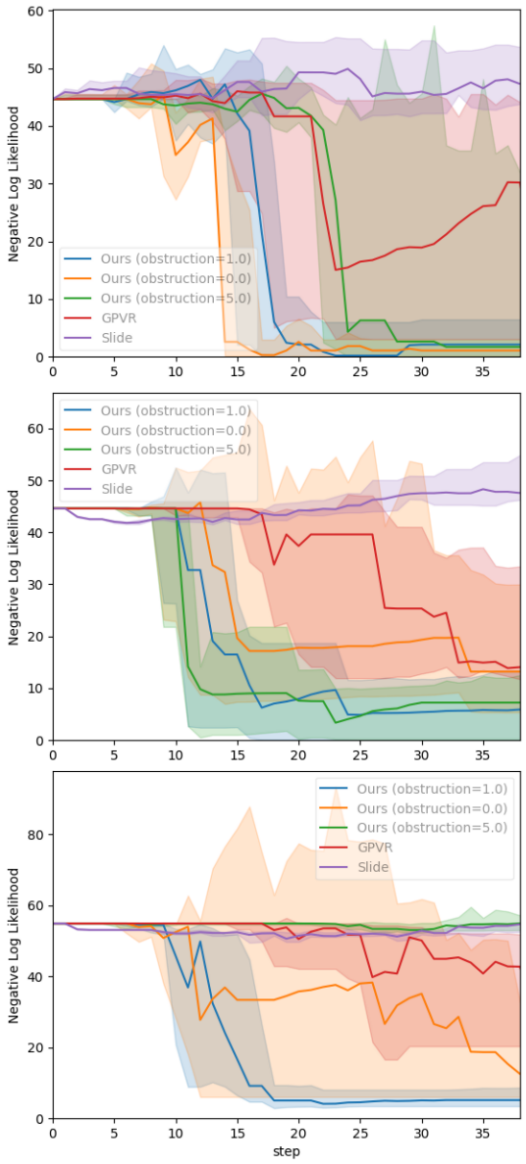


Fig. 19: $nll(\mathcal{X})$ after each execution step for clutter tasks depicted in Fig. 15. The median over 10 runs is plotted, with the 25^{th} to 75^{th} percentile shaded. From top to bottom we have Clutter Easy, Clutter Medium, and Clutter Hard.

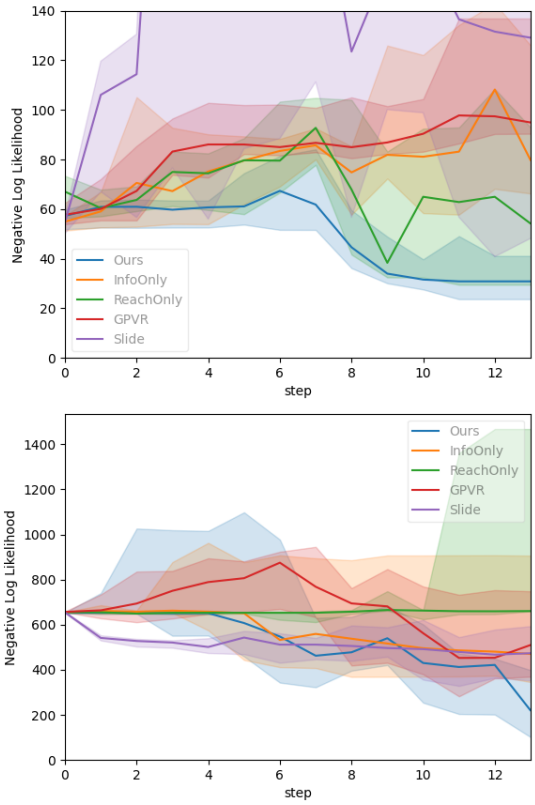


Fig. 20: $nll(\mathcal{X})$ after each execution step for the (top) real mug and (bot) real box tasks depicted in Fig. 12. The median over 10 runs is plotted, with the 25^{th} to 75^{th} percentile shaded.

References

[1] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42:177–196, 2018.

[2] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[3] Sergio Caccamo, Yasemin Bekiroglu, Carl Henrik Ek, and Danica Kragic. Active exploration using gaussian random fields and gaussian process implicit surfaces. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 582–589. IEEE, 2016.

[4] Chenghui Cai and Silvia Ferrari. Information-driven sensor path planning by approximate cell decomposition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(3):672–689, 2009.

[5] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[6] Nannan Cao, Kian Hsiang Low, and John M Dolan. Multi-robot informative path planning for active sensing of environmental phenomena: a tale of two algorithms. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 7–14, 2013.

[7] Henry Carrillo, Philip Dames, Vijay Kumar, and José A Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 487–494. IEEE, 2015.

[8] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[9] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking. *IEEE Transactions on Robotics*, 37(5):1328–1342, 2021.

[10] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *2011 IEEE International Conference on Robotics and Automation*, pages 2845–2850. IEEE, 2011.

[11] Danny Driess, Peter Englert, and Marc Toussaint. Active learning with query paths for tactile object shape exploration. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 65–72. IEEE, 2017.

[12] Danny Driess, Daniel Hennes, and Marc Toussaint. Active multi-contact continuous tactile exploration with gaussian process differential entropy. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7844–7850. IEEE, 2019.

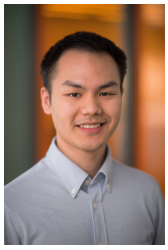
[13] Ryan Elandt, Evan Drumwright, Michael Sherman, and Andy Ruina. A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8238–8245. IEEE, 2019.

[14] Charles J Geyer. Practical markov chain monte carlo. *Statistical science*, pages 473–483, 1992.

[15] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[16] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Mutual information-based exploration on continuous occupancy maps. In *2015 IEEE/RSJ International Conference on Intelligent Robots and*

Systems (IROS), pages 6086–6092. IEEE, 2015.
[17] AK Jaiswal and B Kumar. Vacuum gripper-an important material handling tool. *Int J Sci Technol*, 7:1–8, 2017.
[18] Michael C Koval, Nancy S Pollard, and Siddhartha S Srinivasa. Pose estimation for planar contact manipulation with manifold particle filters. *The International Journal of Robotics Research*, 34(7):922–945, 2015.
[19] Michael Krainin, Brian Curless, and Dieter Fox. Autonomous generation of complete 3d object models using next best view manipulation planning. In *2011 IEEE international conference on robotics and automation*, pages 5031–5037. IEEE, 2011.
[20] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.
[21] Naveen Kuppuswamy, Alex Alspach, Avinash Uttamchandani, Sam Creasey, Takuya Ikeda, and Russ Tedrake. Soft-bubble grippers for robust and perceptive manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9917–9924. IEEE, 2020.
[22] Bhoram Lee, Clark Zhang, Zonghao Huang, and Daniel D Lee. Online continuous mapping using gaussian process implicit surfaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6884–6890. IEEE, 2019.
[23] Soomin Lee, Le Chen, Jiahao Wang, Alexander Liniger, Suryansh Kumar, and Fisher Yu. Uncertainty guided policy for active robotic 3d reconstruction using neural radiance fields. *IEEE Robotics and Automation Letters*, 7(4):12070–12077, 2022.
[24] Cindy Leung, Shoudong Huang, Ngai Kwok, and Gamini Dissanayake. Planning under uncertainty using model predictive control for information gathering. *Robotics and Autonomous Systems*, 54(11):898–910, 2006.
[25] Tiancheng Li, Miodrag Bolic, and Petar M Djuric. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal processing magazine*, 32(3):70–86, 2015.
[26] Ryan A MacDonald and Stephen L Smith. Active sensing for motion planning in uncertain environments via mutual information policies. *The International Journal of Robotics Research*, 38(2-3):146–161, 2019.
[27] Wolfram Martens, Yannick Poffet, Pablo Ramón Soria, Robert Fitch, and Salah Sukkarieh. Geometric priors for gaussian process implicit surfaces. *IEEE Robotics and Automation Letters*, 2(2):373–380, 2016.
[28] Joseph Masterjohn, Damrong Guoy, John Shepherd, and Alejandro Castro. Velocity level approximation of pressure field contact patches. *IEEE Robotics and Automation Letters*, 7(4):11593–11600, 2022.
[29] Leland McInnes, John Healy, Steve Astels, et al. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.
[30] Claudio Melchiorri. Slip detection and control using tactile and force sensors. *IEEE/ASME transactions on mechatronics*, 5(3):235–243, 2000.
[31] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *Proceedings of the AAAI conference on artificial intelligence*, volume 26, pages 2024–2030, 2012.
[32] Takato Miura, Naoki Akai, Kohei Honda, and Susumu Hara. Spline-interpolated model predictive path integral control with stein variational inference for reactive navigation. *arXiv preprint arXiv:2404.10395*, 2024.
[33] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.
[34] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, page 40, 2016.
[35] Rocco A Romeo and Loredana Zollo. Methods and sensors for slip detection in robotics: A survey. *Ieee Access*, 8:73027–73050, 2020.
[36] Lukas Rustler, Jens Lundell, Jan Kristof Behrens, Ville Kyrki, and Matej Hoffmann. Active visuo-haptic object shape completion. *IEEE Robotics and Automation Letters*, 7(2):5254–5261, 2022.
[37] Allison Ryan and J Karl Hedrick. Particle filter based information-theoretic active sensing. *Robotics and Autonomous Systems*, 58(5):574–584, 2010.
[38] Brad Saund, Sanjiban Choudhury, Siddhartha Srinivasa, and Dmitry Berenson. The blindfolded robot: A bayesian approach to planning with contact feedback. In *The International Symposium of Robotics Research*, pages 443–459. Springer, 2019.
[39] Robert Sim and Nicholas Roy. Global a-optimal robot exploration in slam. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 661–666. IEEE, 2005.
[40] Edward Smith, David Meger, Luis Pineda, Roberto Calandra, Jitendra Malik, Adriana Romero Soriano, and Michal Drozdzal. Active 3d shape reconstruction from vision and touch. *Advances in Neural Information Processing Systems*, 34:16064–16078, 2021.
[41] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.
[42] Cyrill Stachniss and Wolfram Burgard. Exploring unknown environments with mobile robots using coverage maps. In *IJCAI*, volume 2003, pages 1127–1134, 2003.
[43] Sudharshan Suresh, Zilin Si, Stuart Anderson, Michael Kaess, and Mustafa Mukadam. Midastouch: Monte-carlo inference over distributions across sliding touch. *arXiv preprint arXiv:2210.14210*, 2022.
[44] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260, 2003.
[45] Emanuele Vespa, Nikolay Nikolov, Marius Grimm, Luigi Nardi, Paul HJ Kelly, and Stefan Leutenegger. Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, 2018.
[46] Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.
[47] Zhengkun Yi, Roberto Calandra, Filipe Veiga, Herke van Hoof, Tucker Hermans, Yilei Zhang, and Jan Peters. Active tactile object exploration with gaussian processes. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4925–4930. IEEE, 2016.
[48] Yufeng Zhang, Jialu Pan, Li Ken Li, Wanwei Liu, Zhenbang Chen, Xinwang Liu, and Ji Wang. On the properties of kullback-leibler divergence between multivariate gaussian distributions. *Advances in Neural Information Processing Systems*, 36, 2024.
[49] Sheng Zhong, Nima Fazeli, and Dmitry Berenson. Soft tracking using contacts for cluttered objects to perform blind object retrieval. *IEEE Robotics and Automation Letters*, 7(2):3507–3514, 2022.
[50] Sheng Zhong, Dmitry Berenson, and Nima Fazeli. Chsel: Producing diverse plausible pose estimates from contact and free space data. In *Robotics: Science and Systems*, 2023.



Sheng Zhong received the B.Sc. degree in engineering science, robotics from the University of Toronto, Toronto, Canada, in 2018, and the Ph.D. degree in robotics from the University of Michigan, Ann Arbor, MI, USA in 2024.

His research interests include contact rich manipulation, parallelizable algorithms, and model predictive control.

Nima Fazeli is an Assistant Professor of Robotics, Computer Science (EECS), and Mechanical Engineering at the University of Michigan, and an Amazon Scholar with Amazon Robotics. He leads the Manipulation and Machine Intelligence (MMint) Lab, which focuses on intelligent and dexterous robotic manipulation through advances in sensing, learning, and control. Nima received his Ph.D. from MIT in 2019, where he worked with Prof. Alberto Rodriguez. He earned his M.Sc. from the University of Maryland, College Park in 2014, where his research focused on modeling the human (and occasionally swine) arterial tree for applications in cardiovascular disease, diabetes, and cancer diagnosis. His work has been recognized with the NSF CAREER Award, support from the National Robotics Initiative and NSF Advanced Manufacturing, and the Rohsenow Fellowship. His research has also been featured in major media outlets including The New York Times, CBS, CNN, and BBC.

Dmitry Berenson is an Associate Professor in the Robotics Department at the University of Michigan, where he has been since 2016. Before coming to University of Michigan, he was an Assistant Professor at WPI (2012-2016). He received a BS in Electrical Engineering from Cornell University in 2005 and received his Ph.D. degree from the Robotics Institute at Carnegie Mellon University in 2011. He was also a post-doc at UC Berkeley (2011-2012). He has received the IEEE RAS Early Career Award and the NSF CAREER award. His current research focuses on robotic manipulation, robot learning, and motion planning.