Deliberative Object Pose Estimation in Clutter

Venkatraman Narayanan   Maxim Likhachev

***Abstract*— A fundamental robot perception task is that of identifying and estimating the poses of objects with known 3D models in RGB-D data. While feature-based and discriminative approaches have been traditionally used for this task, recent work on deliberative approaches such as PERCH and D2P have shown improved robustness in handling scenes with severe inter-object occlusions. These deliberative approaches work by treating multi-object pose estimation as a combinatorial search over the space of possible rendered scenes of the objects, thereby inherently being able to predict and account for occlusions. However, these methods have so far been restricted to scenes comprising only of known objects, and have been unable to handle extraneous clutter—a common occurrence in many real-world settings. This work significantly increases the practical relevance of deliberative perception methods by developing a formulation that: i) accounts for extraneous unmodeled clutter in scenes, and ii) provides object pose uncertainty estimates. Our algorithm is complete and provides bounded suboptimality guarantees for the cost function chosen to be optimized. Empirically, we demonstrate successful object recognition and uncertainty-aware localization in challenging scenes with unmodeled clutter, where previous deliberative methods perform unsatisfactorily. In addition, this work was used as part of the perception system by Carnegie Mellon University's Team HARP in the 2016 Amazon Picking Challenge.**

I. Introduction

Robots that interact with or monitor physical objects in the world typically require the identity and pose of such objects. The availability of economic RGB-D sensors such as the Microsoft Kinect coupled with the imminent need to robustly solve real-world problems such as warehouse perception has ushered in a new wave of research in the field of 3D object pose estimation. Particularly important in settings such as warehouse perception or low-cost manufacturing is that of object *instance* detection, where a priori 3D models of the objects are available (Fig. 1). The persisting challenges of multi-object instance detection are perhaps best captured in a survey of the recently concluded Amazon Picking Challenge [1, 2], where participants ranked perception as the most difficult aspect of the overall system, despite having a priori knowledge of exact object instances.

While much of the work in object instance detection has focused on feature-based discriminative methods, recent work has shown that generative methods [3, 4] can provide robust performance, especially in scenes where inter-object occlusions are prevalent. These methods treat model-based multi-object pose estimation as a combinatorial search over possible renderings of the scene, thereby inherently accounting for inter-object occlusions. Further, these methods provide strong guarantees on solution quality and completeness [4], and allow for better introspection compared to complex discriminative function approximators. These generative methods, despite the promise shown, suffer from two significant limitations. First, they assume that the input (point cloud/depth image) provided to the algorithm contains only the objects of interest and is void of extraneous unmodeled clutter, and second, they do not provide any uncertainty estimates for the object poses.

The Robotics Institute, Carnegie Mellon University, PA, USA {venkatraman,maxim} at cs.cmu.edu.

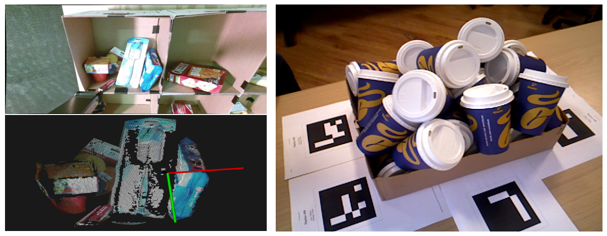

Fig. 1: *Left*: An example scene (RGB image and point cloud) from a warehouse picking task where we need to identify and localize a particular object with known 3D model (say the Oreo box), amidst the clutter of other objects for which we do not have 3D models. *Right*: Another example (from the dataset in [5]) in which 3D models of the coffee cups are available, but not that of the box in which they are placed. In such scenarios, pose estimation algorithms must be robust not only to occlusions between known objects, but also to occlusions by extraneous clutter.

This works addresses the aforementioned limitations, thereby significantly increasing their practical applicability. Specifically, we extend Perception via Search (PERCH) [3] through a novel formulation that

- Facilitates handling of extraneous clutter in the scene for which a priori models are unavailable. This obviates any need for scene pre-processing, clustering or segmentation, thereby allowing applicability in many more realistic situations.
- Produces uncertainty estimates for object poses in a Bayesian sense by reasoning over possible clutter conditions. Such uncertainty estimates are useful in situations where the perception system needs to be failure-aware, or in robotic systems that can actively control their sensing.

Our experimental evaluations on a real-world dataset demonstrate an improved accuracy compared to existing deliberative methods.

II. Related Work

A. Descriptor-based Methods

The most popular approaches for object *instance* detection in point clouds employ local or global 3D feature descriptors.

Approaches that use local descriptors follow a two step procedure: i) compute and find correspondences between a set of local shape-preserving 3D feature descriptors on the model and the observed scene and ii) estimate the rigid transform between a set of geometrically consistent correspondences. Examples of local 3D feature descriptors include Spin Images [6], Fast Point Feature Histograms (FPFH) [7], Signature of Histograms of Orientations (SHOT) [8] etc. Approaches that use global descriptors (e.g., VFH [9], CVFH [10], OUR-CVFH [11], GRSD [12] etc.) follow a three step procedure: i) build a database of 3D global descriptors on renderings corresponding to different viewpoints of each object during the training phase, ii) extract clusters belonging to individual objects in the test scene, and iii) match each cluster's 3D global descriptor to one in the database to obtain both identity and pose together. In both approaches, a final local optimization step such as Iterative Closest Point (ICP) [13] is often used to fine-tune the pose estimates. A comprehensive survey of descriptor-based methods is presented in [14].

Other discriminative approaches for object instance detection are based on template matching [15, 16], Hough forests [17] and deep neural networks trained on colorized versions of synthetically-generated or real depth images of object instances [18, 19].

B. *Generative Approaches*

Despite their speed and prevalence, a primary limitation of descriptor-based and discriminatively-trained methods is their brittleness to occlusions and other variations not captured during the training phase. Further, they are ill-suited for *multi-object* instance detection and pose estimation since the training data needs to capture the combinatorics of the problem (i.e, the features learnt must be capable of predicting inter-object occlusions for arbitrary combinations of objects). Global hypothesis verification methods [5, 20] perform a joint optimization over the individual predicted object poses, but still cannot guarantee completeness [4]—that a feasible solution (e.g., a non-collision constraint on the objects) will be returned if one exists.

Generative approaches address these issues by treating multi-object pose estimation as an optimization or filtering problem over possible renderings of the scene [3, 4, 21, 22]. This allows them to inherently account for inter-object occlusions, while maintaining completeness. Further, they do not require any semantic grouping/segmentation of points into "objects" as required by global descriptor approaches. Unfortunately, their practical applicability is limited due to restrictive assumptions. Specifically, our prior works Perception via Search (PERCH) [3] and D2P [4] assume that the input to the algorithm is "clean" and does not contain points belonging to clutter for which we do not have 3D models. In addition, these methods do not produce any pose uncertainty estimates as well.

In this work, we address the restrictive assumptions made in PERCH, thereby significantly increasing the practical relevance and applicability of generative methods. In addition, we present a technique to produce uncertainty estimates of object poses, which can be helpful for robotic systems that have active sensing, or in systems where critical decisions need to be made based on the confidence associated with perception outputs.

III. Background

A. *Problem Setup*

We first review the problem addressed in PERCH. The task is to simultaneously identify and localize multiple objects with known 3D models in a static depth image or point cloud. Formally, we are given the 3D models of N unique objects, an input point cloud I (which can also be generated from a depth image) containing $K \geq N$ objects, some of which may be duplicates of an unique instance, and the 6 DoF pose of the camera sensor along with its intrinsics. Similar to PERCH, we assume that the number (K) and type of objects in the scene are provided a priori and no "clustering" is required (i.e, the algorithm looks at the scene as a whole, rather than identifying and estimating the pose of each cluster in the scene). Unlike in PERCH however, we *do not* assume any pre-processing or that the points in the scene belong only to the objects of interest—i.e, there can be points corresponding to extraneous clutter for which we do not have 3D models. Note that we still continue to assume objects vary only in 3 DoF pose (x, y, yaw) with respect to their 3D model coordinate axes . This is primarily due to a tractability problem (larger search space with 6 DoF), as opposed to a theoretical bottleneck. In practice, we construct multiple 3D models of an object corresponding to "canonical" poses (stable configurations in the absence of other objects), and treat each of those as distinct objects. While this does not span the full 6 DoF space, it is a reasonable approximation in many real-world settings such as warehouse perception. Nevertheless, we will touch upon potential solutions to this problem in the discussion section.

B. *Notation*

We re-use notation from PERCH [3] for the most part, and introduce some additional ones. These are summarized in Table I. We use upper-case bold-faced letters to denote point clouds (set of points in $\mathbb{R}^3$), and lower-case bold-faced letters to denote a point in $\mathbb{R}^3$.

C. *Optimization Formulation*

PERCH formulates the problem of identifying and obtaining the poses of objects $O_1, O_2, \ldots, O_K$ as that of finding the minimizer of an "explanation" cost function which captures how well the rendered scene matches the input scenes, paying due attention both-ways—i.e, points in the input cloud should have an associated point in the rendered cloud, and vice-versa. Formally,

TABLE I: Symbols and Notation. *Left*: Notation from PERCH. *Right*: Notation introduced in this work.

$\boldsymbol{I}$	The input point cloud
K	The number of objects in the scene
N	The number of unique objects in the scene ($\leq K$)
O	An object state specifying a unique ID and 3 DoF pose
$\boldsymbol{R_j}$	Point cloud corresponding to the rendering of a scene with j objects $O_{1:j}$
$\boldsymbol{\Delta R_j}$	Point cloud containing points of R_j that belong exclusively to object O_j: $\boldsymbol{\Delta R_j = R_j - R_{j-1}}$
$V(O_j)$	The set of points in the volume occupied by object O_j.
V_j	The union of volumes occupied by objects $O_{1:j}$

$\boldsymbol{C}$	Point cloud containing points in $\boldsymbol{I}$ which are considered as "clutter" by the algorithm
$\boldsymbol{R_j}\vert\boldsymbol{C}$	Ditto as $\boldsymbol{R_j}$, but considering points in $\boldsymbol{C}$ as occluders
$\boldsymbol{\Delta R_j}\vert\boldsymbol{C}$	$(\boldsymbol{R_j}\vert\boldsymbol{C}) - (\boldsymbol{R_{j-1}}\vert\boldsymbol{C})$
$\boldsymbol{p} \prec \boldsymbol{P}$	Point $\boldsymbol{p}$ occludes some point in the point cloud $\boldsymbol{P}$
$\boldsymbol{C_j}$	Points in the clutter cloud $\boldsymbol{C}$ which occlude $\boldsymbol{R_j}$: $\boldsymbol{C_j} = \{\boldsymbol{p} \in \boldsymbol{C} : \boldsymbol{p} \prec \boldsymbol{R_j}\}$
$\boldsymbol{\Delta C_j}$	Points in the input cloud $\boldsymbol{I}$ which occlude $\boldsymbol{\Delta R_j}$: $\boldsymbol{\Delta C_j} = \{\boldsymbol{p} \in \boldsymbol{I} : \boldsymbol{p} \prec \boldsymbol{\Delta R_j}\}$

$$J(O_{1:K}) = \underbrace{\sum_{\boldsymbol{p}\in\boldsymbol{I}} \textsc{Outlier}(\boldsymbol{p}|\boldsymbol{R_K})}_{J_{observed}(O_{1:K}) \text{ or } J_o} + \underbrace{\sum_{\boldsymbol{p}\in\boldsymbol{R_K}} \textsc{Outlier}(\boldsymbol{p}|\boldsymbol{I})}_{J_{rendered}(O_{1:K}) \text{ or } J_r} \tag{1}$$

in which $\textsc{Outlier}(\boldsymbol{p}|\boldsymbol{P})$ for a point cloud $\boldsymbol{P}$ and point $\boldsymbol{p}$ is defined as follows:

$$\textsc{Outlier}(\boldsymbol{p}|\boldsymbol{P}) = \begin{cases} 1 & \text{if } \min_{\boldsymbol{p'}\in\boldsymbol{P}} \|\boldsymbol{p'} - \boldsymbol{p}\|_2 > \delta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where δ represents the sensor noise resolution.

While this optimization problem looks completely intractable at the outset due to the combinatorially large search space (joint poses of all objects), it was shown in PERCH that the cost function can be exactly decomposed over individual objects rather, subject to the constraint that objects are added in a non-occluding order. Intuitively, by enforcing the non-occluding order, PERCH is able to penalize points on an individual object for both J_o and J_r, since it is guaranteed that these points continue to exist at a later stage (i.e, they won't be occluded in the future when we add a new object). This allowed for the design of an efficient tree-search algorithm which automatically figures out the correct order in which to introduce objects in to the scene such that the $J(O_{1:K})$ is minimized.

IV. C-PERCH

A. Augmented Objective

The explanation cost used by PERCH is meaningful only when we want both the rendered and input point clouds to exactly match each other. In the presence of unmodeled clutter however, this fails on two counts: first, the rendered scene does not account for occlusion by the clutter (the algorithm assumes that all occlusions occur between objects with known 3D models), and second, the cost function (specifically $J_{observed}$) would unnecessarily penalize points in the input cloud which are extraneous clutter that do not belong to the objects of interest. To overcome these limitations, we first propose a formulation that allows the algorithm to explicitly pick and treat some points as clutter, and secondly demonstrate that this does not add any significant complexity to the existing optimization solved by PERCH.

In our proposed extension C-PERCH (Clutter-PERCH), we jointly optimize over the object poses $O_{1:K}$ *and* a variably-sized clutter cloud $\boldsymbol{C} \subseteq \boldsymbol{I}$. The latter allows the algorithm to mark certain points in the input scene as clutter, so that it can use those as extraneous "occluders" when rendering a scene with known 3D models. However, complete freedom to mark points as clutter could be disastrous: the optimal solution might just be to treat the input entire cloud as clutter, claim that the desired object(s) to be localized are completely occluded by the clutter, and thus incur no cost at all (since the rendered point cloud would be an empty cloud). To strike a balance between optimizing the explanation cost and allowing the algorithm to treat certain points as clutter, we introduce an additional term to the cost function that penalizes the algorithm for marking too many points as clutter—in other words, we would like to minimize the explanation cost while not marking too many points treated as clutter (conversely maximize the number of points observed on the objects of interest). Of course, the amount to penalize depends on the scenario at hand as well. In extreme clutter, it would be okay if the algorithm marks several points as clutter, but in scenes with no clutter, we really don't want to mark any point as clutter. We model this using a multiplicative factor α on the clutter penalty, to represent our uncertainty about the true nature of clutter in the scene. The augmented cost function to minimize is:

$$J_\alpha(O_{1:K}, C) = J_o(O_{1:K}, C) + J_r(O_{1:K}, C) + \alpha|\boldsymbol{C}| \tag{3}$$

$$J_o(O_{1:K}, C) = \sum_{\boldsymbol{p}\in\boldsymbol{I}\cap\boldsymbol{V_K}} \textsc{Outlier}(\boldsymbol{p}|(\boldsymbol{R_K}|\boldsymbol{C}))$$

$$J_r(O_{1:K}, C) = \sum_{\boldsymbol{p}\in\boldsymbol{R_K}|\boldsymbol{C}} \textsc{Outlier}(\boldsymbol{p}|\boldsymbol{I})$$

There are three changes from Eq. 1. First, both J_o and J_r use the cloud $\boldsymbol{R_K}|\boldsymbol{C}$ rather than simply $\boldsymbol{R_K}$ to explicitly acknowledge the occlusions caused by extraneous clutter. Second, the optimization objective has a penalty term for the number of points marked as clutter, weighed by α, a term which models the amount of true clutter in the scene. Third, J_o only penalizes points in the input cloud that fall within the volume of the modeled objects, rather than every point in $\boldsymbol{I}$. An illustration of the different point clouds used is presented in Fig. 2.

Algorithm 1 C-PERCH: Generation of Successor States and Edge Costs

```
1:  procedure GETSUCCESSORS($O_{1:j}$)
2:      $S \leftarrow \emptyset$
3:      $\Delta J \leftarrow \emptyset$
4:      // Iterate over objects not yet added to scene
5:      for all ID $\in$ {All Possible IDs} $\setminus$ IDs($O_{1:j}$) do
6:          // Iterate over all possible poses the new object can take
7:          for all pose $\in$ {All Discrete Poses} do
8:              $O_{j+1} = \{\text{ID}, pose\}$
9:              $s = O_{1:j} \cup O_{j+1}$
10:             if $s$ is not physically-plausible then
11:                 // Prune if objects in scene collide
12:                 continue
13:             $\boldsymbol{R_j}$ = render scene with $O_{1:j}$
14:             $\boldsymbol{R_{j+1}}$ = render scene with $O_{1:j+1}$
15:             $\boldsymbol{\Delta R_{j+1}} = \boldsymbol{R_{j+1}} - \boldsymbol{R_j}$
16:             if $\boldsymbol{\Delta R_{j+1}} \prec \boldsymbol{R_j}$ then
17:                 // Prune if new object occludes existing scene
18:                 continue
19:             $\boldsymbol{\Delta C_j}$ = Points in $\boldsymbol{I}$ which occlude $\boldsymbol{\Delta R_j}$
20:             Compute $\Delta J_o^{j+1}(O_{j+1}, \boldsymbol{\Delta R_{j+1}}, \boldsymbol{\Delta C_{j+1}})$   ▷ Eq.5
21:             Compute $\Delta J_r^{j+1}(\boldsymbol{\Delta R_{j+1}}, \boldsymbol{\Delta C_{j+1}})$   ▷ Eq.6
22:             Compute $\Delta J_{c,\alpha}^{j+1}(\boldsymbol{\Delta C_{j+1}})$   ▷ Eq.7
23:             $\Delta J_\alpha^{j+1} = \Delta J_o^{j+1} + \Delta J_r^{j+1} + \Delta J_{c,\alpha}^{j+1}$
24:             $S \leftarrow S \cup \{s\}$
25:             $\Delta J \leftarrow \Delta J \cup \{\Delta J_\alpha^{j+1}\}$
26:     return $\langle S, \Delta J \rangle$
```

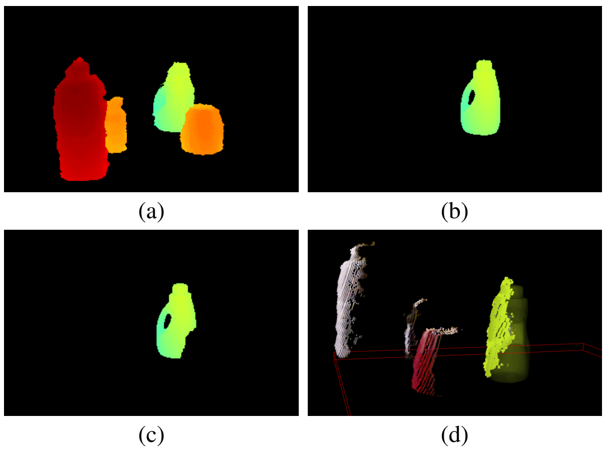

Fig. 2: Illustration of various notations used. (a) The input point cloud $\boldsymbol{I}$ (represented as a depth image and pseudo-colored). (b) Rendering $\boldsymbol{R_1}$ corresponding to a state with one object O_1 (c) Rendering $\boldsymbol{R_1}|\boldsymbol{\Delta C_1}$, considering points in I that occlude R_1 (d) A profile view of the same scene, showing the volume $V(O_1)$, and the points in I contained in it.

B. Tractability

With this formulation, it appears that we have made the problem intractable by introducing a variably-sized point cloud into the already combinatorial search space. However, it can be shown that $\boldsymbol{C}$ is only a dependent variable of $O_{1:K}$, and does not affect the decomposition used by PERCH under a reasonable constraint. Define $\boldsymbol{C}_{intrusive} = \{\boldsymbol{p} \in \boldsymbol{C} : \boldsymbol{p} \prec \boldsymbol{R_K}\}$, the set of clutter points which occlude the scene rendered by considering only the objects of interest, and $\boldsymbol{C}_{superfluous} = \boldsymbol{C} - \boldsymbol{C}_{intrusive}$. Quite clearly, we can replace $\boldsymbol{R_K}|\boldsymbol{C}$ with $\boldsymbol{R_K}|\boldsymbol{C}_{intrusive}$ in the optimization objective (since the superfluous clutter points do not affect the rendering of the objects). Since $\boldsymbol{C} = \boldsymbol{C}_{intrusive} \cup \boldsymbol{C}_{superfluous}$ and all terms except the penalty depend only on $\boldsymbol{C}_{intrusive}$, the optimal solution would involve setting $\boldsymbol{C}_{superfluous} = \emptyset$. Hereon, we simply set $\boldsymbol{C} = \boldsymbol{C}_{intrusive}$ to factor this observation in to account.

If we now require the algorithm to definitely mark every occluding input point as clutter (for a given rendered scene), we can simply drop $\boldsymbol{C}$ from the argument list of J_α, J_o and J_r because $\boldsymbol{C}$ is now purely a function of $\boldsymbol{R_K}$.

It was shown in PERCH that $\boldsymbol{R_K}$ can be constructed in a monotone fashion by introducing objects in a non-occluding order. Formally, $\boldsymbol{R_K} = \cup_{i=1}^{K} \boldsymbol{\Delta R_i}$, s.t, $\boldsymbol{R_{i-1}} \subseteq \boldsymbol{R_i}$. If we now define $\boldsymbol{\Delta C_j} = \{\boldsymbol{p} \in I : p \prec \boldsymbol{\Delta R_j}\}$ and follow a decomposition procedure similar to the one adopted in PERCH, we obtain:

$$
\begin{aligned}
J_\alpha(O_{1:K}) &= \sum_{i=1}^{K} \Delta J_\alpha^i && \text{s.t. } R_{i-1} \subseteq R_i \\
&= \sum_{i=1}^{K} \Delta J_o^i + \Delta J_r^i + \Delta J_{c,\alpha}^i && \text{s.t. } R_{i-1} \subseteq R_i
\end{aligned} \tag{4}
$$

where

$$\Delta J_o^i = \sum_{\boldsymbol{p} \in \{\boldsymbol{I} \cap V(O_i)\}} \text{OUTLIER}(\boldsymbol{p}|(\boldsymbol{\Delta R_i}|\boldsymbol{\Delta C_i})) \tag{5}$$

$$\Delta J_r^i = \sum_{\boldsymbol{p} \in \boldsymbol{\Delta R_i}} \text{OUTLIER}(\boldsymbol{p}|\boldsymbol{I}) \tag{6}$$

$$\Delta J_{c,\alpha}^i = \alpha|\boldsymbol{\Delta C_i}| \tag{7}$$

The end result is that we can still maintain the decomposition of the objective function over individual objects despite introducing the clutter cloud in to the optimization process. Similar to PERCH, we solve the final optimization as a discrete tree-search under the constraint that objects are added in a non-occluding order to the Monotone Scene Generation Tree [3]. The complete procedure followed to generate successor states for a parent state, and the corresponding edge costs is presented in Alg. 1.

C. Pose Uncertainty Estimates

We earlier mentioned how the factor α models the amount of clutter expected in the scene. Low values correspond to scenes with high anticipated clutter (where the penalty for marking points as clutter is low), and vice versa. Solving the optimization problem for different values of α yield potentially distinct solutions:

$$O_{1:K}^{\alpha} = \underset{O_{1:K}}{\text{argmin}}\, J_\alpha(O_{1:K}) \tag{8}$$

Immediately, there is an opportunity to produce uncertainty estimates for the object poses based on the uncertainty in how much clutter there exists in the scene. If $p(\alpha)$ denotes the prior for α (which may be obtained by a priori analysis of the scene or assumed to be uniform), then the density

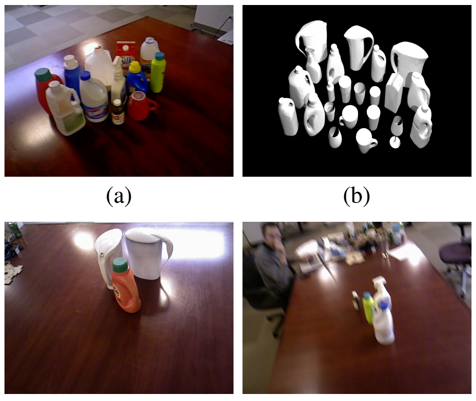
Fig. 3: (a) and (b) A subset of objects in the RGB-D occlusion dataset. (c) and (d) Representative test scenes from the dataset.

estimate for the object poses (represented by the random variable Ω) is given by

$$\begin{aligned} p(\Omega = O_{1:K}) &= \int_{\alpha} p(\Omega = O_{1:K}, \alpha) d\alpha \\ &= \int_{\alpha} p(\Omega = O_{1:K} | \alpha) p(\alpha) d\alpha \\ &= \int_{\alpha} \mathbb{1}\left(O_{1:K} = O^{\alpha}_{1:K}\right) p(\alpha) d\alpha \end{aligned} \tag{9}$$

Eq. 9 is hard to solve in closed form, but sampling from the distribution is trivial: we sample an α from $p(\alpha)$, and solve the optimization problem in Eq. 8 to get a solution, which is a sample from $p(\Omega)$. Intuitively, if we find out that several values of α lead to the same solution for $O_{1:K}$, it implies that the scene clutter model has minimal effect on the object poses and we can therefore be confident about our pose estimate. On the other hand, if solutions are distinct for closely related values of α, we would have greater uncertainty in our object poses due to a spread-out distribution.

V. Experiments

A. Robustness to Clutter

Experiment Setup. We evaluate C-PERCH on the occlusion dataset of Aldoma et al. [14] (Fig. 3, reproduced from [4]) which contains real-world RGB-D images of scenes with multiple objects on a tabletop. The dataset contains 3D models of 36 household objects and 22 RGB-D scenes with 80 object instances in total. In our experiments, we use only the depth image (ignoring RGB) for all methods. To test the ability of C-PERCH to handle extraneous clutter in the scene, we setup evaluation such that C-PERCH and the baselines are required to identify and localize exactly one object in the scene, treating the others as clutter. The process is repeated for every object in each scene.

For both PERCH and C-PERCH, we use a discretization of 0.05 m for translation and 22.5 degrees for yaw. Note that in this dataset, objects vary only in yaw with respect

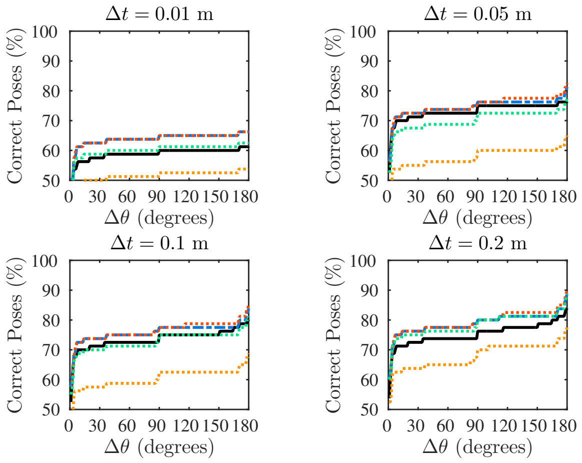

Fig. 4: Comparison of PERCH (■) with C-PERCH for different values of α: 1 (🟦) 0.5 (🟥) 0.25 (🟩) and 0 (🟧), and for different correctness measures. We omit $\alpha = 0.75$ since it yielded identical results to $\alpha = 1$.

to the 3D models. We also use a locally-constrained ICP refinement (with a maximum of 20 iterations) for every rendered state in both PERCH and C-PERCH to compensate for discretization artifacts. Similar to [4], we measure accuracy of an algorithm by counting the number of objects that fall within a given error bound: an estimated object pose (x, y, θ) is marked 'correct' if $\|(x, y) - (x_{\text{true}}, y_{\text{true}})\|_2 < \Delta t$ and $\textsc{ShortestAngularDifference}(\theta, \theta_{\text{true}}) < \Delta\theta$. The latter check is ignored for rotationally symmetric objects.

Accuracy Comparisons. Figure 4 compares the performance of PERCH (baseline) with C-PERCH configured with different values of α, the clutter model parameter. All experiments were run on an m4.10x Amazon AWS instance. The first takeaway is that C-PERCH consistently outperforms PERCH, for $\alpha \in \{1, 0.5, 0.25\}$. This supports our hypothesis that modeling clutter explicitly in the optimization formulation will lead to better performance. A second observation is that C-PERCH with $\alpha = 0.5$ performs marginally better than $\alpha = 1$, and significantly better than $\alpha = 0$. This indicates that there is no one "correct" way to pick α unless we have some prior information about the clutter conditions. In some sense, being cautious ($\alpha = 0.5$) yields the best performance across a variety of scenes. Since PERCH was shown to outperform OUR-CVFH [11], a global descriptor, as well a Brute-Force ICP baseline, we omit similar comparisons here.

Timing. The average time taken by PERCH for a scene was 19.17 s and for C-PERCH (across all values of α) was 17.98 s. While both methods generated the same number of scenes on average (792.31), PERCH takes slightly longer than C-PERCH since it computes the observed cost (J_o) over all points in the input point cloud, as opposed to C-PERCH which only looks at input points within the volumes of the assigned objects.

B. Illustration of Uncertainty Estimation

Next, we provide an example that demonstrates how C-PERCH can be used to generate uncertainty estimates for an object pose. In Fig. 5, the object desired to be localized

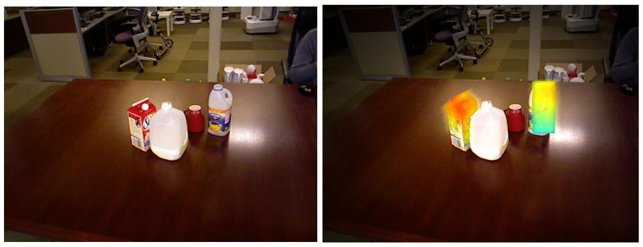
Fig. 5: Example that demonstrates how C-PERCH can be used to obtain pose uncertainty estimates, including multimodal distributions. The object to be localized in this scene (*left*) is the milk carton (for which we have a 3D model) and the other objects are considered as extraneous clutter (no models available). C-PERCH yields two distinct solutions across multiple values of α, which are overlaid on top of the input RGB image (*right*).

is the milk carton, which is partially occluded by the milk jug. Large flat portions on the milk carton as well as on the Odwalla jug (far back on the right) cause some ambiguity to the algorithm since it deals only with the depth image (RGB is not used). If we proceed to estimate the object pose uncertainty (Eq. 9), by running the optimization for values of $\alpha \in [0, 1]$ in steps of 0.01 (i.e., assuming an uniform prior for α), we observe that only two distinct solutions turn up, corresponding to the ranges $[0, 0.21]$ and $(0.21, 1]$. The pose uncertainty distribution can thus be represented as a particle distribution with two particles of weight ~ 0.2 and ~ 0.8 respectively, with the former corresponding to the partially occluded configuration.

VI. Conclusions

In summary, we presented C-PERCH, an extension to Perception via Search (PERCH), that allows deliberative perception algorithms to operate in scenes with unmodeled extraneous clutter. This significantly extends their practical relevance to real-world scenarios where 3D models cannot be obtained for every object in the scene. In addition, we also showed how C-PERCH can produce pose uncertainty estimates by reasoning about the amount of clutter in the scene. This is useful to systems with active sensing, and when introspection capabilities are required.

The main drawback of deliberative approaches at present is their inability to tractably handle the joint 6 DoF poses (as opposed to 3 DoF) of multiple objects. Our future work in this area is to investigate the use of physics simulators to admissibly prune large infeasible portions of the search space, thereby maintaining tractability. A principled approach to integrate color information with depth also remains to be studied in the context of deliberative perception.

Acknowledgment

This research was sponsored by ARL, under the Robotics CTA program grant W911NF-10-2-0016.

References

[1] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Lessons from the Amazon Picking Challenge," *arXiv preprint arXiv:1601.05484*, 2016.
[2] R. Jonschkowski, C. Eppner, S. Höfer, R. Martín-Martín, and O. Brock, "Probabilistic multi-class segmentation for the amazon picking challenge," in *IROS*, 2016.
[3] V. Narayanan and M. Likhachev, "PERCH: Perception via Search for Multi-Object Recognition and Localization," in *ICRA*. IEEE, 2016.
[4] V. Narayanan and M. Likhachev, "Discriminatively-guided Deliberative Perception for Pose Estimation of Multiple 3D Object Instances," in *Robotics: Science and Systems*, 2016.
[5] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd."
[6] A. E. Johnson and M. Hebert, "Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes," *PAMI*, vol. 21, no. 5, pp. 433–449, 1999.
[7] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration," in *ICRA*. IEEE, 2009.
[8] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010.
[9] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram," in *IROS*. IEEE, 2010.
[10] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, and G. Bradski, "CAD-model Recognition and 6DOF Pose Estimation using 3D Cues," in *ICCV Workshops*. IEEE, 2011.
[11] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, "OUR-CVFH–Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation," in *DAGM*, 2012.
[12] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2D–3D Categorization and Classification for Multimodal Perception Systems," *IJRR*, 2011.
[13] Y. Chen and G. Medioni, "Object Modeling by Registration of Multiple Range Images," in *ICRA*, 1991.
[14] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Point Cloud Library," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/12, 2012.
[15] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model Based Training, Detection and Pose Estimation of Texture-less 3D Objects in Heavily Cluttered Scenes," in *ACCV*, 2013, pp. 548–562.
[16] P. Wohlhart and V. Lepetit, "Learning Descriptors for Object Recognition and 3D Pose Estimation," in *CVPR*, 2015.
[17] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class Hough Forests for 3D Object Detection and Pose Estimation," in *ECCV*, 2014, pp. 462–477.
[18] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D Object Recognition and Pose Estimation Based on Pre-trained Convolutional Neural Network Features," in *ICRA*. IEEE, 2015.
[19] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal Deep Learning for Robust RGB-D Object Recognition," in *IROS*, 2015.
[20] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A Global Hypotheses Verification Method for 3D Object Recognition," in *ECCV*, 2012, pp. 511–524.
[21] M. R. Stevens and J. R. Beveridge, "Localized Scene Interpretation from 3D Models, Range, and Optical Data," *Computer Vision and Image Understanding*, 2000.
[22] Z. Sui, O. C. Jenkins, and K. Desingh, "Axiomatic particle filtering for goal-directed robotic manipulation," in *IROS*.