Fast and High-Quality GPU-based Deliberative Perception for Object Pose Estimation

Aditya Agarwal

CMU-RI-TR-20-22

June 29, 2020

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Maxim Likhachev, Chair
Oliver Kroemer
Paloma Sodhi

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Pose estimation of known objects is fundamental to tasks such as robotic grasping and manipulation. The need for reliable grasping imposes stringent accuracy requirements on pose estimation in cluttered, occluded scenes in dynamic environments. Modern methods employ large sets of training data to learn features and object templates in order to find correspondence between models and observed data. However these methods require extensive annotation of ground truth poses. An alternative is to use algorithms that search for the best explanation of the observed scene in a space of possible rendered scenes. A recently developed algorithm, PERCH (PErception Via SeaRCH) does so by using depth data to converge to a globally optimal solution using a search over a specially constructed tree. While PERCH offers strong guarantees on accuracy, the current formulation suffers from low scalability owing to its high runtime. In addition, the sole reliance on depth data for pose estimation restricts the algorithm to scenes where no two objects have the same shape.

In this work, we propose PERCH 2.0, a deliberative pose estimation approach that takes advantage of GPU acceleration and RGB data. We show that our approach can achieve an order of magnitude speedup over PERCH and meets scalability requirements for evaluating thousands of poses in parallel. We demonstrate that the proposed work directly allows for an extension of deliberative pose estimation methods to new domains such as object articulation, conveyor picking and 6-Dof pose estimation. Our combined deliberative and discriminative framework for 6-DoF pose estimation achieves a higher accuracy than purely data-driven approaches without the need for any ground truth pose annotation.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Maxim Likhachev for his extraordinary guidance and support throughout the research work. His constant encouragement has been immensely valuable in keeping me motivated and will continue to do so in the near future. I would also like to express my gratitude towards Prof. Oliver Kroemer and Paloma Sodhi for their valuable feedback as members of my thesis committee.

I would like to thank my colleagues Yupeng Han and Shanshan Xie who worked with me on different aspects of this research and all members of Search Based Planning Lab for their contribution to maintaining a conducive research environment in the lab.

Contents

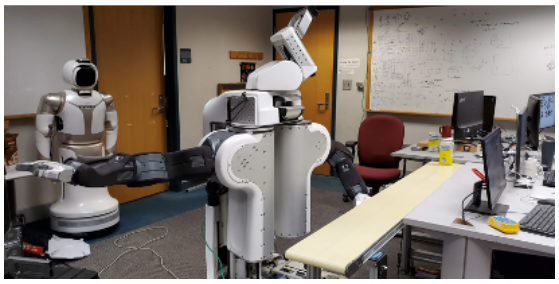
List of Figures

List of Tables

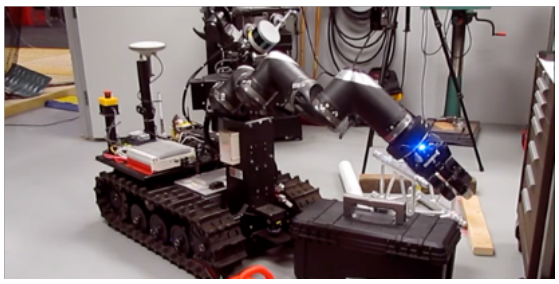
Chapter 1

Introduction

1.1 Motivation



(a) Conveyor Picking



(b) Object Articulation

(c) Table top robotic manipulation

(d) Bin picking

Fig. 1.1: Examples of robotic manipulation scenarios

For robots to operate successfully outside controlled environments they need to be able to interact with objects in a safe and reliable manner. Such interaction requires correct identification of object categories as well as their location and orientation in the 3D world. For instance, industrial robotic manipulators can grasp objects with high precision but only when the location and orientation of the object (or the object pose) in the 3D world is known. Such manipulators are often deployed in settings such as an automated warehouse where the objects of interest could

either be at rest in shelves or in bins or they could be in motion, such as on a conveyor. In either scenario, robust recognition and pose estimation of the objects in the 3D world could greatly enhance the capabilities of a robotic manipulator deployed for pick and place of such objects. The knowledge of the object pose allows the underlying planning system to retain the flexibility of choosing the right grasp for picking up the object while avoiding collisions with other objects and the environment.

Pick and place tasks performed by full body mobile manipulators make their appearance in household environments as well. The objects could be located on a table top, on shelves or within household appliances such as dishwashers, refrigerators etc. More recently, research has also focused on object articulation with the help of mobile robotic manipulators, especially in situations where such articulation would be dangerous for human beings (such as in an active war zone). One such articulation task that involved opening of a crate and removal of the object within for inspection was the focus of the RCTA project (Figure 1.1b), a collaboration between eminent universities and industry leaders.

A key difference in the domains described above is the degree of structure present in them. Objects present in automated warehouses and grocery stores on shelves are often arranged in a predecided orientation (such as upright or lying down). In addition the location of the object in the store, can be used to provide an idea of the identity or category of the object. Similarly in the case crate opening, the crate to be located can vary only in 3 degrees of freedom that is ($x, y,$ yaw). However in other domains such as household environments, the objects could be found in random orientations and locations, varying in all 6 degrees of freedom ($x, y, z,$ roll, pitch, yaw). The identity of the objects would also be independent of their location.

Despite differing domain dependant requirements for interaction with objects, several challenges in the underlying pose estimation task are common. Objects are often occluded by other objects or by unkown obstacles present in the environment. The manipulator itself could be fixed or dynamic (such as when present on a mobile base), the latter requiring the algorithm to be robust to different background scenes and camera viewpoints. At the same time, in several scenarios described previously, a 3D model of objects of interest is available or can be constructed through laser scanning. For instance, in an automated manufacturing warehouse, the inventory of objects is known and regularly updated. While constructing a 3D model is often feasible, continually updating a large annotated dataset with new objects being introduced could be difficult.

1.2 Existing Approaches

Early work in pose estimation focused on detecting features from available 3D object models and matching those to the observed scene. Feature-based methods [7, 8, 9] typically require rich textures to be present on objects and even when features are present, fail to find good estimates when objects are occluded. Moreover, estimating the pose of each object in isolation may not lead to a globally feasible and optimal solution that fully explains the observed scene [10].

Following the success of convolutional neural networks (CNNs) on 2D object detection, they have also been explored to estimate object poses in 3D space [6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. However, these methods require large sets of training data to be able to estimate poses accurately. Dataset sizes scale poorly with number of objects since networks need to be trained on multiple objects from as many viewpoints as possible per object with varying degrees of inter-object occlusions. Moreover, the task of annotating the training dataset with ground truth poses is non-trivial and often labour intensive.

Methods that rely on synthesizing scenes and matching these with observed scenes [1, 2, 3, 26, 27] overcome shortcoming of feature and learning based methods but tend to be slow. Specifically, Perception Via Search or PERCH [1, 2, 3] is a recent work that introduces a global matching objective function and does such a search in a globally efficient manner.

1.3 Proposed Approach

This work focuses on pose estimation of objects for a variety of domains that require interaction with objects. Building on prior work on deliberative methods such as PERCH [1, 2, 3], it seeks to combine advancements in computer vision, search and optimization techniques with advancements in computing hardware to devise an approach that scales with different domain requirements. The primary focus of our proposed approach (referred to as PERCH 2.0) remains on the accuracy of object pose estimation since it has a direct positive impact on robotic manipulation success. Our approach also seeks to exploit structure, wherever present in the environment for improved speed and reduced resource requirements such as the need to collect and annotate large training datasets.

The key contributions of our work can be listed as follows :

- Incorporation of RGB sensor data into the objective function used by PERCH, allowing the algorithm to handle scenarios where depth data alone is not sufficient to estimate the object poses
- PERCH 2.0 : A fully parallel and scalable GPU-based deliberative pose estimation methodology that achieves significant speedup over previous deliberative methods like PERCH
- Demonstration of the application of PERCH 2.0 for pose estimation in new domains such as object articulation and conveyor picking
- A combined discriminative and deliberative framework for 6-Dof pose estimation that eliminates the need for ground truth pose annotation in the training data and overcomes other shortcomings of deliberative methods

Chapter 2

Related Work

In terms of methodology, approaches that address pose estimation of objects in 3D space can be broadly divided into 3 categories : discriminative methods, deliberative methods and optimization methods.

2.1 Discriminative Methods

Discriminative approaches traditionally used hand-crafted local 3D features to establish 2D to 3D correspondences between the observed image and the 3D model and recover the object pose [7, 8, 9, 28, 29, 30, 31, 32]. Other traditional approaches computed similarity scores over regions of observed images with an object template (obtained by rendering 3D models) to obtain the best match and corresponding pose [33, 34, 35]. Feature-based methods typically require rich textures to be present on objects and even when features are present, fail to find good estimates when objects are occluded. Moreover, estimating the pose of each object in isolation may not lead to a globally feasible and optimal solution that fully explains the observed scene [10].

However recent advancements in deep learning has led to 2D object detectors being extended for the task of 6-Dof pose estimation [6, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. These approaches can further be divided into methods that directly regress to 6-Dof pose coordinates and rotations [6, 36, 37], methods that localize object keypoints [12, 13, 16, 18, 19, 20, 24] in the image space or methods that discretize the 6D space and then score each discretized pose using a classifier [21, 25]. Regressing directly to poses, ties the pose estimation to camera intrinsics, thus introducing errors if the camera is changed. Localization to image keypoints often results in ambiguities for objects with symmetries or requires explicit handling of symmetries. Scoring discretized poses [22, 25] is independent of camera parameters and object symmetries but requires post prediction refinement to arrive at the final pose. Moreover, methods in each of these categories require extensive annotation of ground truth 6-Dof poses in the training data. While large scale annotation for bounding boxes and instance segmentation masks can be automated or crowd-sourced through online tools such as Amazon Mechanical Turk [38], preparation of 6-Dof pose labels requires specialized offline tools such as Labelfusion [39]. Dataset sizes scale poorly with number of objects since networks need to be trained on

multiple objects from as many viewpoints as possible per object with varying degrees of inter-object occlusions.

Recent research works such as [40, 41] have proposed to counter the heavy pose annotation requirement by training autoencoders and then using the encoder embedding space to compute similarity scores between observed image and rendered images constructed by uniformly sampling the rotation space. However these methods still require training of a pose estimation neural network in addition to training for standard computer vision tasks like instance segmentation and object detection.

2.2 Deliberative Methods

Analysis-by-synthesis or deliberative approaches [1, 2, 3, 26, 27] rely on rendering and verification. They aim to find the best possible explanation for the observed scene by rendering multiple scenes using available 3D models and then finding the best match. Past work on Perception via Search (PERCH) [1, 2, 3], has demonstrated the capabilities of combining rendering with search for multi-object 3-Dof pose (x, y, yaw) estimation under occlusion and clutter. However PERCH ignores RGB information present in the observed scene as well as in available 3D models. As a result of this, the method fails under some commonly occurring scenarios in homes and retail stores, for example when objects of different brands have the same shape (such as soda cans, cereals etc.). In addition, the current formulation suffers from low scalability owing to its high runtime, restricting its application to only 3-Dof pose estimation of static objects.

The work in [2], proposed an extension to PERCH [1] by using the output of a 2D R-CNN object detector as a heuristic to speed up the tree search. However the underlying parallelization approach in both works limits the achievable speed. Moreover, as we will show in later sections, progress in accuracy of 2D object detection in the recent years presents us with an opportunity to use the outputs of learning based 2D object detectors for developing an integrated deliberative-discriminative framework for 6-Dof pose estimation.

2.3 Optimization Methods

ICP [42] is a popular optimization approach that is used in pose estimation and related tasks such as scan matching and tracking camera pose of a robot in motion. Several enhanced versions of the original ICP algorithm have been proposed over the years such as the probabilistic GICP or Generalized ICP [43] which is capable of modelling different surface types through gaussians. Despite increase in accuracy, generalizability and speed of newer ICP variants [44], ICP alone is insufficient for object pose estimation in clutter since it requires initialization with a candidate pose close to the actual object pose. However this makes ICP and its variants, perfect candidates for refinement of poses, predicted through other methods [6, 45]. It must be noted that during this process, special care must be taken to avoid local minima that ICP suffers from. In [45], a multi-object pose hypothesis is generated for the input scene and the best poses after ICP refinement are

selected based on the proposed alignment metric. However, the proposed method uses a Kalman filter to track the poses after initial estimation. In [6], in order to avoid local minima, the poses predicted by the network are first perturbed, creating a pose hypothesis, from which the best pose is selected using the alignment metric of [45]. However local perturbation can offer little enhancement if the initial predicted pose by the CNN is significantly different from the actual pose. ICP is also an important component of PERCH [1] that contributes directly to its accuracy by helping it resolve artifacts that arise due to discretization of the poses used to generate the space of rendered scenes for evaluation. However as we will demonstrate in this work, the ICP approach utilized in PERCH doesn't scale well with number of objects and number of poses being considered.

Chapter 3

Background

The problem setup and optimization formulation for estimating the 3-DoF pose (x, y, yaw) by PERCH [1] are stated in this section. We also discuss extensions to PERCH which are relevant to our work.

3.1 Perception via Search for Multi-Object Pose Estimation

We use a 3-Dof pose estimation formulation similar to PERCH in our work. PERCH assumes a set of K objects in the input point cloud, given the 3D models of N unique objects. It allows the possibility of $K \geq N$, in cases where multiple copies of a particular instance are present in the scene. The 6-DoF camera pose is further assumed to be given. The notations used by PERCH are listed in Table I.

3.1.1 Problem Formulation

Given the input point cloud I, PERCH [1] estimates poses of $O_{1:K}$ objects in the scene, by seeking to find a rendered point cloud R_j having $j(\leq K)$ objects, such that every point in I has an associated point in R_j and vice-versa. In other words, PERCH seeks to minimize the following objective :

$$J(O_{1:k}) = \underbrace{\sum_{p \in I} \text{OUTLIER}(p|R_k)}_{J_{observed}(O_{1:K}) \text{ or } J_o} + \underbrace{\sum_{p \in R_k} \text{OUTLIER}(p|I)}_{J_{rendered}(O_{1:K}) \text{ or } J_r} \tag{3.1}$$

in which OUTLIER($p|C$) for a point cloud C and point p is defined as follows:

$$\text{OUTLIER}(p|C) = \begin{cases} 1 & \text{if } \min_{p' \in C} ||p' - p|| > \delta \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where δ is the sensor noise resolution. In order to counter the intractability of this joint global optimization problem owing to a large search comprising of all possible joint poses of all objects, PERCH decomposes the cost function over individual objects added to the rendered scene. The

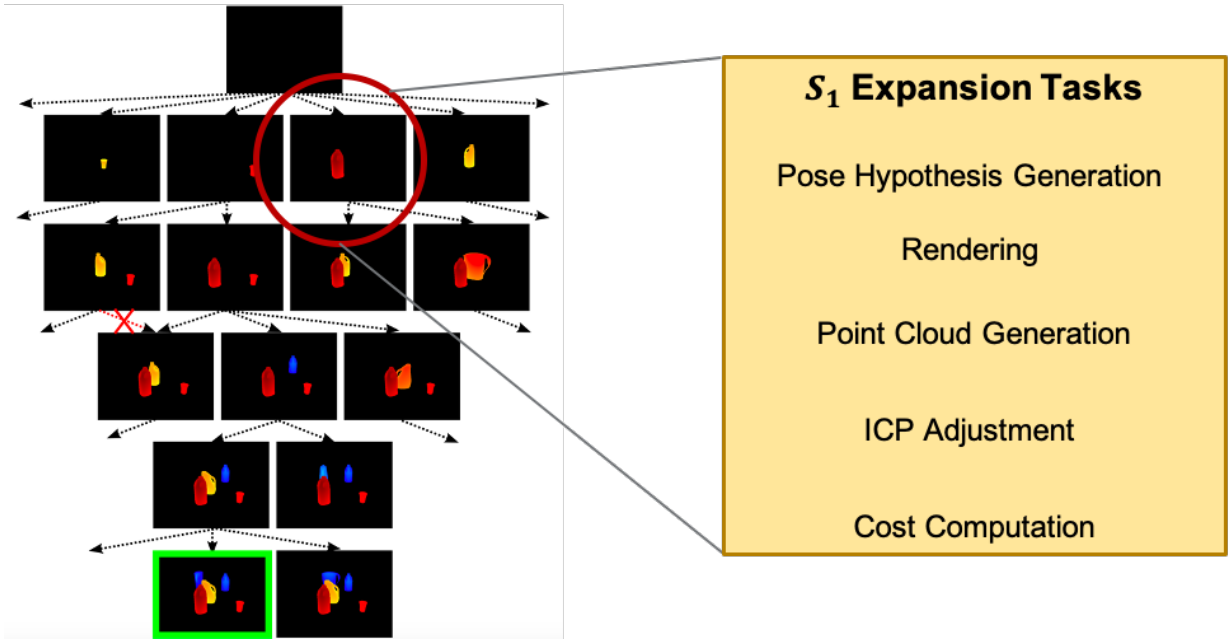


Fig. 3.1: Portion of the Monotone Scene Generation tree constructed by PERCH[1] and corresponding tasks involved in expanding a state S_1

.

decomposition is subject to the constraint that the newly added object does not occlude those already present. This allowed the optimization to be formulated as a tree search problem where a successor state is added to the tree whenever a new object is added to the rendered scene (Figure 3.1). The cost of edge between two such states is given by :

$$c(s_{j-1}, s_j) = \Delta J_r^j + \Delta J_o^j \tag{3.3}$$

where,

$$\Delta J_r^j = \sum_{p \in \Delta R_j} \text{OUTLIER}(p|I)$$

$$\Delta J_r^j = \sum_{p \in \{I \cap V(O_j)\}} \text{OUTLIER}(p|\Delta R_j) + \text{RESIDUAL}(j)$$

$$\text{RESIDUAL}(j) = \begin{cases} \sum_{p \in \{I - V_k\}} \text{OUTLIER}(p|R_k) & \text{if } j = K \\ 0 & \text{otherwise} \end{cases}$$

It is clear that the expansion of each state in the PERCH search tree has a significant computational cost that scales unfavourably with the number of successors to be generated for the state. Figure 3.2 illustrates the steps followed during expansion of a state S_1 in the tree. As shown, the successors are generated by first rendering the object O_j to be added to the state in different poses using OpenGL. For each pose, the algorithm then composes the rendered image with an image containing objects already present in the parent state. This step is essential to check if the

TABLE I: Notations used in PERCH [1]

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_j	An object state specifying a unique ID and 3-DoF pose
R_j	Point cloud for a rendered scene with j objects $O_{1:j}$
ΔR_j	Point cloud with points of R_j belonging exclusively to O_j
$\Delta \tilde{R}_j$	ΔR_j after ICP refinement
$V(O_j)$	The set of points in an admissible (conservative) volume occupied by object O_j, (volume of the inscribed cylinder)
V_j	The union of admissible volumes occupied by objects $O_{1:k}$
J_o	The observed cost of the scene with respect to given R_j
J_r	The rendered cost of the scene with respect to given R_j

current object occludes any object already present or to remove pixels corresponding to occlusions caused by other objects in the scene. This is followed by conversion of the rendered depth image to a point cloud and downsampling it with VoxelGrid downsampling, thus obtaining ΔR_j. In order to account for discretization artifacts, local-ICP [42] is used to refine the pose. Since the adjusted state may change its occlusion properties, it is rendered again, composed with the parent image and finally converted to the downsampled adjusted point cloud $\Delta \tilde{R}_j$. k-d tree [46] based nearest neighbour searches are then performed to calculate the observed and rendered cost for each of the successor states. For computing rendered cost J_r, the k-d tree representation of the observed depth input is used and the distance between every point in ΔR_j and its nearest neighbour in the k-d tree is computed iteratively to classify it as an outlier or inlier according to Equation 3.2. For observed cost J_o, a similar process is followed, though the k-d tree representation of every $\Delta \tilde{R}_j$ needs to be constructed. While PERCH uses OpenMPI to exploit the parallelism by executing these sequential steps in parallel threads for each successor state being added to the tree, the restricted number of CPU cores available in regular PCs places a practical limit on the speedup obtained through this approach. Moreover, the approach fails to take advantage of a much wider parallelism in each independent step.

3.2 Extensions to PERCH

In this section we highlight some extensions to PERCH that were proposed in recent research and form the inspiration for this work.

3.2.1 Discriminatively-guided Deliberative Perception

It was shown in the work on D2P (Discriminatively-guided Deliberative Perception [2]), that the search can be accelerated by using an admissible estimate of the true edge cost $c(s_{j-1}, s_j)$ that is obtained without rendering the full successor state. This admissible estimate or *lazy* edge cost is given by :

$$\tilde{c}(s_{j-1}, s_j) = \sum_{p \in \Delta \tilde{R}_j} \text{OUTLIER}(p|I) \tag{3.4}$$

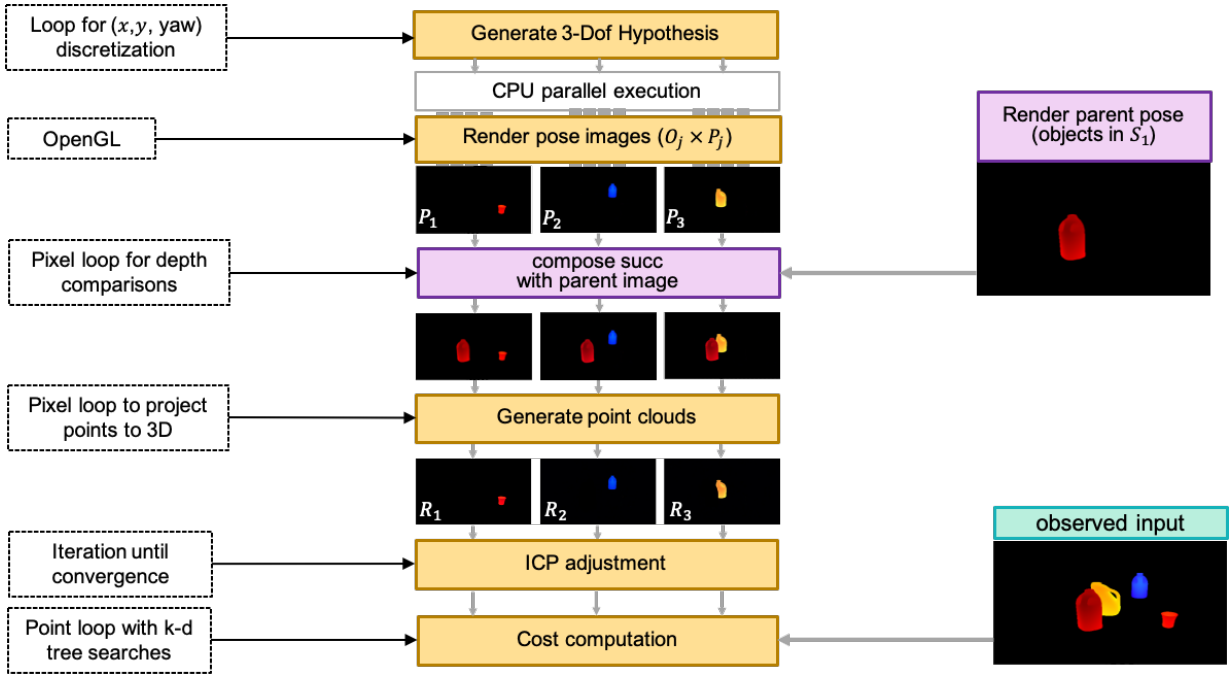


Fig. 3.2: Expansion of a state S_1 in the PERCH flow on CPU [1]

where $\Delta\tilde{R}_j$ is obtained by subjecting the point cloud of the newly added object ΔR_j to ICP refinement and subsequent removal of occluded points.

The work also proposed the use of discriminatively-trained algorithms to guide the search which is based on deliberative reasoning with the objective of reducing the time needed to find a solution. As shown in Figure 3.3, the input point cloud is clustered into K components and the points in each cluster are back-projected to obtain ROIs in the depth image. Then, the ROIs are fed to an R-CNN object detector trained on the complete object instance database, after appropriate scaling and colorization. Finally, every high-confidence class prediction for an ROI is converted to a heuristic for global search. Let l denote the label associated with a unique object model, B_i the set of ROIs (bounding boxes) in the depth image and $c(l|B_i)$ the confidence score for object instance l being present in B_i. For every detection with $c(l|B_i) \geq c_{thresh}$, the heuristic is generated as follows:

$$\bar{p} = \text{PROJECTTOSUPPORTPLANE}\left(\text{CENTROID}\ \left(\{p|p \in B_i\}\right)\right) \tag{3.5}$$

$$h\left(s_j\right) = \begin{cases} \infty & \text{if } \mathrm{id}\left(O_j\right) \neq l \\ 0 & \text{if } \left\|\bar{p} - T\left(O_j\right)\right\|_p \leq r_{\text{detector}} \\ \left\|\bar{p} - T\left(O_j\right)\right\|_p & \text{otherwise} \end{cases} \tag{3.6}$$

where $||\cdot||_p$ is the p-norm and $T(O_j)$ is object O_j's center (assuming all models have been pre-processed such that the z-coordinate of their origins have been set to the height of the supporting surface), ignoring the orientation).

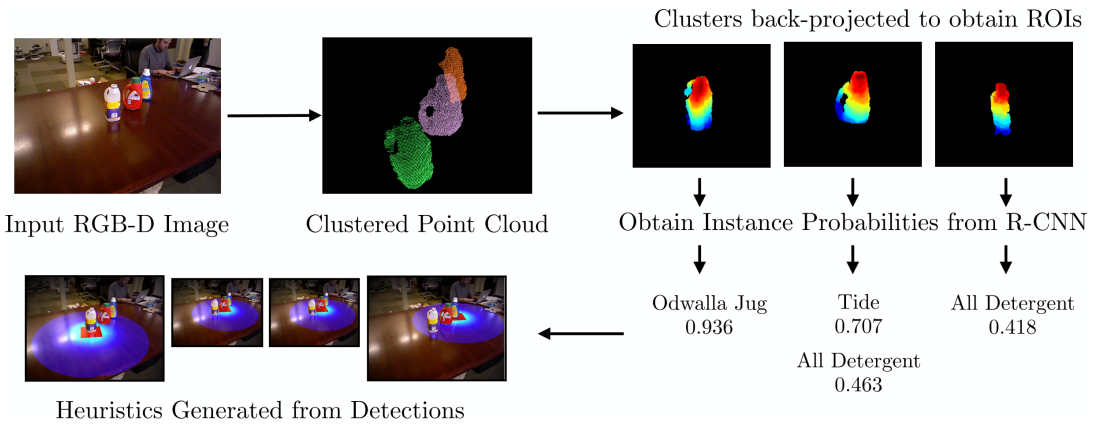


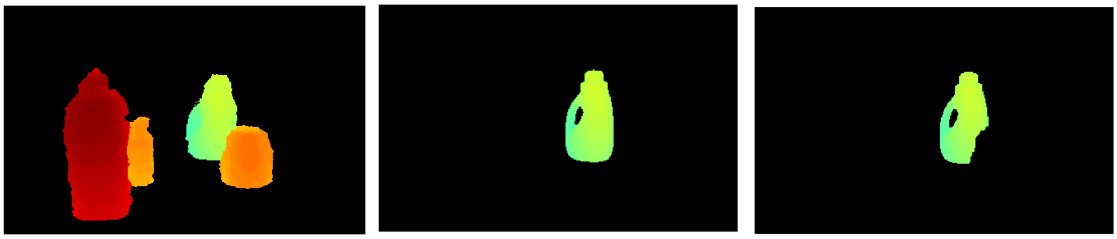
Fig. 3.3: D2P [2] pipeline. In this example, the R-CNN predicts two possible hypotheses for the center ROI, which results in two heuristics being created for that ROI.

3.2.2 Deliberative Perception in Clutter

As is evident from previous sections, the Monotone Scene Generation tree formulation is primarily used to account for inter-object occlusions. However, the work on C-Perch [3] proposed an alternate way to acknowledge inter-object occlusions by marking certain points C in the input scene as clutter and use them as extraneous "occluders" while rendering the object of interest in the scene (Figure 3.4), obtaining the rendered point cloud $R_k|\Delta C_k$ instead of R_k. Here ΔC_k represents the points in the input scene that occlude object O_k. It was shown that this strategy for modelling inter-object occlusions is incredibly useful when models of all objects in the scene are not available and thus the Monotone Scene Generation tree can't be used to account for the same.
From [3], we note the changes to the terms J_o and J_r in Equation 3.3 :

$$\begin{aligned} J_o\left(O_k, C_o\right) &= \textstyle\sum_{p \in I \cap V(O_k)} \mathrm{OUTLIER}\left(\boldsymbol{p} \,|(R_k|\Delta C_k)\right) \\ J_r\left(O_k, C_o\right) &= \textstyle\sum_{\boldsymbol{p} \in R_k|\Delta C_k} \mathrm{OUTLIER}(\boldsymbol{p}|\boldsymbol{I}) \end{aligned} \tag{3.7}$$

Here ΔC_k represents the extraneous "occluders" or the set of clutter points that occlude the object O_k created by rendering the object pose O_k and $R_k|\Delta C_k$ is the corresponding point cloud of the object.



(a) (b) (c)

Fig. 3.4: (a) The input point cloud I (represented as a depth image and pseudo-colored) (b) Rendering R_1 corresponding to one object object O_1 (c) Rendering $\boldsymbol{R}_1|\boldsymbol{\Delta C}_1$, considering points in I that occlude R_1 [3]

Chapter 4

PERCH 2.0

In this section, we describe our proposed methodology or PERCH 2.0 and how we overcome shortcomings of PERCH as well as some of the related work. In Section 4.1 we describe the incorporation of RGB data into the PERCH 2.0 cost function which allows it to distinguish between objects of the same shape. In Sections 4.2, 4.3, 4.4 and 4.5, we describe how GPU parallelization of various components is integrated into PERCH 2.0 for runtime improvement. Lastly, in Section 4.6 we describe the proposed combined discriminative-deliberative framework for accurate 6-Dof pose estimation.

4.1 Augmented Objective Function with RGB

The formulation of explanation cost in PERCH is based on the implicit assumption that depth data alone can be used to capture how well a rendered point cloud matches the observed point cloud. More formally, the classification of a point p in a point cloud C as an outlier in Equation 3.2 is entirely based on the euclidean distance between them in 3D space. However this definition fails in scenarios similar to those depicted in Figure 4.1. In such scenarios, where objects of similar shape are present, PERCH is unable to estimate the (x, y, yaw) correctly because rendering any object at a given (x, y, yaw) results in the same change in cost, owing to an outlier definition based purely on euclidean distance.

Intuitively, the explanation cost in such cases must utilise the difference in point-wise RGB information present in the objects' 3D models and in the observed point cloud. Consequently, we first enhance our rendering process to render the point cloud ΔR_{cj} with per-pixel depth and RGB information whenever a new object O_j is added to the scene. Now any comparison between a point p in I and a point p' in the ICP adjusted $\Delta \tilde{R}_{cj}$ or vice-versa needs to utilise the per-pixel RGB information present in both point clouds and also accommodate changes in perceived color due to lighting. Keeping these requirements in mind, we introduce the CIEDE2000 color difference formula [47] in the CIELAB color space to perform the comparison between a point in the observed cloud I and the rendered point cloud ΔR_{cj} or vice-versa. In this space, each color is represented by 3 values - $L*$, $a*$ and $b*$ and uniform changes in these components are designed to replicate uniform changes in color as perceived by the human eye. Formally, for a

Fig. 4.1: Objects & some sample images from the dataset used for evaluating 3-Dof PERCH 2.0

point p in C, the OUTLIER$(p|C)$ definition in Equation 3.2 can be re-written as :

$$\text{OUTLIER\_RGBD}(p|C) = \begin{cases} 1 & \text{if } \min_{p' \in C} ||p' - p|| > \delta \\ 1 & \text{if } \min ||p''_c - p_c||_c > \tau_c \forall\, p'' \\ & \text{s.t. } \min_{p'' \in C} ||p'' - p|| \leq \delta \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where :

- p_c and p''_c denote the color in CIELAB of points p and p'' respectively
- $||p''_c - p_c||_c$ denotes the CIEDE2000[47] color-difference between the two points
- τ_c denotes the maximum allowed color difference for two colors to be considered same

With this definition of OUTLIER_RGBD$(p|C)$, we penalize points for being distant in color space even though they might satisfy the euclidean constraint in 3D space.

4.2 Parallel Scene Generation

Scene Rendering

At a high level, the process of rendering a given number of objects N_i for state S_i consisting of P_j poses of each object O_j can be thought of as having $N_i \times P_j$ parallel threads. However if we consider each object and its corresponding 3D mesh model to be made up of T_j triangles, a parallelism over $N_i \times P_j \times T_j$ threads can be observed. Consider a simple scenario consisting of 4 objects having 10 poses each and 10,000 triangles in each mesh model. The corresponding rendering task exhibits a parallelism of $400,000$ threads. The scale of this parallelism is ideal for exploitation on a GPU and consequently we use that approach in PERCH 2.0. In addition to speed, having a fine grained control on rendering allows us to combine a 2 step process of rendering a successor image and composing it with its parent, into a single step process that

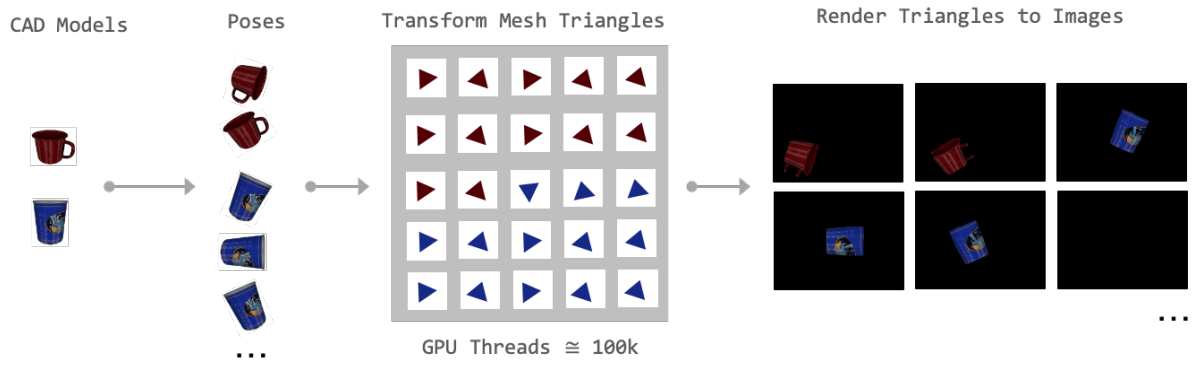


Fig. 4.2: A graphical depiction of the GPU based rendering flow

occurs while a given mesh triangle is being rendered into the scene. The process is illustrated in Figure 4.2.

Scene Point Cloud Generation

Once the rendered RGB and depth images have been obtained for all objects and poses, they need to be converted to point clouds with every pixel transformed to its corresponding 3D point using the depth input and camera intrinsic parameters in parallel. Given a depth input D, we the following relation can be used to get the (X, Y, Z) location of every pixel (x, y) :

$$Z = D(x, y) \tag{4.2}$$

$$X = \frac{(x - c_U) \times Z}{f_U} \tag{4.3}$$

$$Y = \frac{(y - c_V) \times Z}{f_V} \tag{4.4}$$

Here (f_U, f_V) are the camera focal lengths and (c_U, c_V) are the camera centers. This process can be seen to have a parallelism of $N_i \times P_j \times L_j$ threads where L_j is the number of points in the rendered image of the pose P_j.

An observation of the rendered images in Figure 4.2 shows that the Region of Interest or ROI corresponding to the rendered object occupies a relatively small area with respect to the whole image. Even within the ROI, the pixel density is quite high and a direct transformation of every pixel in this region to corresponding (X, Y, Z), would result in a very dense cloud which would in turn limit the scalibility of the downstream tasks. In order to counter this, we introduce a downsampling and masking step that computes a mapping from the pixel space of the rendered images to the reduced 3D point cloud space while ignoring all pixels outside the ROI. This mapping is then used to compute a downsampled point cloud for all rendered images in parallel on the GPU. The process is illustrated in Figure 4.3.

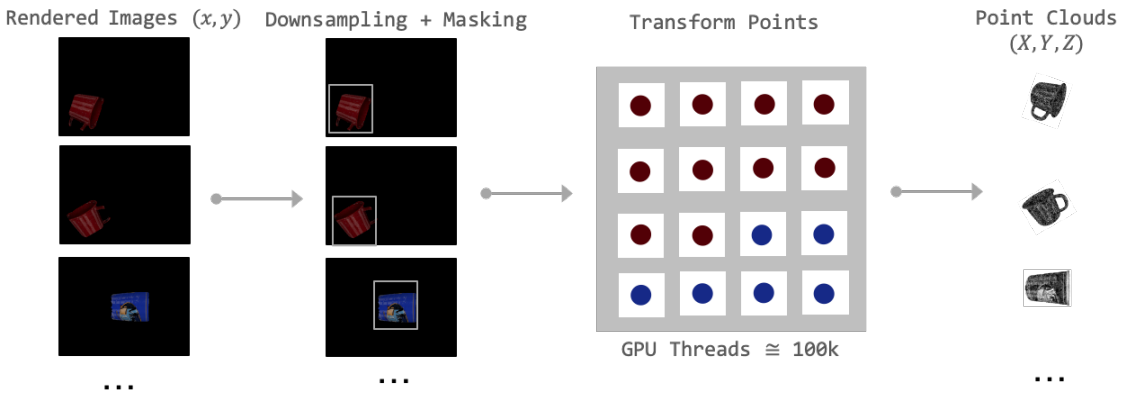


Fig. 4.3: A graphical depiction of the GPU based point cloud generation flow

4.3 Parallel Many to Many GICP

ICP techniques available in literature are designed to align a given source pose to a given target pose. A similar approach of point-to-point non-linear ICP from the PCL library is used by PERCH [1]. However as we will show in later sections, this approach is insufficient to deal with the scalability requirements presented by common pose estimation scenarios. Moreover, as we will demonstrate, a point-to-point ICP approach like the one used by PERCH, leads to low accuracy under high occlusion by converging to the wrong pose. Recent works on GICP [43, 44] have proposed to counter this problem by developing a generalized version of ICP or GICP. GICP combines features of point-to-point and point-to-plane ICP by modelling the surface from which each point is sampled as a Gaussian distribution.

Following [44], we explain GICP in this section. Let a_i and b_i be two points in source pose and target pose found by a nearest neighbor search and the transform between them is given by T. We assume they are both drawn from Gaussian distributions $a_i \sim \mathcal{N}\left(\hat{a}_i, C_i^A\right)$ and $b_i \sim \mathcal{N}\left(\hat{b}_i, C_i^B\right)$ and the transform error between them is defined as follows :

$$\hat{d}_i = \hat{b}_i - \mathbf{T}\hat{a}_i \tag{4.5}$$

The error $\hat{d}_i$ can be represented by :

$$d_i \sim \mathcal{N}\left(\hat{b}_i - \mathbf{T}\hat{a}_i, C_i^B + \mathbf{T}C_i^A\mathbf{T}^T\right) \tag{4.6}$$

$$= \mathcal{N}\left(0, C_i^B + \mathbf{T}C_i^A\mathbf{T}^T\right) \tag{4.7}$$

The GICP objective function is to find the transform T such that :

$$\mathbf{T} = \arg\max_{\mathbf{T}} \sum_i \log\left(p\left(d_i\right)\right) \tag{4.8}$$

$$= \arg\min_{\mathbf{T}} \sum_i d_i^T \left(C_i^B + \mathbf{T}C_i^A\mathbf{T}^T\right)^{-1} d_i \tag{4.9}$$

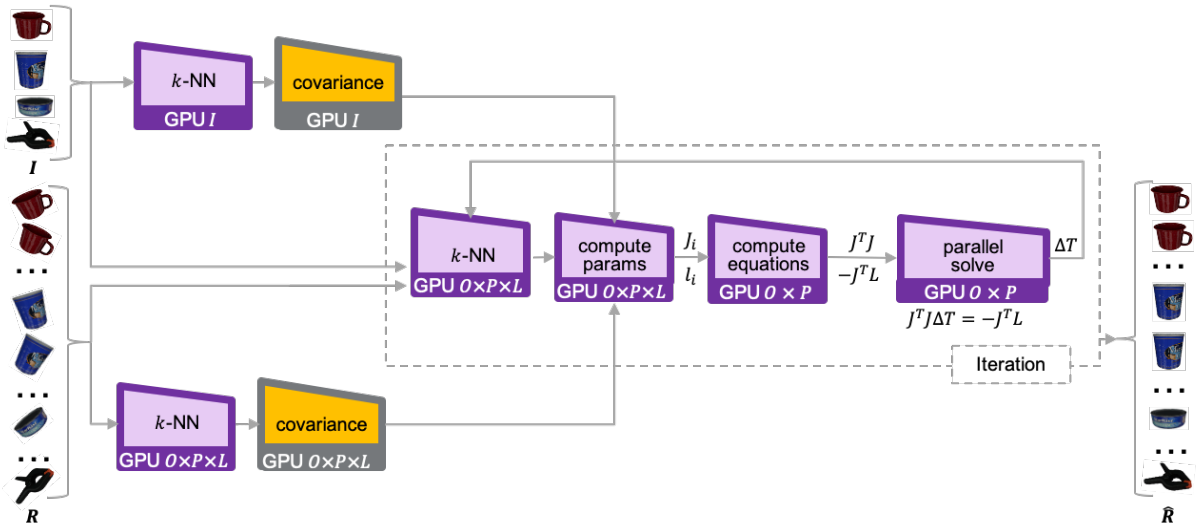


Fig. 4.4: Parallel Many to Many GICP flow (O : Number of object instances, P : Number of poses per object instance, L : Number of points in rendered point cloud of a given pose, I : Input point cloud, R : Rendered point clouds, $\hat{R}$: GICP adjusted rendered point clouds)

The covariance matrices C_i^A and C_i^B are computed from k nearest neighbors in the respective point clouds. In practice Equation 4.8 is solved by using a Gauss Newton optimizer as in [44]. We can see from the outset that performing all the required steps for each candidate pose with the observed scene would be inefficient even if performed in parallel in CPU threads. We propose a modified scalable GPU GICP approach that is shown in Figure 4.4. The kNN computation required in the flow is described later in Section 4.4. In order to compute J^TJ and $-J^TL$ from J_i and l_i, cuBLAS [48] batch matrix multiplication is used where each batch represents one pose. The resulting set of equations are solved in parallel using the cuDNN library [49] to obtain ΔT for every pose. This process is repeated iteratively until all poses have converged.

4.4 Parallel Objective Function Evaluation

In this section we describe two approaches that we incorporate in PERCH 2.0 to speed up the evaluation of the objective function. The primary difference in the two approaches is the method used to compute the nearest neighbour distances or kNN which are then used to compute OUTLIERS in Equation 3.2. In the Experiments section, we present the effects of both approaches on overall runtime.

kNN Approach I

The need to create k-d trees for each successor cloud ΔR_j and then iteratively computing nearest neighbours in I for every point in every ΔR_j leads to slow speeds despite the efficiency of the k-d tree data structure. This parallelism for computing nearest neighbour distances over N_i objects, P_j poses of each object O_j, consisting of L_j points each in the ICP adjusted point cloud can be

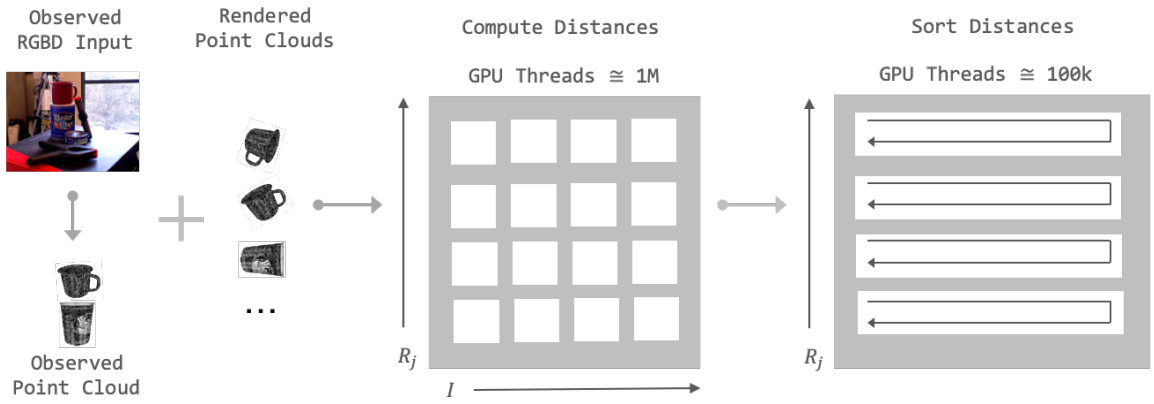


Fig. 4.5: kNN approach I, R_j : Set of points in all rendered poses, I : Set of points in the observed point cloud

considered as requiring $N_i \times O_j \times L_j \times I$ parallel threads. We use the kNN-GPU library proposed in [50] to compute the nearest neighbour in the input point cloud I for each point in every ΔR_j in order to exploit this parallelism.

kNN Approach II

While Approach I is fully parallel, a closer look at the implementation reveals the need to allocate a large 2D array to store pairwise distances between points in ΔR_j and I as they are being computed in parallel. This imposes a larger memory requirement on the GPU which could drive up the peak memory usage and limit the overall number of poses that can be evaluated in parallel. Thus we develop another approach that exploits a reduced parallelism of $N_i \times O_j \times L_j$ threads. In each thread, we loop over the points in I, computing distances to points in ΔR_j and pushing them into a priority queue. When all threads have finished execution we have the nearest neighbours and corresponding distances for every point ΔR_j to every point in I. Unlike kNN Approach I, the reduced parallelism in this approach eliminates the need for the allocation of a large 2D array.

Once distances are computed by using either approaches, another GPU kernel is then used to classify every point as inlier or outlier in parallel, thus obtaining the rendered cost J_r. Finally we use an additional kernel to compute the observed cost J_o, which checks every point in the input scene I and if it lies within the volume occupied by an given object pose $V(O_j)$, simultaneously marking it as inlier or outlier depending on whether it was found as a nearest neighbour for a point in the corresponding ΔR_j in the previous step.

4.5 Parallel Search

Despite speedup from enhancements in the above steps, the runtime remains limited owing to the sequential nature of the Monotone Scene Generation tree. More specifically, the search has to figure out the right non-occluding order in which to place the objects until a solution that satisfies

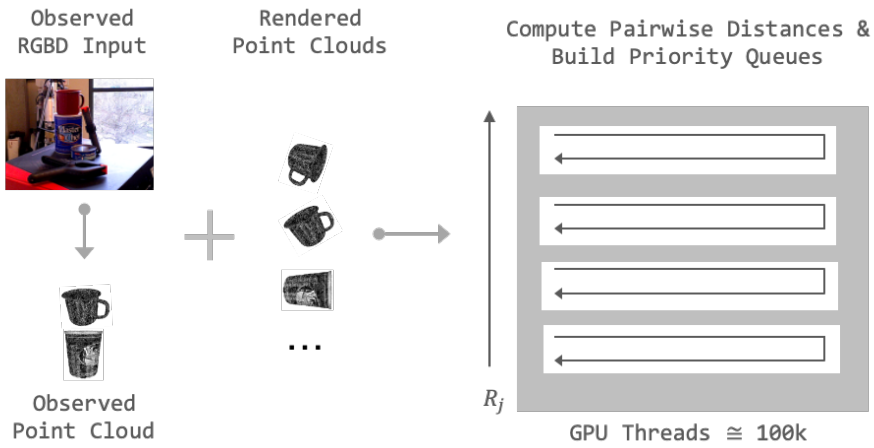


Fig. 4.6: kNN approach II, R_j : Set of points in all rendered poses

the cost bound has been found. We recount from [1] that this process is primarily a way to model inter-object occlusions. We build on the strategy used in C-Perch [3] in PERCH 2.0, by treating the search for each object as an independent search for that object in a cluttered scene where the model for other objects is unknown. This change effectively reduces a sequential search to a parallel one that can be performed efficiently with our GPU based pipeline. Following a strategy similar to [3] for creating $R_k|\Delta C_k$ by using the input depth image, we render and compute costs for all successors and fine the best one corresponding to the minimum cost for each object in parallel on the GPU. The complete 3-Dof pose estimation pipeline is shown in Figure 4.7.

4.6 6-Dof Pose Estimation

6-Dof pose estimation has been the primary focus of pose estimation research in recent years. A large majority of this research has focused on the task of estimating the 6-Dof pose (location and orientation) as well as class labels of every object in the scene. Intuitively, a simple extension of deliberative pose estimation methods like PERCH to this task would be impractical, owing to the exponential increase in the number of poses that results from considering the full 6-Dof pose of the object as opposed to only the 3-Dof pose. Despite speedup from the GPU based formulation, a direct application of PERCH 2.0 to this task would still be insufficient. However, due to the success of convolutional neural networks (CNNs) in 2D instance segmentation, we have the opportunity to extend deliberative pose estimation to 6-Dof by pruning a large part of the search space using the output of a 2D object detector. Under this combined framework, we generate a set of pose proposals $H(O_j)$ by combining the 2D object detector with a uniform sampling of the rotation space and then process these proposals using PERCH 2.0 to estimate 6-Dof poses for every object in the scene. Besides extending deliberative pose estimation to 6-Dof, the proposed combined framework also avoids several disadvantages of purely discriminative data-driven techniques highlighted in Section 2.1. The integration also enables the relaxation of key assumptions made in the PERCH problem formulation such as the pre-requisite knowledge labels of objects in scene and the 6-Dof camera pose.

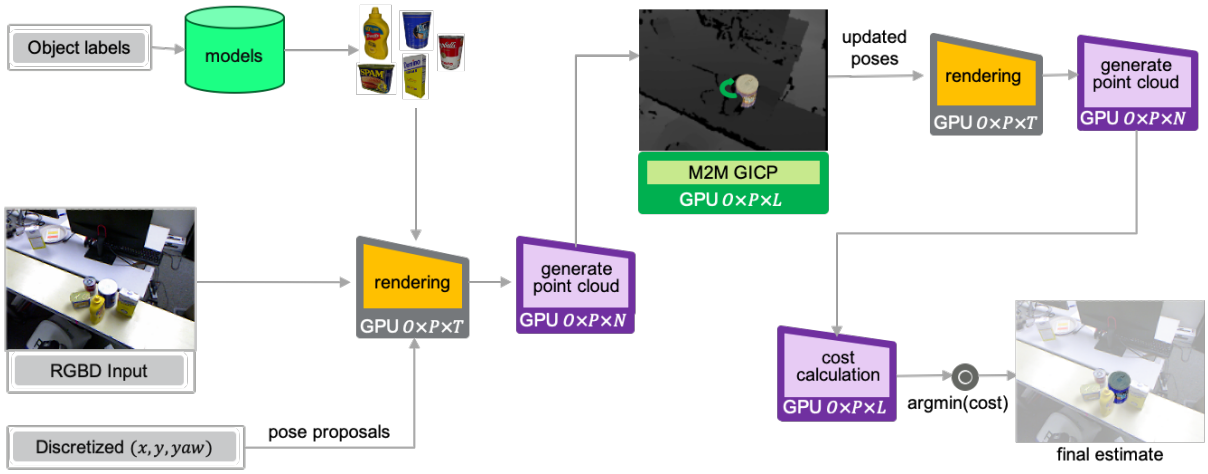


Fig. 4.7: 3-Dof pose estimation flow (O : Number of object instances, P : Number of poses per object instance, T : Number of triangles in an object 3D mesh model, N : Number of pixels in rendered pose image, L : Number of points in given pose point cloud)

4.6.1 2D Object Detection and Instance Segmentation

Instance segmentation and its application towards 6-Dof pose estimation have been studied in works such as [6, 37]. However a majority of such works have focused on using large datasets annotated with 6-Dof poses of objects to train CNNs for 6-Dof pose estimation. In this work we show that when combined with a deliberative framework, the information contained in a 2D bounding box and an instance segmentation mask is sufficient to obtain 6-Dof pose, thus eliminating the need to train additional networks or create 6-Dof pose annotated datasets.

Several 2D object detection methods in recent years have been shown to be capable of recognizing and segmenting objects in the image space even under high occlusion [51, 52, 53, 54, 55]. While 6-Dof pose estimation research has successfully recognized the capability of such methods in overcoming occlusion, bounding boxes and their role in countering occlusion has been largely left unexplored. On the contrary bounding boxes have received notable attention in detection of pedestrians and vehicles in outdoor scenes [56]. Recent outdoor datasets such as [4] have focused on annotation of "full" bounding boxes as opposed to only "visible" bounding boxes in the image database. The difference between the two kinds of annotations is highlighted in Figure 4.8. In our 6-Dof pose estimation framework, we recognize and utilize this insight of employing "full" bounding box annotation to assist in pose estimation of occluded objects by giving us a more accurate location of the 2D projection of the object's 3D bounding box in the image. With the help of full bounding box and instance segmentation mask, we can generate a set of translation proposals for a detected object O_j as follows:

$$H_{tj} = \langle x_c, y_c, z_i \rangle \text{ where } z_{min} \leq z_i \leq z_{max} \tag{4.10}$$

In the above equation, $\langle x_c, y_c \rangle$ is obtained by back projecting the center of object's 2D full bounding box into 3D space using the camera's projection matrix. z_i ranges from z_{min} , the closest point to the camera corresponding to the given object in the observed depth image to, z_{max} , the farthest

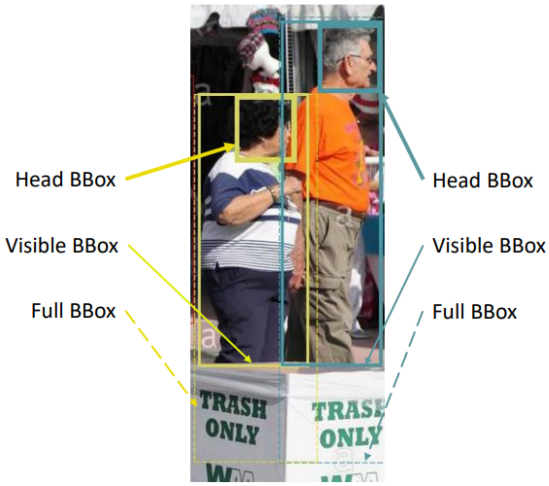


Fig. 4.8: Different kinds of bounding box annotation [4]

point from the camera corresponding to the given object in the observed depth image. These are obtained by combining the segmentation mask for the object with the input depth image.

4.6.2 Uniformly Sampling the Rotation Space

In our approach, we uniformly sample the rotation space by representing all possible rotations as a set of viewpoints and in-plane rotation angles [25, 41]. We equidistantly-sample M viewpoints v from unit sphere and N in-plane rotation angles θ from $[0, 2\pi]$ and combine each with the other to generate $M \times N$ possible set of rotation proposals for object O_j:

$$H_{rj} = \langle v_i, \theta_k \rangle \text{ where } 1 \leq i \leq M \text{ and } 1 \leq k \leq N \tag{4.11}$$

An advantage of sampling rotations using viewpoints and in-plane rotations is that it allows for explicit handling and exploitation of symmetries in objects while ensuring that all object orientations have been considered. This is in contrast to CNN based pose estimation methods where learning object symmetries is left to the neural network. For an object that exhibits rotational symmetric about one axis such as the can shown in Figure 4.9, the sampling of in-plane rotation can be skipped, significantly reducing the number of proposals to be considered. Similarly for an object such as a sugar box which is semi-symmetric or exhibits indistinguishable views if rotated along the axis by 180 degrees, only half of the viewpoint sphere needs to be sampled.

4.6.3 6-Dof Pose from Instance Segmentation

A pictorial representation of the entire 6-Dof pose estimation pipeline can be seen in Fig 4.10. The input RGB image is passed through a standard instance segmentation network, obtaining object labels, segmentation masks and 6-Dof pose proposals for the detected objects.

For each pose proposal the corresponding point cloud $R_j|\Delta C_j$ is generated through parallel

(a) (b) (c)

Fig. 4.9: (a) Objects from the YCB object database [5] with rotational symmetry about one axis (b) Objects from the YCB object database [5] with 180 degree semi-symmetry about 2 axis (c) Objects from the YCB object database [5] with no symmetry about any axis

rendering described in Section 4.2. ΔC_j corresponds to points marked as extraneous clutter, contributed by other objects in the scene O_k such that $L_k \neq L_j$ where L is the object label. The proposed GICP approach (Section 4.3) is used to refine the rendered poses. The rendered point clouds are treated as the source point clouds in ICP that need to be aligned to the segmented object point cloud created by combining the object's instance segmentation mask and depth image. We note that instance segmentation fits neatly into our parallel GICP framework, allowing us to align multiple source poses to multiple target poses in parallel. In addition, we use instance segmentation labels to speed up k-NN by computing nearest neighbours only for matching labels in rendered and observed point clouds.

Finally, the updated poses are rendered again and point clouds are generated. In the last step, we compute the cost of each pose proposal in parallel. For calculation of observed cost component J_o, instead of explicitly computing points within the object volume $V(O_j)$, the pixel-wise segmentation labels are used directly to determine the set of observed points belonging to a given object.

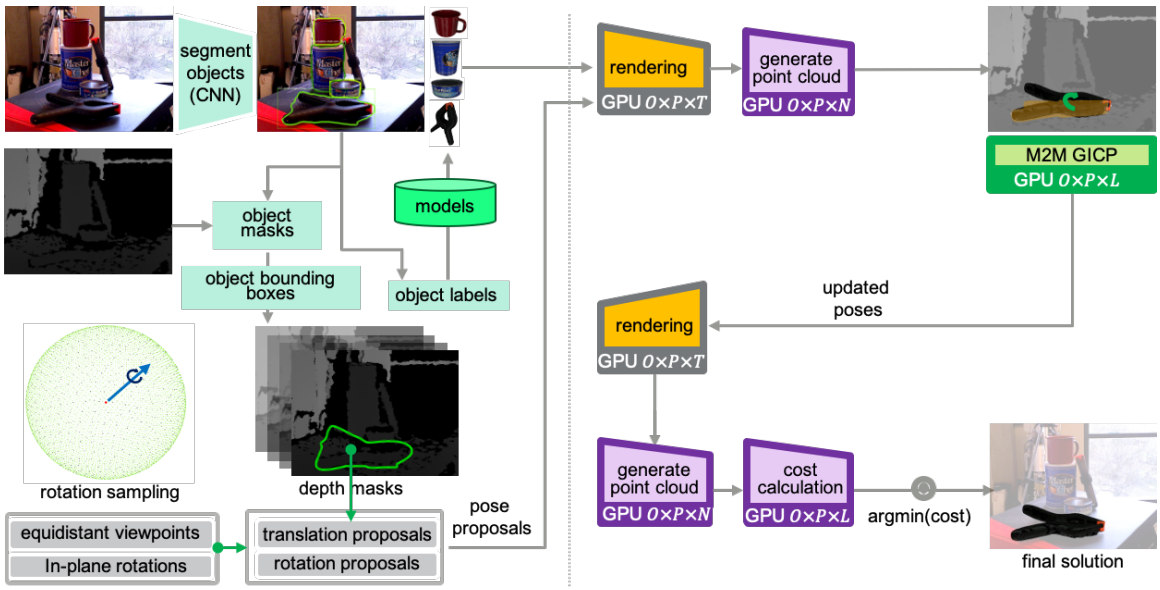


Fig. 4.10: 6-Dof Pose Estimation Pipeline (O : Number of object instances, P : Number of poses per object instance, T : Number of triangles in an object 3D mesh model, N : Number of pixels in a rendered pose image, L : Number of points in given pose point cloud)

Chapter 5

Experimental Results

5.1 3-Dof Pose Estimation

5.1.1 Tabletop Object Manipulation

Dataset

Early experimentation revealed that PERCH can exploit minute differences in shape and estimate poses accurately. Thus, for evaluating PERCH 2.0 against PERCH, we focus on images of common objects that have same shape but different appearance. Such objects are commonly found in grocery stores but to our knowledge, no database exists in literature that consists of depth and RGB images of such objects. Moreover for PERCH, we require variation only in 3-Dof pose (x, y, yaw) for every object while common annotated pose estimation databases consist of pose variation in 6-Dof. Subsequently, we constructed a synthetic photo-realistic dataset of 75 scenes with corresponding RGBD and depth images using the recently released NVidia NDDS [57] plugin for Unreal Engine 4 (objects shown in Figure 4.1). Within the plugin, we randomly vary 3D pose (x, y, yaw) of every object on a tabletop while keeping (z, roll and pitch) constant. The plugin allows generation of images with realistic lighting conditions and inter-object occlusion.

Baselines

For this experiment we use PERCH 2.0 with the same ICP as PERCH and kNN I (Section 4.4). In addition to a comparison with PERCH, we compare our method with the following baselines :

- DOPE [24] + ICP : DOPE is a leading RGB based 6-Dof pose estimation method directly compatible with NDDS generated data which we combine with ICP refinement on the depth input for our experiments.
- The Brute Force ICP (BF-ICP) baseline proposed in [1] is also used for comparison. Here the object's model is transformed by the same candidate poses considered by PERCH and PERCH 2.0 and then local ICP alignment is performed and the pose with the best ICP fitness score is selected.

TABLE I: Evaluation of 3-Dof pose estimation for similar shape objects

Objects	BF-ICP [1]		PERCH [1]		DOPE [24] + ICP		PERCH 2.0-A (W/O Occluder Marking)		PERCH 2.0-B (W/O Full Parallelization)		PERCH 2.0 (ICP + kNN I)	
	AUC	<1cm	AUC	<1cm	AUC	<1cm	AUC	<1cm	AUC	<1cm	AUC	<1cm
coke_bottle	46.61	0.00	55.43	58.00	90.00	94.00	96.59	100.0	96.6	100.00	**96.59**	**100.00**
sprite_bottle	46.16	0.00	55.37	58.00	87.99	84.44	97.06	100.0	96.65	100.00	**97.09**	**100.00**
sprite_can	17.62	0.00	43.04	30.00	90.71	80.00	57.41	60.00	95.42	100.00	**95.61**	**100.00**
pepsi_can	38.10	0.00	48.63	48.57	94.82	96.00	95.66	100.0	95.63	100.00	**95.69**	**100.00**
coke_can	46.61	0.00	40.58	40.00	89.18	89.18	93.39	97.30	95.61	100.00	**95.95**	**100.00**
7up_can	28.27	0.00	32.46	25.00	75.21	68.00	79.33	68.00	95.03	100.00	**95.26**	**100.00**
All Objects	37.51	0.00	47.16	43.26	88.16	85.27	80.49	87.55	95.26	100.00	**95.72**	**100.00**
Mean Runtime (s)	220.7		137.2		1.0		1.64		11.9		1.31	

- PERCH 2.0 (A) which doesn't use the input depth data to mark occluded points in the rendered scenes
- PERCH 2.0 (B) which doesn't use full parallelization like PERCH 2.0 but instead uses a tree search formulation similar to PERCH.

For training DOPE, we construct a training dataset of 12K images containing each of the selected 6 objects using NDDS[57]. The network was trained for 60 epochs (pretrained on ImageNet) on each object individually, taking approximately 12 hours for each on 2 NVidia P100 GPUs. For inference of DOPE and detection using PERCH and PERCH 2.0, an 8 core Intel i7-6700 CPU with an NVidia GeForce GTX 1070 8 GB GPU was used.

Metrics

We use the ADD-S [6, 33] metric for evaluation which computes the average distance between the closest points in the object's 3D model, transformed with ground truth pose and the same model transformed with the predicted pose. We vary the ADD-S distance threshold up to 0.1 m and obtain the area under the accuracy-threshold curve (AUC) for all methods as shown in Table I and Figure 5.1. We also compute ADD-S<1cm, which denotes the percentage of poses with less than 1cm ADD-S error.

Accuracy

PERCH 2.0 achieves the best performance among all variants with a 100% of poses below ADD-S 1cm error. It can also be noted that PERCH 2.0 and DOPE + ICP outperform PERCH and BF-ICP. This shows that PERCH 2.0 and DOPE are able to utilise the RGB information present in the object model and observed scene and closer inspection reveals that these methods don't get confused between similar looking objects even in occlusion (like sprite_can and pepsi_can).

Robustness

The robustness of the RGBD cost function used by PERCH 2.0 is highlighted by its ability to differentiate between objects of different sizes (bottle vs can), objects with minute color differ-

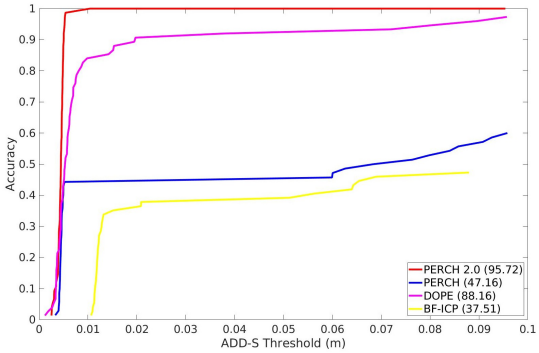


Fig. 5.1: ADD-S threshold curves for 3-Dof Pose Estimation

ences (pepsi can vs sprite can) and objects with a non-uniform color distribution (sprite can, 7up can). PERCH 2.0 also handles occlusions more effectively as compared to DOPE and PERCH 2.0 (A), which is exhibited in its better performance as compared to both.

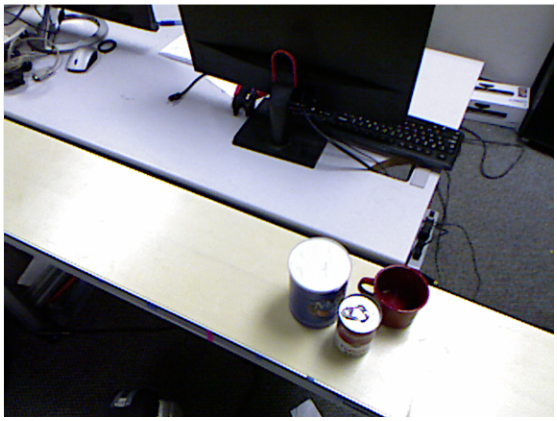
Runtime

From Table I it is clear that we are able to achieve an order of magnitude improvement in runtime ($\sim$100X) with PERCH 2.0 over PERCH. A comparison between PERCH 2.0 and PERCH 2.0 (B) also reveals that PERCH 2.0 is able to achieve the same accuracy with full parallelization that PERCH 2.0 (B) is able to obtain using the Monotone Scene Generation tree [1]. However PERCH 2.0 is 10 times faster than the latter. Moreover, PERCH 2.0 has a runtime close to the DOPE + ICP pipeline which suggests that it can achieve speeds comparable to popular learning based approaches followed by depth-based refinement without requiring any training for estimating 3-Dof poses and object categories.

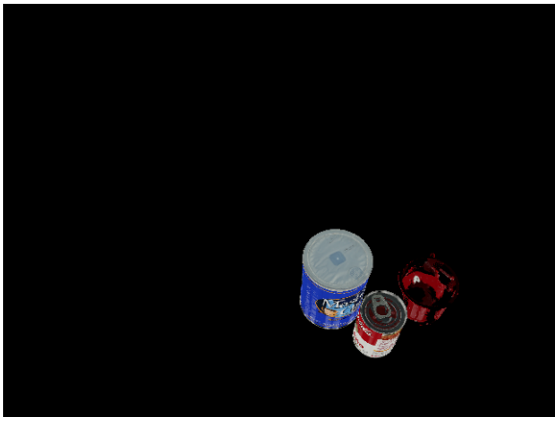
Qualitative Results

We analyzed the performance of PERCH 2.0 with several scenes captured on 2 robot systems - a PR2 mobile manipulator and a Ubtech Walker robot. The results are shown in Figure 5.2 and Figure 5.3. The results on PR2 in Figure 5.2 demonstrate that the pose estimation accuracy and runtime transfer well to the real world scenes that consist of multiple objects with inter-object occlusion. The results on the Walker robot in Figure 5.3 show that the RGB aware cost function is capable of distinguishing objects of similar shape but different appearance in the real world.

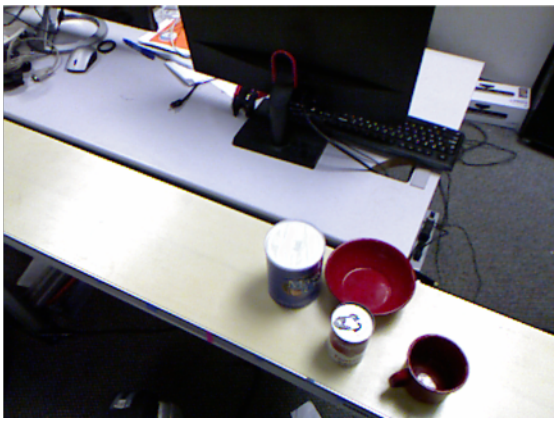
(a) PR2 Setup



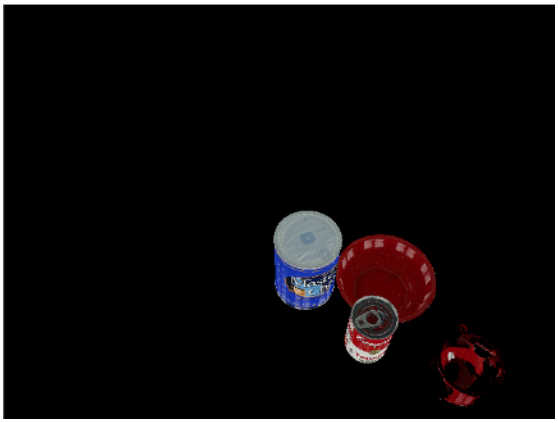
(b) Input Scene



(c) Reconstructed Scene (Runtime : 0.6s)

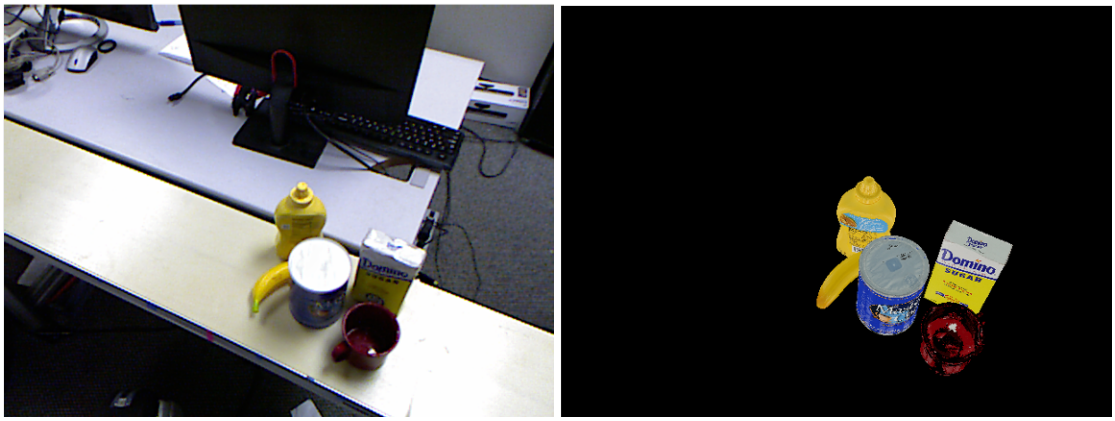


(d) Input Scene



(e) Reconstructed Scene (Runtime : 0.9s)

(f) Input Scene (g) Reconstructed Scene (Runtime : 3.4s)

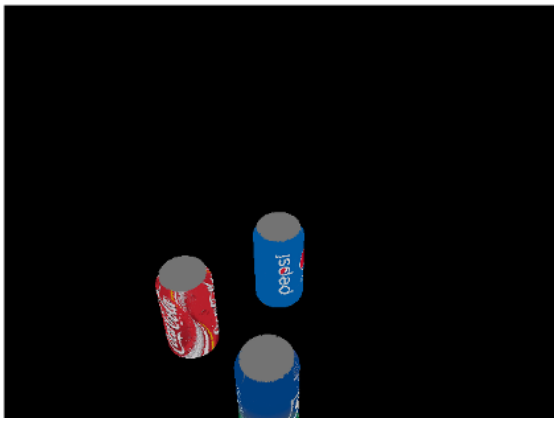


(h) Input Scene (i) Reconstructed Scene (Runtime : 4.9s)

Fig. 5.2: A few qualitative 3-Dof pose estimation results on PR2 for runtime evaluation

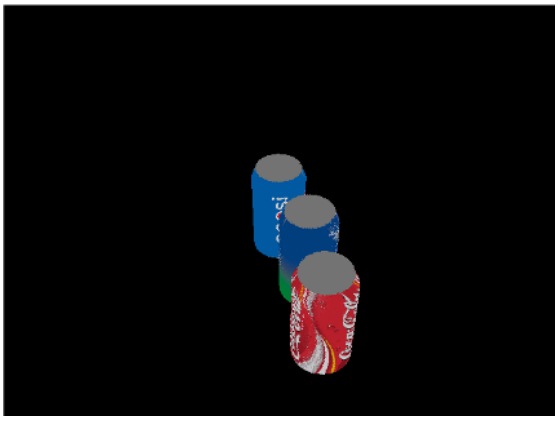
(a) Walker Robot Setup

(b) Input Scene



(c) Reconstructed Scene from Predicted Poses

(d) Input Scene



(e) Reconstructed Scene from Predicted Poses

(f) Input Scene (g) Reconstructed Scene from Predicted Poses

(h) Input Scene (i) Reconstructed Scene from Predicted Poses

Fig. 5.3: A few qualitative 3-Dof pose estimation results on Walker Robot for RGBD cost function evaluation

5.1.2 Conveyor Object Manipulation

Task

Recent research has explored mobile manipulators being deployed for the task of pick-and-drop of moving objects on a conveyor [58, 59]. As argued in [59], a fast and accurate perception system is needed for robust pick-and-place of moving objects. However due to high runtime, deliberative pose estimation methods like PERCH are unable to be deployed for tasks such as these. In this section we demonstrate that due runtime enhancement, PERCH 2.0 can successfully estimate the 3-Dof poses of objects moving on a conveyor with high accuracy and speed.

Dataset

Since there is no publicly available database that consists of sequences of moving objects with known 3D models and varying 3-Dof poses, we constructed a database by collecting 10 videos of 4 YCB objects moving along the conveyor. The objects are shown in Figure 5.4. The 6-Dof camera pose was also captured along with intrinsic camera parameters. In order to annotate the ground truth 3-Dof pose, we first split the recorded videos into RGB and depth images. For each captured frame, we convert the corresponding 3D point cloud to the frame of the robot' base and then filter out all points except those belonging to the object of interest using pass-through filters in the PCL library [60]. Next we manually annotate the 3-Dof pose of the object in the first captured frame. For every successive frame, we use ICP alignment from PCL to annotate the pose using the annotation from the previous frame as the initial pose estimate. In this manner, we generate 961 RGB and depth images with annotated 3-Dof poses.

Baselines

For this experiment we use PERCH 2.0 with the same ICP as PERCH and kNN I (Section 4.4). Similar to the previous experiment, our objective here is to examine the adaptability of deliberative pose estimation methods like PERCH to this task as well as test the runtime improvement offered by PERCH 2.0. Consequently, we evaluate both algorithms on the generated conveyor dataset. We evaluate both algorithms against the BF-ICP baseline as before.

Metrics

We use three metrics to compare our results - area under ADD-S threshold curve for every object (ADD-S AUC $<$ 0.1m), the percentage of poses with ADD-S$<$2cm and the mean ADD-S error. The results are shown in Table II.

Accuracy

PERCH 2.0 and PERCH achieve comparable performance on the dataset with more than 99% of the pose swithin the 2cm ADD-S error limit. The slight difference in performance can be attributed to differences in downsampling used by the two methods. While PERCH uses Voxel downsampling in 3D point cloud space, PERCH 2.0 samples directly in the image space using a

(a) PR2 and Conveyor setup for data collection

(b) A few frames from the conveyor dataset

Fig. 5.4: A few extracted frames from the conveyor dataset

TABLE II: Evaluation of 3-Dof pose estimation of objects on a conveyor

Objects	BF-ICP[1]			PERCH [1]			PERCH 2.0 (ICP + kNN I)			Database
	AUC	<2cm	Mean ADD-S	AUC	<2cm	Mean ADD-S	AUC	<2cm	Mean ADD-S	Image Count
035_power_drill	91.38	100.000	0.0087	91.60	99.785	0.0085	91.37	99.57	0.0086	465
006_mustard_bottle	85.27	82.278	0.0149	91.93	98.101	0.0082	91.70	98.73	0.0084	158
004_sugar_box	87.85	96.842	0.0151	91.20	99.648	0.0089	91.06	99.30	0.0090	284
005_tomato_soup_can	92.25	92.593	0.0083	94.18	100.000	0.0062	94.85	100.00	0.0054	54
All Objects	89.29	95.738	0.0116	**91.61**	**99.479**	**0.0084**	91.47	99.38	0.0086	961
Mean Runtime (seconds)	8.065			25.938			**0.619**			
Mean Pose Count	58			60			56			

TABLE III: Maximum number of possible poses processed for given conveyor speed

Conveyor Speed (m/s)	Number of Poses Processed	
	PERCH	PERCH 2.0
0.025	2	90
0.050	1	45
0.100	1	23
0.150	0	15
0.200	0	11
0.250	0	9
0.300	0	8
0.350	0	6

stride to skip pixels. Both methods perform significantly better than BF-ICP, again highlighting the importance of rendering in achieving high accuracy.

Runtime

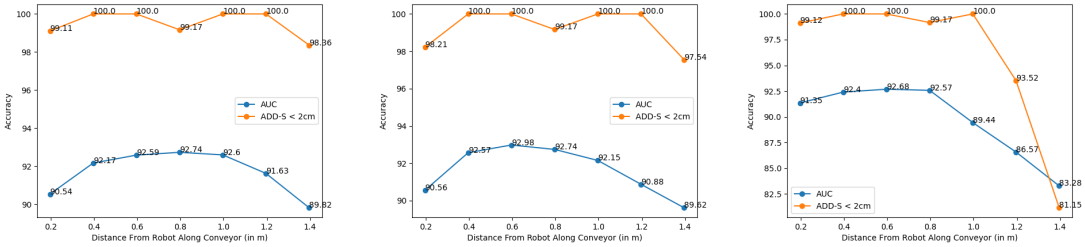
We can observe from Table II that PERCH 2.0 is significantly faster than PERCH and offers a $\sim$ 43X speedup. In order to understand the impact of runtime on the given task, we note that the region of the conveyor visible to the robot's Kinect camera is only 1.4m in length. Based on this length and for different speeds of the conveyor we can easily compute the number of pose estimates each method would be able to obtain in the visible region (as shown in Table III). It is evident from this table, that PERCH, due to its high runtime, is able to provide estimates for very low conveyor speeds only. Such low speeds are not practical in a real-world industrial scenario where conveyors need to run at high speeds to maintain high throughput. Moreover if planning and execution time for the robot are taken into account, the PERCH pose estimates may be infeasible even for low conveyor speeds. In contrast, PERCH 2.0 due to its enhanced runtime can offer several pose estimates for the moving object even at high conveyor speeds.

Spatial Analysis

A key component of any method being applied to the given task is the variation in the method's accuracy as the object moves from end of the conveyor to the other end closer to the robot. A method that can offer higher accuracy even if the object is at the far end of the conveyor allows sufficient time for planning and execution of the pickup motion by the robot arm before the object reaches within the robot workspace.

TABLE IV: Variation of accuracy metrics with distance from the robot along the conveyor

Distance From Robot (m)	BF-ICP		PERCH		PERCH 2.0	
	AUC	<2cm	AUC	<2cm	AUC	<2cm
0.0 - 0.2	91.35	99.12	90.54	99.11	90.56	98.21
0.2 - 0.4	92.40	100.0	92.17	100.0	92.56	100.0
0.4 - 0.6	92.68	100.0	92.59	100.0	92.98	100.0
0.6 - 0.8	92.57	99.17	92.74	99.17	92.74	99.17
0.8 - 1.0	89.44	100.0	92.60	100.0	92.15	100.0
1.0 - 1.2	86.57	93.52	91.63	100.0	90.87	100.0
1.2 - 1.4	83.28	81.15	89.82	98.36	89.62	97.54

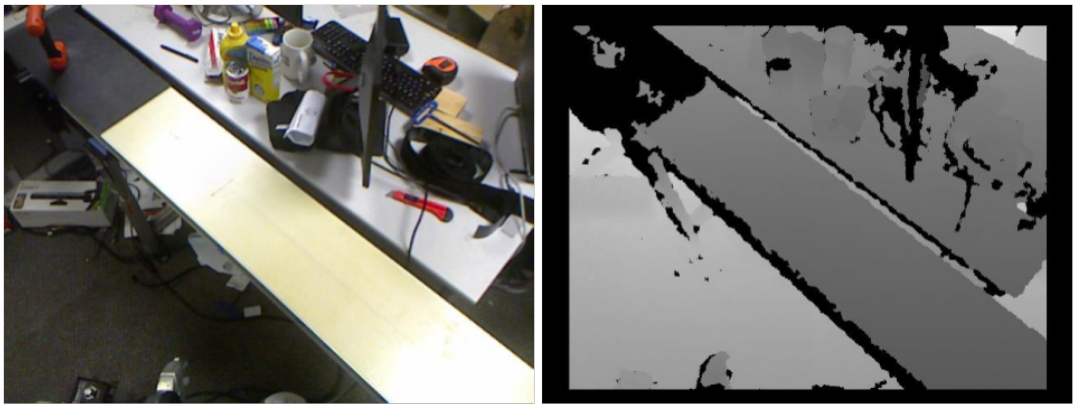


(a) PERCH (b) PERCH 2.0 (c) BF-ICP

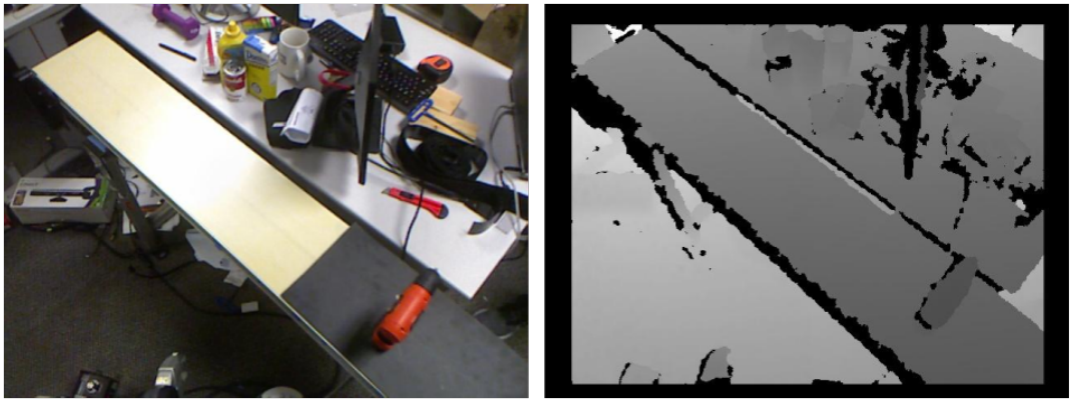
Fig. 5.5: Variation of pose estimation accuracy with distance from robot along the conveyor

In order to understand the spatial variation in accuracy for each of the methods, we divide the conveyor into bins of 0.2m and compute AUC as well as ADD-S<2cm for poses belonging to each bin. The results are shown in Table IV and Figure 5.6 for PERCH, PERCH 2.0 and BF-ICP.

The overall observation consistent with all three methods is that the pose estimation accuracy reduces when the object is too close or too far from the robot. A close inspection of scenes when the object is far away revealed significant depth holes in the captured depth images which inturn reduces the number of points available for pose estimation in the observed point cloud (Figure 5.6a). Similarly for some objects in certain poses such as the 035_power_drill, the visible parts of the object when its close to the robot are insufficient to discern the correct 3-Dof pose. This can be attributed to the creation of ambiguous top-down views of the object, again reducing the number of points available in the observed point cloud for pose estimation (Figure 5.6b). The second observation that is evident from Figure 5.6 is that the drop in accuracy when the object is far way is significantly higher for BF-ICP than for PERCH and PERCH 2.0. The drop clearly highlights that rendering plays a key role in determining the correct 3-Dof pose when objects are far from the camera and only few points from the object are actually visible. It also points to PERCH 2.0 being a favorable candidate for the given task owing to its high runtime when compared to PERCH and higher spatially consistent accuracy when compared to BF-ICP.



(a) Depth holes in far away objects



(b) Objects in close proximity result in ambiguous top-down views

Fig. 5.6: Difficult scenarios for pose estimation of objects moving along the conveyor

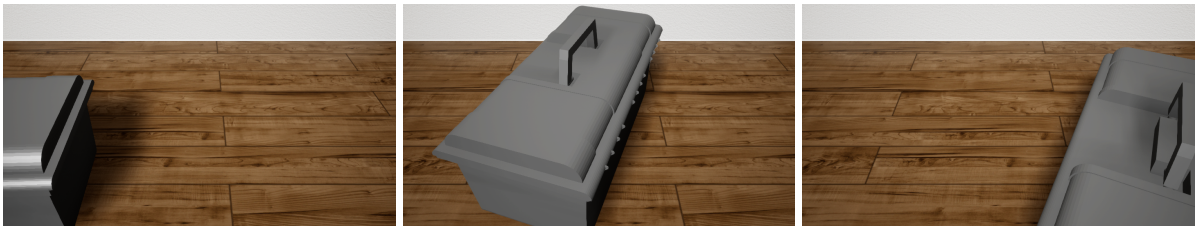


Fig. 5.7: A few sample scenes from the crate test dataset

5.1.3 Object Articulation

Task

PERCH 2.0 was evaluated for the task of object articulation on the RoMaN platform shown in Figure 5.9(a) where the robot is required to autonomously open a container such as a crate. Specifically, PERCH 2.0 estimates the 3-Dof pose of the crate (x, y, yaw) to be opened which then serves as an input for the manipulation planner. The task of opening a container such as a crate, imposes stringent accuracy requirements on estimated pose due to manipulation work space limitations of the robot and the necessity to locate the handle of the crate which occupies a limited area with respect to the entire crate. In addition, the size of the crate and its close proximity to the robot for the task of crate opening, renders large parts of the crate occluded from the camera, thus making the pose estimation accuracy requirement harder to meet.

Dataset

For evaluating PERCH 2.0 based pose estimation of the crate and to compare its accuracy and runtime against the accuracy of PERCH, we require multiple scenes consisting of the crate with varying (x, y, yaw) and different degrees of visibility. In addition, computation of accuracy requires ground truth pose annotations for each scene, a task which is difficult to obtain in the real world. Subsequently, we constructed a synthetic photo-realistic dataset of 25 scenes containing the crate with corresponding RGB and depth images using NVidia NDDS [57] plugin for Unreal Engine 4 (as shown in Figure 5.7) as before. Within the plugin, we randomly vary 3-Dof pose (x, y, yaw) of the crate located on the ground. The height of the camera is fixed according to the height of the RealSense mounted on RoMaN and scenes bearing close resemblance to real world requirements where the crate is highly occluded and in close proximity to the RealSense are generated.

Baselines

In this experiment, we use two variants of PERCH 2.0. The first variant is PERCH 2.0 A (ICP + kNN I) used in experiments in Section 5.1.1 and 5.1.2. The second variant is PERCH 2.0 (M2M GICP + kNN II) which uses the proposed parallel many to many GICP approach (Section 4.3) and kNN II (Section 4.4). We compare these two variants with PERCH and BF-ICP baseline. For running PERCH 2.0, a machine with a P100 16GB GPU with 8 CPU cores is used. For PERCH a machine with 8 CPU cores is used.

TABLE V: Evaluation of 3-Dof pose estimation of crate

Objects	BF-ICP[1]			PERCH[1]			PERCH 2.0 A (ICP + kNN I)			PERCH 2.0 (M2M GICP + kNN II)		
	AUC	<2cm	Mean ADD-S	AUC	<2cm	Mean ADD-S	AUC	<2cm	Mean ADD-S	AUC	<2cm	Mean ADD-S
crate	51.12	24.00	0.0609	88.41	96.00	0.0128	90.35	96.00	0.0106	**93.14**	**96.00**	**0.0077**
Mean Runtime (s)		2.47			162.36			5.07			**3.23**	
Poses Rendered		773			752			720			737	

Metrics

We use 3 metrics to compare our results - area under ADD-S threshold curve for every object ADD-S AUC <0.1m, the percentage of poses with ADD-S<2cm and the mean ADD-S error. The results are shown in Table I.

Accuracy

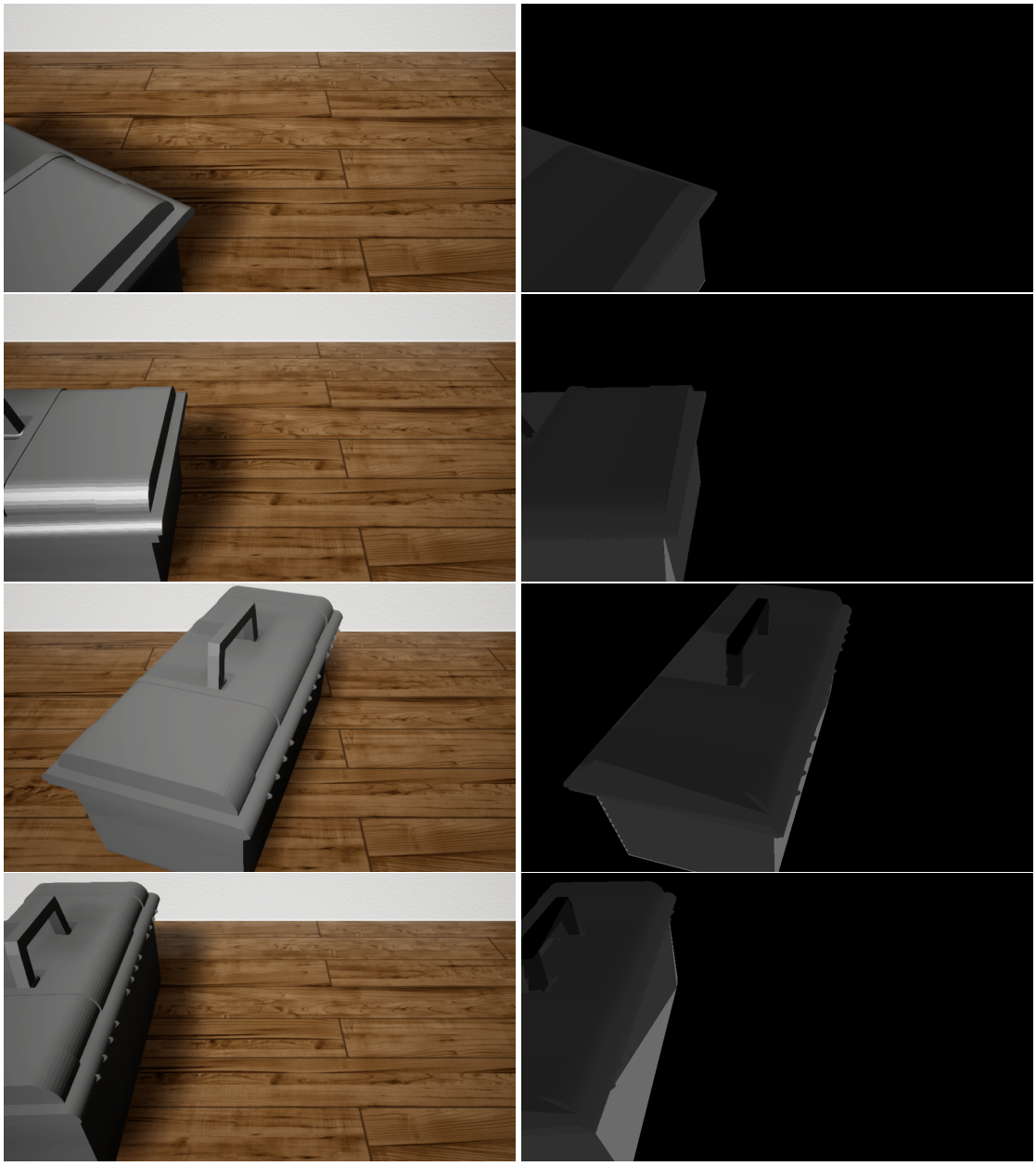
The first observation is that PERCH 2.0 A and PERCH achieve comparable performance on the ADD-S<2cm metric with 96% of the poses within the 2cm ADD-S error limit. However PERCH 2.0 (M2M GICP + kNN II) performs significantly better than the other approaches on the metrics ADD-S AUC(<0.1m) and mean ADD-S error. This shows that GICP improves accuracy by performing better alignment under occlusion. We note that all PERCH variants have a significantly better accuracy than the BF-ICP approach which highlights the importance of rendering and using the PERCH cost function in the estimation of the crate pose especially when large parts of the crate are occluded.

Runtime

The results show that PERCH 2.0 (M2M GICP + kNN II) is $\sim$50X faster that PERCH on the given dataset. It is also $\sim$1.5X faster than PERCH 2.0 (ICP + kNN Approach I) which shows that for a large number of candidate poses, parallel GICP as well as kNN Approach II help in reducing runtime further. For all methods the average number of poses rendered is within 700-800.

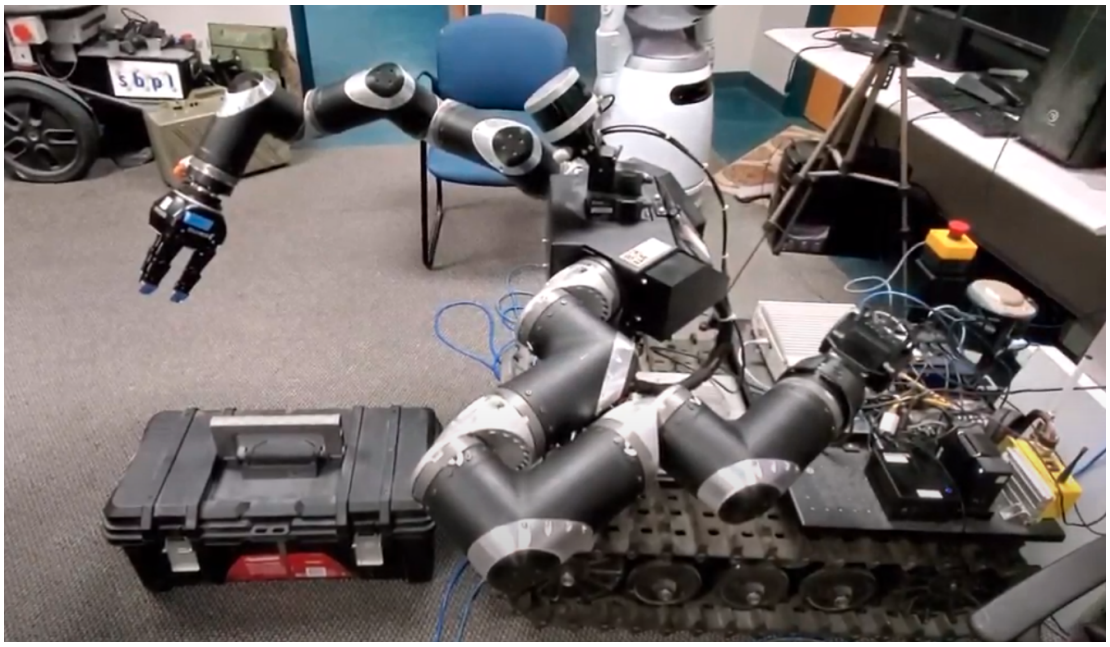
Qualitative Results

Figure 5.8 shows a few scenes from crate database with corresponding predicted poses by PERCH 2.0. The predictions highlight the capability of PERCH 2.0 in being able to accurately estimate the 3-Dof pose even under high occlusion. Figure 5.9 represents scenes and predictions captured from the RoMaN robot and show that the runtime and accuracy capabilities of PERCH 2.0 transfer well to the real world. Note that the minor discrepancies in the predicted pose are due to the crate model being slightly different from the actual crate used.

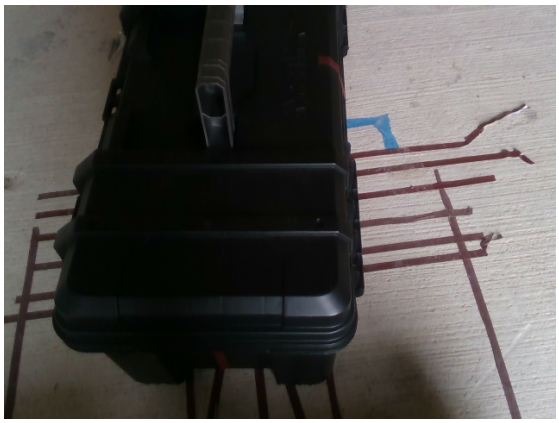


(a) Input scene (b) Reconstructed scene with predicted poses

Fig. 5.8: A few results from the crate test dataset

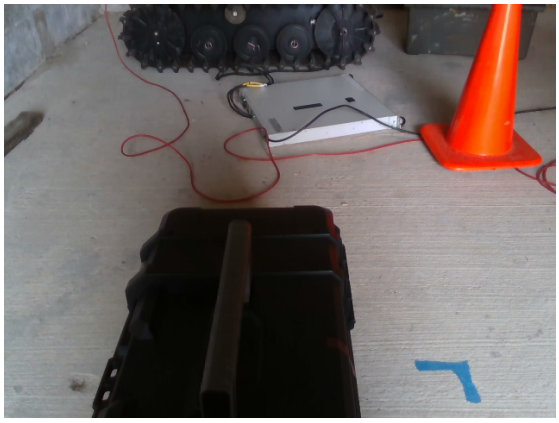


(a) RoMaN setup for crate opening



(b) Input scene

(c) Reconstructed scene (Runtime : 3.0s)



(d) Input scene

(e) Reconstructed scene (Runtime : 1.4s)

Fig. 5.9: A few test scenes from the RoMaN robot platform

5.2 6-Dof Pose Estimation

5.2.1 Tabletop Object Manipulation

Baselines

In order to evaluate the performance of PERCH 2.0 for 6-Dof pose estimation, we compare our results with DenseFusion [37] and PoseCNN + ICP [6] on objects from the YCB-Video Dataset [6]. The results are computed for the 2,949 keyframes used for testing in prior works. Additionally, we use the following variants of PERCH 2.0 :

- *PERCH 2.0 A (PoseCNN Mask + M2M GICP + kNN II)* : Here we directly instance segmentation masks published online by PoseCNN[1] in conjunction with *PERCH 2.0 (M2M GICP + kNN II)* used in experiments in Section 5.1.3. Since the mask is directly used, we do not have the full bounding boxes available. The bounding box is computed from the mask provided and the center of bounding box is used in PERCH 2.0. We note that this is the visible bounding box as opposed to the full bounding box that our method proposes to use.
- *PERCH 2.0 B (MaskRCNN + M2M GICP + kNN II)* : Here we use a MaskRCNN model that outputs instance segmentation masks and full bounding boxes in conjunction with *PERCH 2.0 (M2M GICP + kNN II)*. The MaskRCNN implementation is described in Section 5.2.1 below.
- *PERCH 2.0 C (GT Mask + M2M GICP + kNN II)* : Here we use the ground truth instance segmentation masks and full bounding boxes in conjunction with *PERCH 2.0 (M2M GICP + kNN II)* used previously in Section 5.1.3.

For running the above PERCH 2.0 variants, we use a machine with 16 GB P100 NVidia GPU and 32 CPU cores. For rotation proposals, we sample 80 viewpoints from the unit sphere and combine it with 3 in-plane rotation angles (for unsymmetrical objects).

Metrics

We use two metrics to compare our 6-Dof pose estimation results - area under ADD-S threshold curve (ADD-S AUC $<$ 0.1m) and the percentage of poses with ADD-S$<$2cm. The results are shown in Table VI. For understanding the accuracy of our instance segmentation and bounding box regression, we use the average precision or AP metric [51]. The results are shown in Table VII.

MaskRCNN Implementation Details

In order to train a CNN for instance segmentation and full bounding box detection, we use a Mask RCNN [51] model to train on RGB images, segmentation and class labels from the 80 training videos provided in the YCB-Video Dataset. Our implementation is based on [61]. However since the YCB-Video dataset doesn't contain annotation for full bounding boxes, we use the ground truth 6-Dof pose and project it onto the image to obtain the full bounding box. We note that

[1] https://rse-lab.cs.washington.edu/projects/posecnn/

though we are using the 6-Dof pose to obtain the full bounding box, this could instead be done through crowdsourced human annotation as done for the CrowdHuman [4] dataset, eliminating the need to collect 6-Dof pose annotations. The training is performed on 4 NVidia V100 GPUs.

Accuracy

The results in Table VI show that without using any ground truth pose annotation, the variant *PERCH 2.0 A* outperforms all external baselines on both AUC($<$0.1m) and ADD-S$<$2cm metrics even though it uses the same PoseCNN Mask. We observe that the variant *PERCH 2.0 B* that uses full bounding boxes further improves on the accuracy and performs well across objects of varying shape, size, texture and symmetry, estimating 99.29% of the poses within the 2cm ADD-S error and hence within the tolerance limit of most robot grippers. However unlike DenseFusion [37] and PoseCNN + ICP [6], no pose estimation network on images from the 80 training videos and 80,000 synthetic images was trained. The results in Table VII show that our MaskRCNN model is able to accurately estimate full bounding boxes and instance segmentation masks for objects of different shapes and sizes.

Robustness

A comparison between *PERCH 2.0 C* or *PERCH 2.0 B* with *PERCH 2.0 A* shows the improvement that can be achieved with the help of full bounding boxes. Specifically, a significant jump in accuracy metrics is seen for highly occluded objects such as the 003_cracker_box, 024_bowl, and 021_bleach_cleanser.

Runtime

We use two variants of PERCH 2.0 for runtime evaluation. *PERCH 2.0 A* uses kNN I and the publicly available CPU parallelized version of GICP. *PERCH 2.0 B* uses kNN II and our proposed parallel M2M GICP. From Table VIII, we can observe that *PERCH 2.0 A* on 6-Dof poses takes an average of 75.43s to estimate poses for all objects in the scene. A closer inspection reveals that out of the 75.4s an average of 88.9% time is spent on refining poses through ICP while other parts of the pipeline take an average of only 8.4s to run for all objects and poses in the scene. In contrast, *PERCH 2.0 B* takes only 7.6s on average to estimate poses for all objects in the scene. It achieves a $\sim$10X improvement over *PERCH 2.0 A* highlighting the importance of parallel GICP and kNN II when the number of poses to be evaluated is high. For both variants, an average of 2400 poses are evaluated per scene for all objects combined.

Qualitative Analysis

A few test scenes from the YCB Video Dataset along with corresponding bounding box, instance segmentation and 6-Dof pose predictions are shown in Figure 5.10. From the first scene, we can observe that despite severe occlusion of the 037_scissors object, our Mask RCNN model is able to accurately localize the full 2D bounding box and hence the object center in the image. The predicted poses in scenes show that 6-Dof poses of objects of various shapes, sizes, symmetries with varying levels of occlusion are predicted accurately.

TABLE VI: Evaluation of 6-Dof pose estimation on objects from the YCB Video Dataset [6]

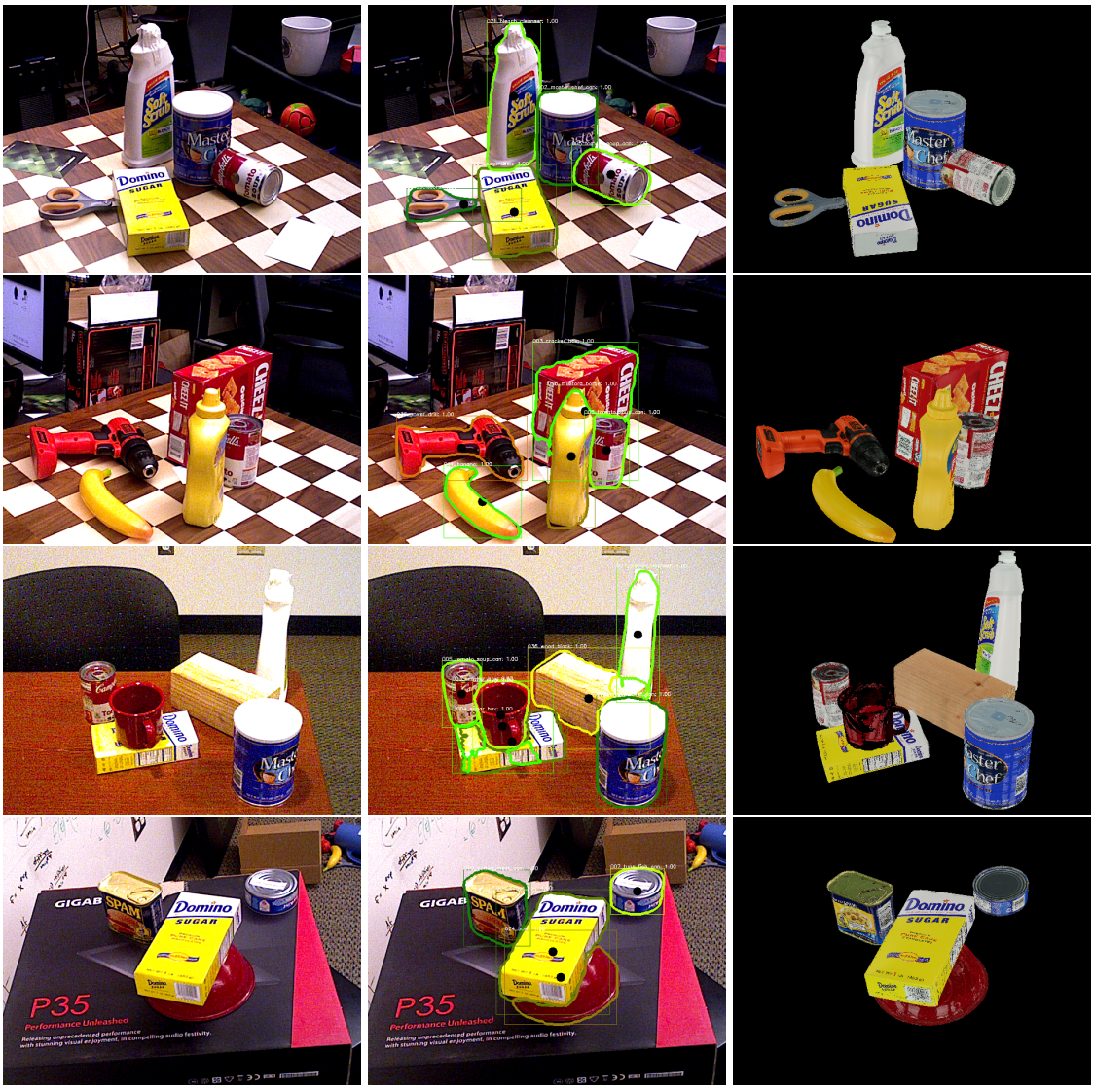
Objects	PoseCNN + ICP [6]		DenseFusion (Per-Pixel) [37]		DenseFusion (Iterative) [37]		PERCH 2.0 A (PoseCNN Mask)		PERCH 2.0 B (MaskRCNN Mask)		PERCH 2.0 C (GT Mask)	
	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm
002_master_chef_can	95.80	100.00	95.20	100.00	96.40	100.00	**96.06**	100.00	**96.25**	100.00	95.99	**100.00**
003_cracker_box	92.70	91.60	92.50	99.30	**95.50**	**99.50**	93.54	97.81	94.69	99.65	94.78	99.42
004_sugar_box	98.20	100.00	95.10	100.00	**97.50**	**100.00**	95.86	99.66	96.11	99.58	96.20	99.69
005_tomato_soup_can	94.50	96.90	93.70	96.90	94.60	96.90	97.26	99.77	97.30	100.00	**97.31**	**100.00**
006_mustard_bottle	98.60	100.00	95.90	100.00	97.20	100.00	97.51	100.00	97.42	100.00	**97.56**	**100.00**
007_tuna_fish_can	97.10	100.00	94.90	100.00	**96.60**	100.00	95.50	99.91	95.97	**100.00**	95.37	99.91
008_pudding_box	97.90	100.00	94.70	100.00	**96.50**	100.00	93.04	94.03	93.55	99.53	94.52	**100.00**
009_gelatin_box	98.80	100.00	95.80	100.00	**98.10**	100.00	96.77	100.00	96.56	100.00	96.50	**100.00**
010_potted_meat_can	92.70	93.60	90.10	93.10	91.30	93.10	95.13	97.82	95.45	99.72	**95.81**	**99.72**
011_banana	97.10	99.70	91.50	93.90	96.60	100.00	96.53	99.74	**96.88**	99.74	96.78	**100.00**
019_pitcher_base	97.80	100.00	94.60	100.00	**97.10**	100.00	92.37	100.00	92.11	100.00	92.38	**100.00**
021_bleach_cleanser	96.90	99.40	94.30	99.80	**95.80**	100.00	93.39	96.99	95.25	100.00	95.57	**100.00**
024 bowl	81.00	54.90	86.60	69.50	88.20	98.80	93.42	97.04	97.22	100.00	**97.37**	**100.00**
025_mug	95.00	99.80	95.50	100.00	**97.10**	100.00	96.96	100.00	96.96	100.00	97.09	**100.00**
035_power_drill	98.20	99.60	92.40	97.10	96.00	98.70	96.10	99.91	95.72	99.72	**96.30**	**100.00**
036_wood_block	87.60	80.20	85.50	93.40	89.70	**94.60**	90.31	90.08	91.58	93.61	94.47	**100.00**
037_scissors	91.70	95.60	96.40	100.00	95.20	100.00	95.11	100.00	96.49	100.00	**96.53**	**100.00**
040_large_marker	97.20	99.70	94.70	99.20	97.50	100.00	97.56	99.85	97.78	100.00	**97.99**	**100.00**
051_large_clamp	75.20	74.90	71.60	78.50	72.90	79.20	72.25	77.06	92.41	97.99	**92.75**	**99.44**
052_extra_large_clamp	64.40	48.80	69.00	69.50	69.80	76.30	86.12	82.58	88.54	90.24	**92.74**	**94.23**
061_foam_brick	97.20	100.00	92.40	100.00	92.50	100.00	95.89	100.00	95.72	100.00	**96.58**	**100.00**
All Objects	93.00	93.20	91.20	95.30	93.10	96.80	94.56	98.00	95.48	99.29	**95.72**	**99.61**

TABLE VII: Evaluation of bounding box regression and instance segmentation on the YCB Video Dataset [6] using Mask-RCNN

Objects	Bounding Box Regression						Instance Segmentation					
	AP	AP50	AP75	APs	APm	APl	AP	AP50	AP75	APs	APm	APl
002_master_chef_can	0.825	1.000	1.000	-1.000	-1.000	0.825	0.711	1.000	0.854	-1.000	-1.000	0.744
003_cracker_box	0.785	1.000	0.990	-1.000	-1.000	0.785	0.693	1.000	0.781	-1.000	-1.000	0.693
004_sugar_box	0.849	1.000	0.990	-1.000	-1.000	0.849	0.837	1.000	1.000	-1.000	-1.000	0.837
005_tomato_soup_can	0.794	0.980	0.970	0.042	0.804	0.805	0.772	0.970	0.948	0.000	0.730	0.803
006_mustard_bottle	0.890	1.000	1.000	-1.000	-1.000	0.890	0.851	1.000	1.000	-1.000	-1.000	0.851
007_tuna_fish_can	0.824	1.000	1.000	-1.000	0.828	0.822	0.816	1.000	0.989	-1.000	0.810	0.839
008_pudding_box	0.720	0.958	0.938	-1.000	0.745	0.728	0.548	0.958	0.548	-1.000	0.495	0.618
009_gelatin_box	0.823	1.000	1.000	-1.000	-1.000	0.823	0.846	1.000	0.970	-1.000	-1.000	0.846
010_potted_meat_can	0.720	0.938	0.842	-1.000	0.565	0.843	0.759	0.990	0.882	-1.000	0.657	0.849
011_banana	0.802	1.000	0.975	-1.000	0.822	0.814	0.807	1.000	1.000	-1.000	0.807	0.810
019_pitcher_base	0.908	1.000	1.000	-1.000	-1.000	0.908	0.931	1.000	1.000	-1.000	-1.000	0.931
021_bleach_cleanser	0.773	1.000	0.956	-1.000	0.608	0.785	0.811	1.000	0.988	-1.000	0.684	0.821
024_bowl	0.726	1.000	0.867	-1.000	0.451	0.731	0.812	0.987	0.969	-1.000	0.026	0.828
025_mug	0.817	1.000	1.000	-1.000	0.854	0.795	0.665	1.000	0.861	-1.000	0.656	0.676
035_power_drill	0.724	1.000	0.939	-1.000	-1.000	0.724	0.629	1.000	0.721	-1.000	-1.000	0.631
036_wood_block	0.557	0.693	0.679	-1.000	-1.000	0.557	0.483	0.692	0.581	-1.000	-1.000	0.485
037_scissors	0.368	0.748	0.309	-1.000	0.416	-1.000	0.471	0.798	0.606	-1.000	0.477	-1.000
040_large_marker	0.717	1.000	0.906	-1.000	0.727	-1.000	0.497	0.984	0.441	-1.000	0.497	-1.000
051_large_clamp	0.456	0.658	0.578	-1.000	0.539	0.314	0.316	0.653	0.160	-1.000	0.303	0.397
052_extra_large_clamp	0.513	0.774	0.725	-1.000	0.601	0.538	0.437	0.774	0.492	-1.000	0.095	0.572
061_foam_brick	0.798	1.000	0.948	-1.000	0.804	0.848	0.752	1.000	0.969	-1.000	0.742	0.867
Mean AP	0.733	0.940	0.886	0.042	0.674	0.757	0.688	0.943	0.798	0.000	0.537	0.742

TABLE VIII: Evaluation of runtime on YCB Video Dataset

Method	Runtime (s)
PoseCNN + ICP	10
DenseFusion (Iterative)	0.06
PERCH 2.0 A (kNN I + CPU GICP)	75.43
PERCH 2.0 B (kNN II + M2M GICP)	7.6



(a) Input scene (b) 2D object detection (c) Predicted 6-Dof poses

Fig. 5.10: Results from the YCB Video Dataset

Chapter 6

Conclusion

In this work we introduced PERCH 2.0, a parallel GPU-based deliberative pose estimation methodology that finds the best explanation of an observed scene by searching over a space of possible rendered scenes. The proposed methodology scales proportionally with the number of objects and number of poses to be considered for each object by breaking down every task into the smallest possible unit which can fully utilize parallelization capabilities of the GPU.

Within the proposed deliberative framework we first render a set of candidate poses for every object of interest in the scene. Our GPU based renderer is capable of rendering thousands of scenes at a time and inherently accounts for occlusion from other objects by using the input depth data to mark certain points in a given scene as extraneous occluders. Rendering is followed by 3D point cloud creation wherein the camera intrinsic parameters are used to simultaneously project every valid pixel in the rendered scenes to 3D space. The 3D point clouds are fed into our many to many GICP framework that aligns every pose to the corresponding observed point cloud. In order to counter the sequential nature of ICP iteration, we perform ICP iterations for all point clouds in parallel, thereby reducing a multi-sequential step of aligning several point clouds to a single sequential step. The updated poses after GICP are rendered again and converted to 3D point clouds. In the final step, we evaluate the candidate poses by computing a cost function that relies on a parallel nearest neighbour search of every rendered 3D point in the observed scene.

Our experiments showed that PERCH 2.0 is an order of magnitude faster than its predecessor PERCH [1] for 3-Dof pose estimation. As a direct result of the speedup, we demonstrated the application of PERCH 2.0 to new domains such as object articulation and object pose estimation of moving objects on a conveyor belt. For object articulation where the designated task was to estimate the 3-Dof pose of a large crate, we showed that despite high occlusion due to close proximity of the crate to the robot, our method is able to achieve high accuracy and low runtimes. In the case of moving objects on a conveyor, PERCH 2.0 retains high accuracy even when the object is at the far end of the conveyor belt and offers several pose estimates as the object gets closer. Intuitively, both capabilities are favourable for a planner looking to grasp the object as it reaches within the robot workspace.

For 6-Dof pose estimation where the number of poses to consider for every object increases exponentially, we demonstrated a successful and scalable integration of a discriminative learning based 2D object detector with the proposed deliberative methodology PERCH 2.0. The combined framework allows prediction of 6-Dof poses directly from an instance segmentation mask generated by the 2D object detector, thus eliminating the need for ground truth 6-Dof poses in the training database used by the detector. Our 6-Dof experiments demonstrated that our framework can achieve higher accuracy than state of the art discriminative methods that don't have a deliberative component. Additionally, we showed the impact of localizing full bounding boxes as opposed to visible bounding boxes on overall pose estimation accuracy through assistance in occlusion handling.

6.1 Future Work

Our proposed framework provides several directions for future work on deliberative pose estimation by formulating the deliberative methodology in a more scalable and practically feasible manner.

For 3-Dof pose estimation, so far only indoor scenes have been explored in our work. This could be extended to outdoor scenes where Lidar data can be used to accurately estimate 3-Dof poses of vehicles by self-driving cars. Since outdoor scenes as well typically deal with higher distance magnitudes, a voxelized version of PERCH 2.0 can be developed to speed up computation even further. An extension to vehicle pose detection also presents an opportunity to extend deliberative pose estimation to scenarios where it may not be possible to have exact models of all objects.

In the context of 6-Dof pose estimation, we presented a framework that elegantly combines discriminative, deliberative and optimization based approaches for pose prediction. However, in our method, predictions flow from the discriminative side to the deliberative side but there is no feedback of information. As an extension, the costs predicted for each of the poses in the pose hypothesis could be used to learn a sampling distribution that takes the depth image of the scene as input. This sampling distribution can then be used to sample poses by PERCH 2.0 on the deliberative side instead of uniformly sampling the rotation space. This would enable a further speedup of the deliberative part of the framework and also ensure a continually improving self-supervised discriminative methodology that enables the deliberative part to get faster as it receives more data.

Bibliography

[1] V. Narayanan and M. Likhachev, "PERCH: Perception via search for multi-object recognition and localization," in *Proceedings of IEEE ICRA*, vol. 2016-June, 2016, pp. 5052–5059. (document), 1.2, 1.3, 2.2, 2.3, 3, 3.1.1, 3.1, I, 3.2, 4.3, 4.5, 5.1.1, I, 5.1.1, II, V, 6

[2] ——, "Discriminatively-guided Deliberative Perception for Pose Estimation of Multiple 3D Object Instances." in *RSS*, 2016. (document), 1.2, 1.3, 2.2, 3.2.1, 3.3

[3] ——, "Deliberative object pose estimation in clutter," in *Proceedings - IEEE ICRA*, jul 2017, pp. 3125–3130. (document), 1.2, 1.3, 2.2, 3.2.2, 3.4, 4.5

[4] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, "Crowdhuman: A benchmark for detecting human in a crowd," *arXiv preprint arXiv:1805.00123*, 2018. (document), 4.6.1, 4.8, 5.2.1

[5] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in Manipulation Research: Using the Yale-CMU-Berkeley Object and Model Set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, sep 2015. (document), 4.9

[6] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2018. (document), 1.2, 2.1, 2.3, 4.6.1, 5.1.1, 5.2.1, 5.2.1, VI, VII

[7] 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 *2011 IEEE ICCV workshops*, 2011, pp. 585–592. 1.2, 2.1

[8] 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 *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. Springer, 2012, pp. 113–122. 1.2, 2.1

[9] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints," *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006. 1.2, 2.1

[10] M. R. Stevens and J. R. Beveridge, *Integrating graphics and vision for object recognition*. Springer Science & Business Media, 2000, vol. 589. 1.2, 2.1

[11] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *Proceedings of the IEEE CVPR*, 2015, pp. 3109–3118. 1.2, 2.1

[12] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *Proceedings of the IEEE CVPR*,

2015, pp. 1510–1519. 1.2, 2.1

[13] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DOF object pose from semantic keypoints," in *2017 IEEE ICRA*. IEEE, 2017, pp. 2011–2018. 1.2, 2.1

[14] J. Liu and S. He, "6D Object Pose Estimation without PnP," *CoRR*, vol. abs/1902.01728, 2019. 1.2, 2.1

[15] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE CVPR*, 2017, pp. 7074–7082. 1.2, 2.1

[16] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," in *2017 IROS*. IEEE, 2017, pp. 545–551. 1.2, 2.1

[17] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from rgb images," in *Proceedings of ECCV*, 2018, pp. 699–715. 1.2, 2.1

[18] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi, "Discovery of latent 3D keypoints via end-to-end geometric reasoning," in *NIPS*, 2018, pp. 2059–2070. 1.2, 2.1

[19] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *Proceedings of the IEEE CVPR*, 2018, pp. 292–301. 1.2, 2.1

[20] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," *CoRR*, vol. abs/1703.10896, 2017. 1.2, 2.1

[21] E. Corona, K. Kundu, and S. Fidler, "Pose Estimation for Objects with Rotational Symmetry," in *IEEE IROS*, 2018, pp. 7215–7222. 1.2, 2.1

[22] C. Mitash, A. Boularias, and K. E. Bekris, "Improving 6D pose estimation of objects in clutter via physics-aware Monte Carlo tree search," in *2018 IEEE ICRA*. IEEE, 2018, pp. 1–8. 1.2, 2.1

[23] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE CVPR*, 2019, pp. 3343–3352. 1.2, 2.1

[24] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," in *CoRL*, 2018. 1.2, 2.1, 5.1.1, I

[25] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proceedings of the IEEE ICCV*, 2017, pp. 1521–1529. 1.2, 2.1, 4.6.2

[26] M. Stevens and J. Beveridge, "Localized Scene Interpretation from 3D Models, Range, and Optical Data," *Computer Vision and Image Understanding*, vol. 80, pp. 111–129, 02 2000. 1.2, 2.2

[27] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze, "A global hypotheses verification

method for 3D object recognition," in *European conference on computer vision*. Springer, 2012, pp. 511–524. 1.2, 2.2

[28] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 433–449, 1999. 2.1

[29] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *2009 IEEE ICRA*. IEEE, 2009, pp. 3212–3217. 2.1

[30] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010, pp. 356–369. 2.1

[31] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the viewpoint feature histogram," in *2010 IEEE IROS*, 2010, pp. 2155–2162. 2.1

[32] D. G. Lowe, "Object recognition from local scale-invariant features." in *ICCV*, vol. 99, no. 2, 1999, pp. 1150–1157. 2.1

[33] 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 *Asian conference on computer vision*. Springer, 2012, pp. 548–562. 2.1, 5.1.1

[34] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2011. 2.1

[35] Z. Cao, Y. Sheikh, and N. K. Banerjee, "Real-time scalable 6DOF pose estimation for textureless objects," in *2016 IEEE ICRA*. IEEE, 2016, pp. 2441–2448. 2.1

[36] G. Billings and M. Johnson-Roberson, "SilhoNet: An RGB Method for 3D Object Pose Estimation and Grasp Planning," sep 2018. 2.1

[37] C. Wang, D. Xu, Y. Zhu, R. Martin-Martin, C. Lu, L. Fei-Fei, and S. Savarese, "DenseFusion: 6D object pose estimation by iterative dense fusion," in *Proceedings of the IEEE CVPR*, vol. 2019-June, 2019, pp. 3338–3347. 2.1, 4.6.1, 5.2.1, 5.2.1, VI

[38] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011. [Online]. Available: https://doi.org/10.1177/1745691610393980 2.1

[39] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, "Label Fusion: A Pipeline for Generating Ground Truth Labels for Real RGBD Data of Cluttered Scenes," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018, pp. 3235–3242. 2.1

[40] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "Poserbpf: A rao-blackwellized particle filter for 6d object pose tracking," *CoRR*, vol. abs/1905.09304, 2019. 2.1

[41] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *The European Conference on Computer Vision (ECCV)*, September 2018. 2.1, 4.6.2

[42] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992. 2.3, 3.1.1

[43] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP." 2.3, 4.3

[44] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Voxelized gicp for fast and accurate 3d point cloud registration," EasyChair Preprint no. 2703, EasyChair, 2020. 2.3, 4.3, 4.3

[45] "SegICP: Integrated deep semantic segmentation and pose estimation," in *IEEE International Conference on Intelligent Robots and Systems*, 2017, pp. 5784–5789. 2.3

[46] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975. 3.1.1

[47] G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations," *Color Research & Application*, vol. 30, no. 1, pp. 21–30, 2005. 4.1, 4.1

[48] "cublas," https://developer.nvidia.com/cublas, accessed: 2020-05-20. 4.3

[49] "cudnn," https://developer.nvidia.com/cudnn, accessed: 2020-05-20. 4.3

[50] V. Garcia, Debreuve, F. Nielsen, and M. Barlaud, "K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching," in *2010 IEEE International Conference on Image Processing*, Sep. 2010, pp. 3757–3760. 4.4

[51] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE ICCV*, 2017, pp. 2980–2988. 4.6.1, 5.2.1, 5.2.1

[52] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271. 4.6.1

[53] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37. 4.6.1

[54] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440. 4.6.1

[55] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017. 4.6.1

[56] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Occlusion-Aware R-CNN: Detecting Pedestrians in a Crowd," Tech. Rep., 2018. 4.6.1

[57] T. To, J. Tremblay, D. McKay, Y. Yamaguchi, K. Leung, A. Balanon, J. Cheng, W. Hodge, and S. Birchfield, "NDDS: NVIDIA deep learning dataset synthesizer," 2018. 5.1.1, 5.1.1, 5.1.3

[58] A. Menon, B. Cohen, and M. Likhachev, "Motion planning for smooth pickup of moving objects," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., sep 2014, pp. 453–460. 5.1.2

[59] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and mo-

tion planning for pick-and-place of dynamic objects," in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 816–823. 5.1.2

[60] R. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 1 –4. 5.1.2

[61] F. Massa and R. Girshick, "maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch," https://github.com/facebookresearch/maskrcnn-benchmark, 2018, accessed: [Insert date here]. 5.2.1