

# ZePHyR: Zero-shot Pose Hypothesis Rating

Brian Okorn\*, Qiao Gu\*, Martial Hebert and David Held

**Abstract**—Pose estimation is a basic module in many robot manipulation pipelines. Estimating the pose of objects in the environment can be useful for grasping, motion planning, or manipulation. However, current state-of-the-art methods for pose estimation either rely on large annotated training sets or simulated data. Further, the long training times for these methods prohibit quick interaction with novel objects. To address these issues, we introduce a novel method for zero-shot object pose estimation in clutter. Our approach uses a hypothesis generation and scoring framework, with a focus on learning a scoring function that generalizes to objects not used for training. We achieve zero-shot generalization by rating hypotheses as a function of unordered point differences. We evaluate our method on challenging datasets with both textured and untextured objects in cluttered scenes and demonstrate that our method significantly outperforms previous methods on this task. We also demonstrate how our system can be used by quickly scanning and building a model of a novel object, which can immediately be used by our method for pose estimation. Our work allows users to estimate the pose of novel objects without requiring any retraining. Additional information can be found on our website <https://bokorn.github.io/zephyr/>

## I. INTRODUCTION

6D pose describes the position and orientation of an object, defined in a reference frame relative to a predefined model of the object. An object’s 6D pose fully describes the state of a static rigid object and, as such, is commonly used as a representation for planning [1], [2]. A robot can use an estimate of an object’s pose to perform complex manipulation interactions with the object [3], [4], [5], [6].

Current state-of-the-art methods for object pose estimation train a new model for each object they are being evaluated on [7], [8], [9]. This requires a large amount of annotated training data, either produced by capturing and annotating large datasets or through rendering the object in synthetically generated scenes. For example, the YCB-Video dataset [7] contains 133,827 human-annotated images with roughly 25,000 images per object. Although this dataset has enabled the training of powerful deep learning methods [7], [8], curating such a human-labeled dataset (including both capturing a diverse dataset and labeling the data) for each new object that a robot must interact with is cumbersome. Methods that rely on purely simulated data [10], [11], [12] avoid this limitation but must instead contend with the sim2real gap between the synthetic data and real sensor observations. Improved rendering [13] and domain randomization techniques [14] have been suggested to alleviate this gap, but ensuring

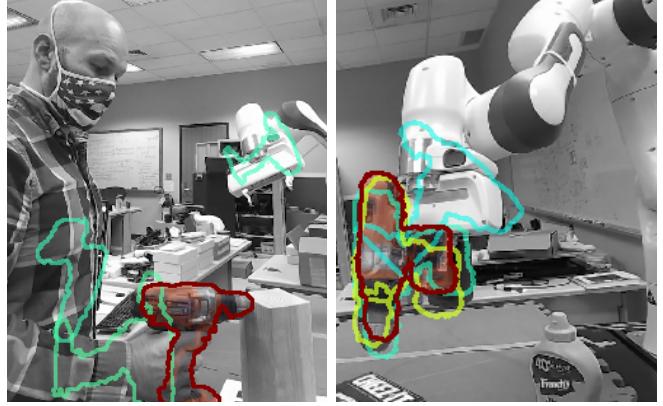


Fig. 1: Pose hypotheses scored using Zero-shot Pose Hypothesis Rating on novel drill object, reconstructed at test time. The highest scoring pose is rendered in color. Poses are outlined in color corresponding to score, with highly-rated poses in red and to lower ones in blue.

that the simulated data accurately represents the variations observed in the real world continues to be an open problem.

Regardless of how this data is obtained, training new networks has a time and space cost. This training can take many hours, which prevents robots using such systems from quickly being able to interact with new objects. Additionally, new network weights are trained for each new object, which presents a difficulty for memory-constrained robot systems. These constraints do not scale well in cases where robots need to interact with many different types of objects.

One approach to mitigate these issues is to use a non-learned geometry-based method [15], [16]. These methods, however, do not typically capture visual texture well, and they rely on hard-coded, rather than learned, invariances, which limits the potential accuracy of the system (based on our experiments in Section IV-D). A few recent learning-based approaches have attempted to perform zero-shot object pose estimation [17], [18] but these methods require instance segmentation masks to be provided as input, which limits their use in a “zero-shot” system, as such masks are typically trained per-object.

We seek to remove these limitations by developing a novel learning-based method for zero-shot object pose estimation that can handle both textured and untextured objects in cluttered scenes and does not require object masks as input. Our method uses the paradigm of pose hypothesis generation and evaluation: given a scene, a large number of candidate poses consistent with the observation are generated. The fitness of each hypothesis is then evaluated and the best-fit candidate is selected. Such an approach requires the

\* indicates equal contribution.

B. Okorn, Q. Gu, M. Hebert and D. Held are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213. (bokorn@andrew.cmu.edu, qiaog@andrew.cmu.edu, hebert@cs.cmu.edu, dheld@andrew.cmu.edu)

hypothesis rating function to give appropriate weight to the features that most correlate with the correct pose. The variation between sensor data and the object model, caused by sensor noise or lighting changes, as well as partial occlusions, can make designing this scoring function challenging. Past approaches to hypothesis scoring have used voting over hypotheses or feature matching [19], [20], [15]; in contrast, this paper proposes a scoring function that learns to compare the observed images and rendered model points. Our learned scoring function demonstrates a significant improvement on zero-shot object pose estimation over a wide set of objects and environmental variations.

The key insight of our method is to use a learned scoring function that compares the sensor observation to a sparse rendering of each candidate pose hypothesis. This scoring function receives as input *an unordered set of point differences*, shown in Fig. 2, which we show is crucial to perform zero-shot generalization to novel objects not seen in the training set. Our method is trained over a disparate set of objects and then evaluated on novel objects not included in the training set.

We demonstrate that our Zero-shot Pose Hypothesis Rating method (ZePhyR) works on objects in clutter without requiring object masks as input, unlike past zero-shot methods [18], [17]. ZePhyR handles both untextured objects as well as objects with significant visual texture, not seen at training time. Therefore, ZePhyR achieves the goal of zero-shot object pose estimation mentioned earlier:

- We require no new human annotations or large-scale synthetic data generation to interact with novel objects.
- We require no retraining for novel objects.
- ZePhyR uses only a single set of network weights, rather than requiring new weights for each unique object, reducing the memory constraints.

We evaluate our method on YCB-Video and LineMOD-Occlusion, two challenging pose estimation datasets. Our method achieves state-of-the-art results over previous zero-shot pose estimation methods.

## II. RELATED WORK

### A. Non-learned Zero-shot Pose Estimation

Zero-shot pose estimation is the task of estimating the pose of objects not seen at training time. Non-learning based approaches [21], [22], [23], [24], [25], [26], [27], [28], [29] are inherently zero-shot, leveraging robust features and the available object model at test time. Point Pair Features (PPF) [16], [15], [30], [31], [30], [32] use pairs of oriented points to generate geometrically consistent pose hypotheses and select the best hypothesis using voting and clustering. These are the top-performing zero-shot methods on the BOP leader board [33], when averaged over all datasets, but struggle to compete with deep learned methods on the highly textured YCB dataset due to the methods being exclusively based on depth.

### B. Learned Zero-shot Object Pose Estimation

Several learned methods solve the zero-shot pose estimation problem using class-based pose estimation [34], [35] as

opposed to instance-based pose estimation. These methods learn a pose estimator capable of generalizing among objects in the given class, but such methods are not intended to generalize to novel classes. While this is a step in the direction of zero-shot pose estimation, it still requires training a new network for each class.

Pose refinement methods like DeepIM [36] learn to estimate the residual pose between the observed data and a rendered viewpoint and have shown to generalize well to unseen classes of objects. These methods, however, require the initial rendered pose to be relatively close to the observation to produce accurate results, and as such is primarily used to refine a coarse pose prediction. Our method requires no such close initialization.

A few recent zero-shot methods use a learned representation of the object in their pose estimation pipeline [37], [18], [17]. While these methods have been shown to generalize across objects, they require a bounding box for the target object, which is obtained using an object-specific learned detector (and hence not a zero-shot system) or the ground-truth bounding box. This requirement is avoided in the MOPED dataset [17], as there is only a single object in the scene, which greatly simplifies the task of estimating the object mask [38]. For the LineMOD-Occlusion dataset, ground truth object masks are used [17]. Our method does not require such bounding boxes or masks as input, making it truly zero-shot.

### C. Pose Scoring

There has been some study of learned fitness functions. Differentiable RANSAC (DSAC) [20] explores learning a fully differentiable RANSAC algorithm. Specifically, they study the use of a REINFORCE style loss for scoring candidate hypotheses. We take inspiration from this work; however, their method focuses on a different task of camera localization rather than object pose estimation; as a result, many important details of our method, such as the input featurization and network architecture, are significantly different from their approach. Pose Proposal Critic [39] learns to regress to the reprojection error between a rendered pose and the observation. They numerically differentiate this error function as a means of pose refinement. However, they only evaluate this approach as a pose refinement technique, with a close initial pose estimate; in contrast, our focus is on evaluating a large set of pose hypotheses that span the entire observation space.

## III. METHOD

### A. Overview

The primary objective of this work is zero-shot object pose estimation in clutter. To achieve this, we train our pose estimation method on one set of objects and then evaluate on a set of novel objects, without requiring any re-training. This differentiates our work from previous work that requires real or synthetic training data of the test objects [7], [8]. Our work additionally differentiates from other zero-shot pose estimation work [17], [18], [16], in that it operates well in cluttered scenes, requires no object masks as input, and

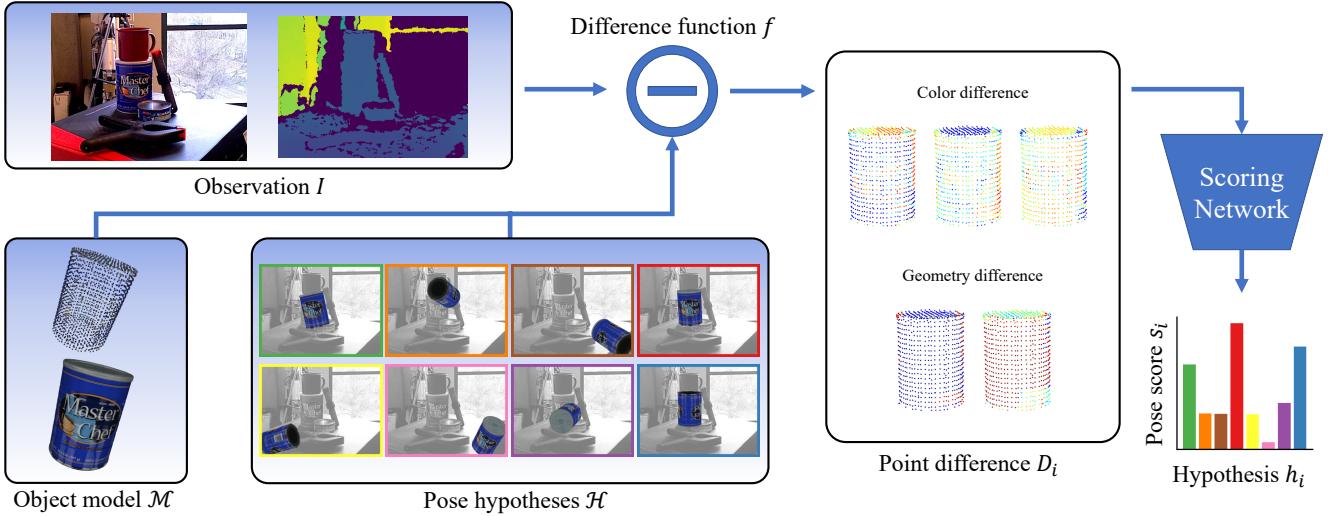


Fig. 2: System Pipeline. Our method first projects the sampled model points  $\mathcal{M}$  onto the observation  $I$  according to a pose hypothesis  $h_i$ . Then  $D_i$  are extracted as the point-wise differences between the observation and the projected model points, describing the alignment of the pose hypothesis at each projected point. A network takes in  $D_i$  and regresses to a score  $s_i$  for each pose  $h_i$  which evaluates how well the pose matches the observation.

produces accurate poses for both textured and untextured objects.

An overview of our method is shown in Figure 2. Given a set of 6D pose hypotheses, we first project each hypothesis into the scene. Our method learns to score each hypothesis by comparing differences in the projected object model point cloud to the RGB-D observation. For each projected model point, we extract the color and geometry information from both the model and the observation and compute the local differences of the extracted information. This yields a set of *point-differences*, one for each projected model point. Each element in this set encodes the local alignment between the model and the observation with respect to color and geometry. We adopt a point-based network [40], [41] to analyze this unordered set of point-differences and regress to an overall score for each pose hypothesis. Focusing on differences as well as adopting a point-based neighborhood structure helps us avoid overfitting to object-specific properties from the training set and allows us to generalize to unseen objects at test time.

In this work, our primary focus is the learned scoring function and we use existing methods to generate our initial pose hypothesis set. While many algorithms could be used to generate these potential object pose hypotheses [20], [42], [43], we use a combination of Point Pair Features [15] and SIFT features [44].

### B. Learned Scoring Function

The main goal of our method is to score pose hypotheses by projecting them into the observed scene and learning to compare their local geometric and color differences. Suppose that we have a set of 6D pose hypotheses  $\mathcal{H} = \{h_i\}_{i=1}^m$  that we wish to evaluate. We represent the object as a point cloud  $\mathcal{M} = \{x_j\}_{j=1}^n$ , sampled from the provided object mesh model, or obtained from a 3D reconstruction pipeline. Each point contains both geometric (depth and normal) and

color information drawn from its local region on the object. Similarly the observation image  $I$  contains geometric and color values from the observation. To evaluate hypothesis  $h_i$ , we project each object point  $x_j$  onto the observation's image plane, using the known camera parameters. This projection gives a point at image coordinates  $y_{ij}$  with transformed point values  $\tilde{x}_{ij}$  (the point depth and normal vector are transformed; the color of the projected point is unchanged). For each pose hypothesis, the difference between the projected values,  $\tilde{x}_{ij}$ , and their corresponding image values,  $I(y_{ij})$ , is computed according to a simple distance function,  $d_{ij} = f(\tilde{x}_{ij}, I(y_{ij}))$  (see Section III-D.2 for details).

The set  $D_i = \{d_{ij}\}_{j=1}^m$  represents an unordered set of point differences for pose hypothesis  $h_i$ , each of which is associated with a given point  $x_j$  in the model and a location  $y_{ij}$  in the observation image. We train a deep neural network  $g_\theta(D_i)$  with parameters  $\theta$  to analyze this difference set and regress to a pose fitness score,  $s_i$ . While one might assume that a simple hand-designed function for  $g$  would be sufficient, in practice, however, occlusions, lighting differences and other confounding factors make such simple methods ineffective. Our learned function can intelligently combine point differences on multiple parts of the object to robustly estimate the most likely pose hypothesis.

### C. Loss Function

To train this hypothesis scoring function, we adopt the probabilistic selection loss proposed by DSAC [20], as it directly optimizes the expected pose error when hypotheses are sampled according to the predicted scores. For each pose hypotheses  $h_i$  with corresponding true pose error  $\epsilon_i$ , we compute the expected pose error of sampling according to the softmax distribution induced by  $s_i$ ,  $\mathcal{L} = \sum_{i=1}^m \text{softmax}(s_i) \epsilon_i$ .

In our experiment,  $\epsilon_i$  is defined as the log of the average point distance (ADD) for non-symmetric objects and its sym-

metric analog (ADD-S) for symmetric ones [7]. Empirically, we find that the log of this error better dampens the effects of outliers. More discussion can be found in Section III-D.2. At test time, the highest-scoring pose hypothesis is selected. The inference pipeline is described in Algorithm 1.

---

**Algorithm 1:** Hypothesis Scoring Pose Estimation

---

```

Compute initial pose hypothesis set  $\mathcal{H} = \{h_i\}_{i=1}^m$ ;
foreach  $h_i$  in  $\mathcal{H}$  do
    Project all model points according to  $h_i$  onto the
    image plane to get projected model points  $\tilde{x}_{ij}$  at
    projected image coordinates  $y_{ij}$ ;
    Get observation points  $I(y_{ij})$ ;
    Compute point differences  $d_i = f(\tilde{x}_{ij}, I(y_{ij}))$ ;
    Score point-differences  $s_i = g_\theta(\{d_{ij}\}_{j=1}^m)$ ;
end
Return hypothesis  $h_{i^*}$ , where  $i^* = \arg \max_i s_i$ ;

```

---

#### D. Implementation details

1) *Hypothesis Generation*: We generate the initial hypotheses set using the commercially available Point Pair Feature software, HALCON 20.05 Progress software [45], which implements the PPF algorithm described in Drost *et al.* [15]. For each observation, we use the top 100 pose hypotheses generated by PPF. For detecting objects with high visual texture (e.g. for all objects in YCB-V), we augment these hypotheses using Dense SIFT feature matching. We obtain pose hypotheses from these features by aligning the surface normals and SIFT orientations of pairs of matched SIFT features; aligning the SIFT orientations and normals enables a single pair of matched SIFT features to define a 6D pose hypothesis.

2) *Network Input*: As input to the hypothesis scoring function, we use very simple geometric and color information for both the model and observation data. For each point on the model, we compute its 3D location, surface normal, and color in HSV space. When projecting each point into the observation frame, we transform both the normals and 3D coordinates to compute the depth and normal with respect to the camera. The color data is unaffected by the projection. Similarly, we compute local surface normals from the observation, and thus we obtain depth, normal and HSV color information at each pixel of the observation image.

To create the network inputs  $d_{ij}$ , we compute the signed difference between the projected and observed points for both depth and color. For surface normals, we use the cosine of the angle difference between the projected and observed normals. Additionally, we concatenate the projected image coordinates,  $y_{ij}$ , of the associated image point, normalized to zero mean and unit variance, as an additional input, to provide the structural neighborhood information.

3) *Network Structure*: Our network takes in the set of point-differences  $D_i = \{d_{ij}\}_{j=1}^m$  and outputs a single score,  $s_i$ , that estimates how well the pose hypothesis  $h_i$  matches the observation. Because  $D_i$  is an unordered set of point-differences, we use a network architecture designed to

handle unordered sets of points; specifically, we use PointNet++ [41]. Our experiments show that the loose neighborhood structure of this architecture enables zero-shot generalization to unseen objects. To define the spatial neighborhood for grouping points in PointNet++'s point set abstraction layers, we use the normalized image coordinates. We explore the effect of networks with different neighborhood structures in Section IV-F. See the supplementary material for more experimental details and hyperparameters.

## IV. EXPERIMENTS

### A. Datasets

We evaluated our method on two of the most popular datasets in the BOP Challenge [33], the YCB-Video (YCB-V) dataset [7] and the LineMOD-Occlusion (LM-O) dataset [9]. In these experiments, we follow the evaluation protocol set up by the BOP Challenge, with the additional constraint that our method is not trained on the objects it is tested on. This allows us to test our ability to perform zero-shot generalization to novel objects.

**YCB-Video dataset** (YCB-V) [7] contains 92 RGB-D video sequences of 21 YCB objects [48] of varying shape and texture, annotated with 6D poses. This a particularly challenging dataset for object pose estimation due to its varying lighting conditions, occlusions, and sensor noise. We follow the dataset split in [7], and for the evaluation, we adopt the BOP testing set [33], where 75 images with higher-quality ground-truth poses from each of 12 test videos are used. To demonstrate the generalization ability of our method, one half of the objects are used for training, and the other half are used for testing. To accommodate the full dataset, a second network is trained with train and test objects exchanged, such that each network only sees half of the objects during training, and no network is trained on the objects that it will be tested on. Note that we train our network on the training (seen) objects in the YCB-V training split and test on the testing (unseen) objects in the testing split, so not a single test image or object is seen during training. When evaluating on YCB-V, we use hypotheses generated from both PPF and SIFT matching to handle the high degree of visual texture. We also adopt a ICP refinement step [21] for post-processing.

**LineMOD-Occlusion dataset** (LM-O) [9] adopted a single scene from the test set of the larger LineMOD (LM) dataset [24] and provides ground-truth 6D pose annotations for 8 low-textured objects. For training, we used the PBR-BlenderProc4BOP [49] training images provided by the BOP challenge. This dataset contains photo-realistic synthetic images of LM objects dropped onto a table, with randomized background texture and object materials. Our model is only trained on synthetic images of the 7 objects that are in the LM dataset but *not* in the LM-O dataset; we then evaluate on the LM-O objects, which were not seen at training time. When evaluating on LM-O, we only use hypotheses generated by PPF; we find that SIFT hypotheses are ineffective on this dataset since the objects do not contain much visual texture.

	Zero-Shot Methods				Object Specific Methods	
	Drost [15]	Vidal [16]	Multipath [18]	ZePhyR + Drost (Ours)	CosyPose [46]	Pix2Pose [47]
YCB-V	0.344	0.450	0.289	<b>0.516</b>	0.861	0.675
LM-O	0.527	0.581	0.217	<b>0.598</b>	0.714	0.588

TABLE I: AR scores for methods of zero-shot and object specific pose estimation on object pose datasets (higher is better).

### B. Metrics

As suggested by the BOP challenge, we report the average recall (AR) scores as the average of the following three average-recall pose error metrics: Visible Surface Discrepancy (VSD), Maximum Symmetry-Aware Surface Distance (MSSD), and Maximum Symmetry-Aware Projection Distance (MSPD). For a detailed formulation of each metric, please refer to the supplementary material and [33].

### C. Baselines

We compare our method to both zero-shot and object-specific methods. As we are most concerned with our performance as compared to other zero-shot methods, we compare to two variants of Point Pair Features, Drost [15] and Vidal [16]. An implementation of Drost’s PPF [45] is used as the hypothesis generation algorithm in our work. Vidal had until recently been the top-performing method in the BOP challenge, and demonstrates the peak performance of PPF-only systems (although their code is not available). Other recent papers have proposed learning-based methods for zero-shot pose estimation, namely Multipath Augmented Autoencoders [18], which we compare against. While this method has been shown to generalize to unseen objects, the reported results that we include are a product of training a single model on the test objects; further, their method utilizes an object-specific detection network (also trained on the test objects) [50]. In addition to the zero-shot baselines, we report the current state of the art in object-specific methods as CosyPose [46] and Pix2Pose [47]. Both of these methods train a network on annotated instances of the test objects and have weights specifically associated with each object. While we are not attempting to match the performance of these systems, we report their results to illustrate the still remaining gap between zero-shot and object-specific methods.

### D. Zero-shot Pose Estimation Results

In Table I, we find that our method outperforms all zero-shot methods, significantly improving over our initial pose hypotheses produced by Drost and outperforming the best PPF-only solution in Vidal [16]. We see the largest improvement on the YCB dataset, where PPF is unable to fully resolve the pose of the geometrically symmetric but textually asymmetric objects, seen in failure to match the cylindrical objects in Figure 3. Our method is able to leverage both color and geometry, selecting the most accurate pose hypothesis. Additionally, we find our method to be comparable to the object-specific results produced by Pix2Pose [47]. While DeepIM [36] is a local refinement method, and not directly comparable to ZePhyR, we do evaluate its performance based on PPF in the supplementary material.

### E. Evaluating Generalization

As we stated previously, in order to ensure our network is not trained on the test objects we split the objects in YCB-V into two halves, training a network on each set of objects. We select via index parity, as it separates the dataset into splits with roughly equal numbers of symmetric and asymmetric objects, with “Object Set 1” and “Object Set 2” representing the set of objects with even and odd object IDs respectively. To evaluate how well our network generalizes, we compare our results on unseen objects to the objects each network was trained on. The full breakdown of each network’s scores are shown in Table II. Although there is some performance drop on unseen objects, the gap is relatively small, showing the generalization abilities of our method. The “Zero-Shot” column of shows the zero-shot performance of each model on the objects it does *not* see during training.

Tested on	Our method trained on		Zero-Shot
	Set 1	Set 2	
Object Set 1	0.624	0.543	0.543
Object Set 2	0.488	0.496	0.488
All Object	0.557	0.520	<b>0.516</b>

TABLE II: AR scores on YCB-V object subsets.

### F. Neighborhood Structure

We explore the effects of different neighborhood structures on the accuracy and generalization of our method. Our method uses a PointNet++ [41] architecture that uses a hierarchical neighborhood structure; we compare this to a CNN architecture that uses a strict neighborhood structure and a PointNet-based architecture [40] that uses a global structure. For the CNN approach, we generate a sparse difference image using the projected point differences before passing it to a ResNet18 network [51]. Our PointNet++ approach uses normalized image coordinates for neighborhood grouping. The PointNet approach contains the normalized image coordinates but it does not perform explicit neighborhood grouping. In Table III, we see that the loose local neighborhood structure found in PointNet++ outperforms the global structure of PointNet as well as the strict structure used in image convolutions. This implies that some neighborhood structure is important for evaluating these sparse point differences, but a too strict neighborhood hampers both performance and generalization.

On YCB-V dataset	PointNet++ (Hierarchical)	PointNet (Global)	CNN (Strict)
Seen (Training) Objects	<b>0.624</b>	0.477	0.533
Unseen (Test) Objects	<b>0.488</b>	0.355	0.386
Total	<b>0.557</b>	0.416	0.459

TABLE III: Comparison of the performance of the different neighborhood structure through network architectures.



Input Image



Original PPF Results



Our Improved Results

Fig. 3: Qualitative results on image from YCB-V dataset showing the improved accuracy of our method.

### G. Input Ablations

To determine the relative importance of each of our input channels, we retrain our networks without each dimension. We show results on YCB in Table IV, training on the “Object Set 1” and testing on “Object Set 2”. Additionally, this table shows the effects of concatenating observation and model inputs (“Model without Diff”), as opposed to computing their difference (as in our method). As can be seen, using concatenation instead of differencing gives little change in performance for seen objects, whereas it gives worse performance for unseen objects. Unsurprisingly, the color information has the greatest effect on the accuracy of our system, as it is the most orthogonal to the information used by our PPF hypotheses.

	Model without				
	Color	Depth	Normal	Coords	Diff
Unseen Objects (Zero-shot)	-18%	-15%	-7.1%	-8.9%	-6.3%
Seen Objects (Training)	-24%	-4.2%	0.8%	1.1%	2.1%

TABLE IV: Percent change in AR scores on YCB Video dataset caused by removal of each input to our method.

### H. Timing analysis

We analyze the inference speed of our method in Table V. We separate our method into 5 stages, including generating pose hypotheses from SIFT feature matching (“SIFT”), generating pose hypotheses from PPF (“PPF”), computing, transforming and comparing the observation and model values for all hypotheses (“Projection”) and inference with our scoring network (“Scoring”). Note that we only use 100 PPF hypotheses for LM-O, whereas we use additional 1000 SIFT hypotheses for YCB-V. We found that the LM-O dataset required more accurate initial pose hypotheses, requiring significantly more processing time. To compensate for this, we evaluate the time-performance trade-off of different sets of PPF parameters on the LM-O dataset, shown in blue on Figure 4. Since the LM-O dataset is challenging due to strong occlusions and limited scales of objects in the scene, PPF methods [15], [16] need a high sampling rate to produce reasonable pose estimates. Therefore, increased speed comes at the cost of performance, but our method consistently improves the accuracy of the initial hypotheses, shown in red, at all stages of the curve.

	SIFT	PPF	Projection	Scoring	Total
YCB-V	0.142	0.291	0.051	0.135	0.619
LM-O	0	2.900	0.014	0.034	2.949

TABLE V: Test time spent (sec) in each stage of our pipeline.

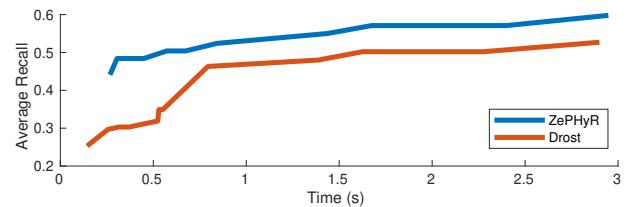


Fig. 4: Speed accuracy analysis of our method (blue) over various PPF hypothesis generation hyperparameters on LM-O. Base PPF accuracy shown in orange.

### I. Reconstructed Model Results

To show the effectiveness of our method in robotic scenarios, we test our pipeline on newly generated object model reconstructions. Using fiducial markers [52] and TSDF based surface reconstruction [53], we build textured mesh model of a novel drill object. As shown in Figure 1, we are able to estimate the pose of the target object while in human hands and while being manipulated by the robot; because our method is zero-shot, we do not require any retraining to estimate the pose of new objects such as this one. We find that these poses, coupled with annotated grasp locations, allow the robot to perform task specific grasps. See our video for visualizations of the predicted poses as well as demonstration of our grasping pipeline.

## V. CONCLUSION

We propose a method for zero-shot object pose estimation, focusing on pose hypothesis scoring. By extracting point differences between the projected object points and the observation and imposing a loose neighborhood structure on these points, we learn a pose scoring function that generalizes well to novel objects. On the challenging YCB-Video and LineMOD-Occlusion datasets, our method achieves state-of-the-art performance for zero-shot object pose estimation in clutter, evaluated on both textured and untextured objects. We hope that our method paves the way for roboticists to obtain accurate pose estimates for novel objects without needing additional training or data annotation.

## VI. ACKNOWLEDGEMENTS

This work was supported by NASA NSTRF, United States Air Force and DARPA under Contract No. FA8750-18-C-0092, National Science Foundation under Grant No. IIS-1849154, and LG Electronics.

## REFERENCES

- [1] N. T. Dantam, Z. K. Kingston, S. Chaudhuri, and L. E. Kavraki, “Incremental task and motion planning: A constraint-based approach,” in *RSS*, 2016.
- [2] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, “Chomp: Covariant hamiltonian optimization for motion planning,” *IJRR*, 2013.
- [3] S.-K. Kim and M. Likhachev, “Planning for grasp selection of partially occluded objects,” in *ICRA*, 2016.
- [4] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, “Learning robotic assembly from cad,” in *ICRA*, 2018.
- [5] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, “The columbia grasp database,” in *ICRA*, 2009.
- [6] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Sucan, “Towards reliable grasping and manipulation in household environments,” in *Experimental Robotics*, 2014.
- [7] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *RSS*, 2018.
- [8] 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 *CVPR*, 2019, pp. 3343–3352.
- [9] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, “Learning 6d object pose estimation using 3d object coordinates,” in *ECCV*, Cham, 2014, pp. 536–551.
- [10] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, “Deep object pose estimation for semantic robotic grasping of household objects,” *arXiv preprint arXiv:1809.10790*, 2018.
- [11] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, “Poserbpf: A rao-blackwellized particle filter for 6d object pose estimation,” in *RSS*, 2019.
- [12] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, “Augmented autoencoders: Implicit 3d orientation learning for 6d object detection,” *IJCV*, vol. 128, no. 3, pp. 714–729, 2020.
- [13] J. Tremblay, T. To, and S. Birchfield, “Falling things: A synthetic dataset for 3d object detection and pose estimation,” in *CVPRW*, 2018, pp. 2038–2041.
- [14] D. Dwibedi, I. Misra, and M. Hebert, “Cut, paste and learn: Surprisingly easy synthesis for instance detection,” in *ICCV*, 2017, pp. 1301–1310.
- [15] B. Drost, M. Ulrich, N. Navab, and S. Ilic, “Model globally, match locally: Efficient and robust 3d object recognition,” in *CVPR*, 2010, pp. 998–1005.
- [16] J. Vidal, C.-Y. Lin, X. Lladó, and R. Martí, “A method for 6d pose estimation of free-form rigid objects using point pair features on range data,” *Sensors*, vol. 18, no. 8, p. 2678, 2018.
- [17] K. Park, A. Mousavian, Y. Xiang, and D. Fox, “Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation,” in *CVPR*, 2020.
- [18] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel, “Multi-path learning for object pose estimation across domains,” in *CVPR*, June 2020.
- [19] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [20] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, “Dsac-differentiable ransac for camera localization,” in *CVPR*, 2017, pp. 6684–6692.
- [21] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *TPAMI*, vol. 14, no. 2, pp. 239–256, 1992.
- [22] M. Ulrich, C. Wiedemann, and C. Steger, “Combining scale-space and similarity-based aspect graphs for fast 3d object recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 10, pp. 1902–1914, 2012.
- [23] 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, 2012.
- [24] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, “Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes,” in *ACCV*, Berlin, Heidelberg, 2013, pp. 548–562.
- [25] E. Muñoz, Y. Konishi, V. Murino, and A. Del Bue, “Fast 6d pose estimation for texture-less objects from a single rgb image,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5623–5630.
- [26] J. J. Lim, A. Khosla, and A. Torralba, “Fpm: Fine pose parts-based model with 3d cad models,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 478–493.
- [27] E. Muñoz, Y. Konishi, C. Beltran, V. Murino, and A. Del Bue, “Fast 6d pose from a single rgb image using cascaded forests templates,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4062–4069.
- [28] Z. Guo, Z. Chai, C. Liu, and Z. Xiong, “A fast global method combined with local features for 6d object pose estimation,” in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2019, pp. 1–6.
- [29] H. Yu, Q. Fu, Z. Yang, L. Tan, W. Sun, and M. Sun, “Robust robot pose estimation for challenging scenes with an rgb-d camera,” *IEEE Sensors Journal*, vol. 19, no. 6, pp. 2217–2229, 2019.
- [30] B. Drost and S. Ilic, “3d object detection and localization using multimodal point pair features,” in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE, 2012, pp. 9–16.
- [31] E. Kim and G. Medioni, “3d object recognition in range images using visibility context,” in *IROS*, 2011, pp. 3800–3807.
- [32] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige, “Going further with point pair features,” in *ECCV*, 2016, pp. 834–848.
- [33] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, “BOP: Benchmark for 6D object pose estimation,” *ECCV*, 2018.
- [34] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, “kpam: Keypoint affordances for category-level robotic manipulation,” *arXiv preprint arXiv:1903.06684*, 2019.
- [35] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, “Normalized object coordinate space for category-level 6d object pose and size estimation,” in *CVPR*, 2019, pp. 2642–2651.
- [36] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, “Deepim: Deep iterative matching for 6d pose estimation,” in *ECCV*, 2018.
- [37] Y. Xiao, X. Qiu, P.-A. Langlois, M. Aubry, and R. Marlet, “Pose from shape: Deep pose estimation for arbitrary 3d objects,” *arXiv preprint arXiv:1906.05105*, 2019.
- [38] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, “The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation,” in *CoRL*, 2020, pp. 1369–1378.
- [39] L. Brynte and F. Kahl, “Pose proposal critic: Robust pose refinement by learning reprojection errors,” *arXiv preprint arXiv:2005.06262*, 2020.
- [40] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *CVPR*, July 2017.
- [41] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in *NeurIPS*, 2017, pp. 5099–5108.
- [42] V. Narayanan and M. Likhachev, “Perch: Perception via search for multi-object recognition and localization,” in *ICRA*, 2016, pp. 5052–5059.
- [43] A. Collet, M. Martinez, and S. S. Srinivasa, “The moped framework: Object recognition and pose estimation for manipulation,” *IJRR*, vol. 30, no. 10, pp. 1284–1306, 2011.
- [44] D. G. Lowe, “Object recognition from local scale-invariant features,” in *ICCV*, vol. 2. Ieee, 1999, pp. 1150–1157.
- [45] MVtec Software GmbH, “Halcon.” [Online]. Available: <https://www.mvtc.com/products/halcon/documentation/release-notes-1911-0/>
- [46] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, “Cosopose: Consistent multi-view multi-object 6d pose estimation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

- [47] K. Park, T. Patten, and M. Vincze, “Pix2pose: Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.
- [48] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The ycb object and model set: Towards common benchmarks for manipulation research,” in *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.
- [49] T. Hodaň, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. Sinha, and B. Guenter, “Photorealistic image synthesis for object instance detection,” *ICIP*, 2019.
- [50] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *ICCV*, 2017, pp. 2961–2969.
- [51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016, pp. 770–778.
- [52] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.
- [53] Q.-Y. Zhou and V. Koltun, “Dense scene reconstruction with points of interest,” *ACM Transactions on Graphics (ToG)*, vol. 32, no. 4, pp. 1–8, 2013.
- [54] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
- [55] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [56] M. S. GmbH, *find\_surface\_model [HALCON Operator Reference / Version 13.0.4]*, 2019 (accessed Nov 2nd, 2020), [https://www.mvtec.com/doc/halcon/13/en/find\\_surface\\_model.html](https://www.mvtec.com/doc/halcon/13/en/find_surface_model.html).

## APPENDIX

### A. Pose Error Metrics

For the evaluation of our method, we adopt three metrics proposed by the BOP challenge 2019 [33]. Given an object model  $\mathcal{M}$ , an estimated pose  $\hat{\mathbf{P}}$  and its corresponding ground truth  $\mathbf{P}$ , we calculate three metrics as follows.

1) *Visible Surface Discrepancy (VSD)*: Given the estimated and ground truth pose  $\hat{\mathbf{P}}$  and  $\mathbf{P}$ , the object model is rendered to obtain the estimated and ground truth distance maps  $\hat{D}$  and  $D$  respectively. The distance maps are then compared with the observed distance map to obtain the visibility masks  $\hat{V}$  and  $V$ , which are the sets of pixels where the object is visible in the test image. Then the VSD measures the discrepancy of the estimated and ground truth distance maps that are visible as follows.

$$e_{\text{VSD}} = \operatorname{avg}_{p \in \hat{V} \cup V} \begin{cases} 0 & \text{if } p \in \hat{V} \cap V \wedge |\hat{D}(p) - D(p)| < \tau \\ 1 & \text{otherwise,} \end{cases} \quad (1)$$

where  $p \in \hat{V} \cap V$  iterates over all pixels that is both visible under  $\hat{\mathbf{P}}$  and  $\mathbf{P}$ . Note here VSD only measures geometry alignment (color agnostic) and treats indistinguishable poses as equivalent by considering only the visible object part.

2) *Maximum Symmetry-Aware Surface Distance (MSSD)*: Consider a object point cloud  $\mathcal{M} = \{x_j\}$  and a set of symmetric transformations  $\mathcal{T}$  for this object, MSSD is defined as

$$e_{\text{MSSD}} = \min_{T \in \mathcal{T}} \max_{x_j \in \mathcal{M}} \|\hat{\mathbf{P}}x_j - \mathbf{P}Tx_j\|_2. \quad (2)$$

MSSD measures the surface deviation in 3D, and thus is relevant for robotics applications.

3) *Maximum Symmetry-Aware Projection Distance (MSPD)*: Let  $\text{proj}$  denote the 2D projection operations. Then MSPD is defined as

$$e_{\text{MSPD}} = \min_{T \in \mathcal{T}} \max_{x_j \in \mathcal{M}} \|\text{proj}(\hat{\mathbf{P}}x_j) - \text{proj}(\mathbf{P}Tx_j)\|_2. \quad (3)$$

Therefore MSPD measures the maximum perceivable discrepancy in 2D image space.

Based on the above three metrics, a overall Average Recall (AR) score is computed. Given an estimated pose, it is considered correct if  $e < \theta_e$  w.r.t pose error metric  $e$ , where  $e \in \{e_{\text{VSD}}, e_{\text{MSSD}}, e_{\text{MSPD}}\}$  and  $\theta_e$  is the threshold of correctness. The ratio of correctly-estimated poses over all pose estimation targets, is defined as recall. Then  $\text{AR}_e$  is the Average Recall w.r.t. metric  $e$ , which can be calculated for multiple thresholds  $\theta_e$  and multiple misalignment tolerance  $\tau$  in the case of  $e_{\text{VSD}}$ . The final AR score is computed as the average of three:

$$\text{AR} = (\text{AR}_{\text{VSD}} + \text{AR}_{\text{MSSD}} + \text{AR}_{\text{MSPD}})/3. \quad (4)$$

### B. Ground Truth Translation Results

The Multipath Augmented Autoencoders [18] baseline assumes that the object is cropped from the scene prior to input. In contrast, the focus of ZePhyR is to perform zero-shot pose estimation in cluttered scenes which contain multiple objects. In such cluttered scenes, finding the correct object crop of a novel object is non-trivial.

In BOP leaderboard<sup>1</sup>, Multipath Autoencoders [18] reports their performance with the assistance of a dataset-wise trained MaskRCNNs as a segmentation network. Consider that the main contribution of [18] is learning rotation encoding that generalizes over objects, we resolve the scale ambiguity and isolate the orientation error by providing this network with the ground truth translation for each object at test time. As shown in Table VI, our method still outperforms [18], especially on the YCB-V dataset where most objects have rotational symmetry.

Method	Multipath AutoEncoder		Ours
	w/o GT trans	w/ GT trans	w/o GT trans
YCB-V	0.289	0.355	0.516
LM-O	0.217	0.560	0.598

TABLE VI: AR scores for different method with and without ground truth translation (“GT trans”).

### C. Pose Hypothesis Ablations Results

We test our scoring method on different subsets of pose hypotheses to explore our sensitivity to the hypothesis generation method. In Table. VII, we report the AR scores of the Point Pair Features baseline (“PPF”) [15], our scoring method using pose hypotheses generated only from PPF (“PPF+Scoring”), our scoring method using pose hypotheses generated only from SIFT feature matching (“SIFT+Scoring”) and our scoring method using pose hypotheses generated from both PPF and SIFT (“Both+Scoring”). The results indicates that on the YCB-V dataset, where most objects have high-quality mesh models and rich textures, the SIFT feature matching method provides valuable pose hypotheses. When combining PPF and SIFT hypotheses with our scoring method, the results improve over using our scoring method with PPF hypotheses alone. LineMOD (LM-O), however, contains mostly low texture or textureless objects. For this dataset, SIFT hypotheses are less useful and adding them mildly reduces the accuracy of our method but needs more processing time.

Method	PPF	PPF+ZePhyR	SIFT+ZePhyR	Both+ZePhyR
YCB-V	0.344	0.458	0.390	0.516
LM-O	0.527	0.598	0.011	0.595

TABLE VII: BOP AR scores for ZePhyR based on different hypothesis generation methods.

<sup>1</sup>[https://bop.felk.cvut.cz/method\\_info/96/](https://bop.felk.cvut.cz/method_info/96/)

#### D. Network Details

1) *PointNet++*: As mentioned in Section III-D.3, we reduce the sizes of MLP and adjust parameters of original PointNet++ design, to enable the training of the whole network with 1100 pose hypotheses in 11 GB GPU memory. We uniformly downsample the object mesh models so that the leaf size for the voxel grid is 7 millimeter and each object has 1000 points on average, and further randomly subsampled the input points down to 2000 when the number of points in the downsampled object model still exceeds this number. The detailed network architecture is described as follows.

We use the single scale grouping (SSG) version of PointNet++. Following architecture protocol in [41], we denote  $SA(K, r, [l_1, \dots, l_d])$  as a set abstraction (SA) level with  $K$  local regions of ball radius  $r$  using PointNet of  $d$  fully connected layers with width  $l_i$  ( $i = 1, \dots, d$ ).  $SA([l_1, \dots, l_d])$  represents a global set abstraction level that converts set to a single vector.  $FC(l, dp)$  represents a fully connected layer with width  $l$  and dropout ratio  $dp$ . All fully connected layers are followed by batch normalization [54] and ReLU activation functions, except for the last score prediction layer. The resulting PointNet++ architecture is as follows:

$$\begin{aligned} SA(128, 0.2, [16, 32]) &\rightarrow SA(16, 0.5, [32, 64]) \rightarrow \\ SA([64, 128]) &\rightarrow FC(64, 0.4) \rightarrow FC(16, 0.4) \rightarrow FC(1) \end{aligned}$$

2) *PointNet*: For the ablation experiment on PointNet in Section V-C, we also use a reduced version of Classification Network described in [40]. We remove the input transform and feature transform layers. We use a three-layer MLP, with the size of the hidden layer to be 16, pre-bottleneck, a bottleneck max pooling layer of dimension 16, and a 3-layer MLP with the hidden layer size 64 post-bottleneck. All except the last MLP layers are followed by a batch normalization layer [54] and a ReLU activation. The final output of the last layer estimates a single score for each input point cloud.

3) *Convolutional Network*: For the CNN mentioned in Section V-C, we use a vanilla ResNet-18 [51] with no pretrained-weight. The the number of input channels of the first layer is expanded to match the number of error features, and the last layer is changed to a 2-layer MLP with the hidden layer size 64. The final output is a single score for each pose hypothesis.

#### E. Training Details

For computational efficiency, we subsample the training data points in the YCB-V and LM-O datasets and pre-process them for fast training. Specifically, from the YCB-V training split, we evenly sampled 4716 observations, containing 2346 observations of objects with even IDs and 2370 of objects with odd IDs. From the synthetic training set of LineMOD dataset [49], we evenly sampled 1749 observations of objects that are not in LM-O dataset as the training set. The observations of the training objects are then split, with 90% used for training and 10% used for validation. After training, the model weights at the epoch with lowest error on validation

set of the “seen” objects are selected for evaluation, and the observations of “unseen” objects are not used during training or validation.

To train the PointNet and PointNet++ architectures, we use an Adam optimizer [55] with an initial learning rate  $3 \times 10^{-4}$ . For the CNN training, the initial learning rate is  $1 \times 10^{-5}$ . We trained each network for 100 epochs and the learning rate reduces to 1/10 after epoch 30 and 80.

We augment the training data by randomly jittering the brightness, contrast, saturation and hue of the observation images by factor of 0.2, 0.2, 0.2 and 0.05 respectively. To prevent overfitting to the training objects, we also jointly perturb the color of the model and the observation color, changing the color of both the real and rendered data in the same way. The factors for brightness, contrast, saturation and hue in this process are all 0.5.

#### F. Comparison of DeepIM

Method	Drost [15]	Drost [15] + DeepIM [36]	Drost [15] + ZePhyR(ours)
YCB-V	0.344	0.324	0.516
LM-O	0.527	0.165	0.598

TABLE VIII: BOP AR scores for DeepIM taking pose hypothesis from PPF.

ZePhyR is a pose hypothesis scoring method, which is different from other learned pose refinement methods in the literature in the sense that ZePhyR can select the best one among multiple pose hypotheses over the entire search space while pose refinement only improves a single pose in a local region. To quantify this difference, we evaluate DeepIM [36] using the publicly available implementation<sup>2</sup> and the model checkpoint trained on the YCB-V dataset (model trained on LM-O is not provided). Note that for YCB-V, the DeepIM performance is not zero-shot as YCB objects are seen during training, while its performance on LM-O is zero-shot. The pose for DeepIM is initialized from the results of Drost *et al.* [15], and the results are shown in Table. VIII.

According to Drost *et al.* [15], ICP algorithm [21] is already used as a post-processing step in the PPF pipeline and thus their pose estimation results are already very accurate geometrically. Therefore, the room of improvement left for DeepIM is very limited as it only refines on a single pose hypothesis in the local region. We observe that for seen objects in the YCB-V dataset, DeepIM does not make obvious improvement based on Drost and even makes it slightly worse. And when DeepIM is tested on unseen objects (trained on YCB objects and tested on the LM-O dataset), it makes the initial estimation drastically worse.

#### G. Qualitative Results

Figure 5 shows the qualitative results of both our method and the baseline over the YCB-V and LM-O datasets. The left column shows the full scene; the second column shows the ground-truth pose for the target object. The third column

<sup>2</sup><https://github.com/NVlabs/DeepIM-PyTorch>

shows the highest-scoring pose according to our method, and the last column shows the highest-scoring pose according to the PPF baseline [15]. In the 3rd and 4th columns, the selected pose hypothesis for each method is rendered into the frame.

Overall, Our method demonstrates a better performance than the PPF baseline. As PPF only considers geometry, it cannot determine the correct orientation on some objects that are symmetrical in shape but have distinguishing texture, like the “Master Chef” can and tomato can in row (5), (7) and (8) in Figure 5. But our method considers both shape and color information, and thus can make correct estimations in such cases. PPF also tends to match the flat side of an object to the flat top of a table, such shown in row (3), (6), (7) and (9) in Figure 5; our method fixes such errors.

Figure 5 also shows some cases where our method fails. In row (8), due to the over exposure on the surface of the sugar box, our method mixes the back side of the box with the front side. In row (7), our method fails to detect the “Soft Scrub” bottle probably because only its side is facing towards the camera, where almost no texture or color information is present. The toy cat in row (3) and the egg box in row (2) are two failure cases where the occlusion is so strong that the whole object is almost invisible.

#### *H. Failure Case Analysis*

Figure 6 further elaborates the failure case of the sugar box in the row (8) of Figure 5. As we can see, due to the reflection, the upper surface of the sugar box in the observation is overly lightened, which makes the saturation and value errors of the wrongly-picked hypothesis smaller than those of the correct one. However, our method correct recover the geometry and still presents a reasonable result.

#### *I. Time-Accuracy Trade-off on LM-O dataset*

In Table IX, we report the detailed data for the time-accuracy trade-off curve in Figure 4 in the main paper. We here only vary the PPF parameters and thus its inference time. The speed of our scoring network (ZePhyR) is unchanged. In the table, “Model SD” and “Scene SD” are the sampling distance on the model point cloud and the scene point cloud respectively, relative to the model diameter. Higher numbers lead to smaller point clouds and faster processing times. “Ref Pt Rate” is the ratio of the points on the scene point cloud that are used as reference points when sampling point pairs [15]. “Dense Object PC” means the input object model to PPF is directly converted from the mesh model without downsampling. “Sparse Object PC” means PPF uses the downsampled object point cloud that is used in the scoring network, as described in Section VI-D.1. “Sparse” and “Dense” in “Refinement” column indicates the spacial density of the point cloud used for ICP step in PPF. We refer readers to [15] and [56] for more details.

Note that ZePhyR is a scoring network on the provided pose hypotheses, and in the table, our PPF+ZePhyR demonstrate a constant improvement over the PPF baseline by a large margin with only little time overhead. This means our

method is able robustly pick better hypothesis from the PPF’s output. Comparing the first and the third row in the table, we can find that PPF+ZePhyR achieves comparable results with PPF but is sped up by more than 3 times.

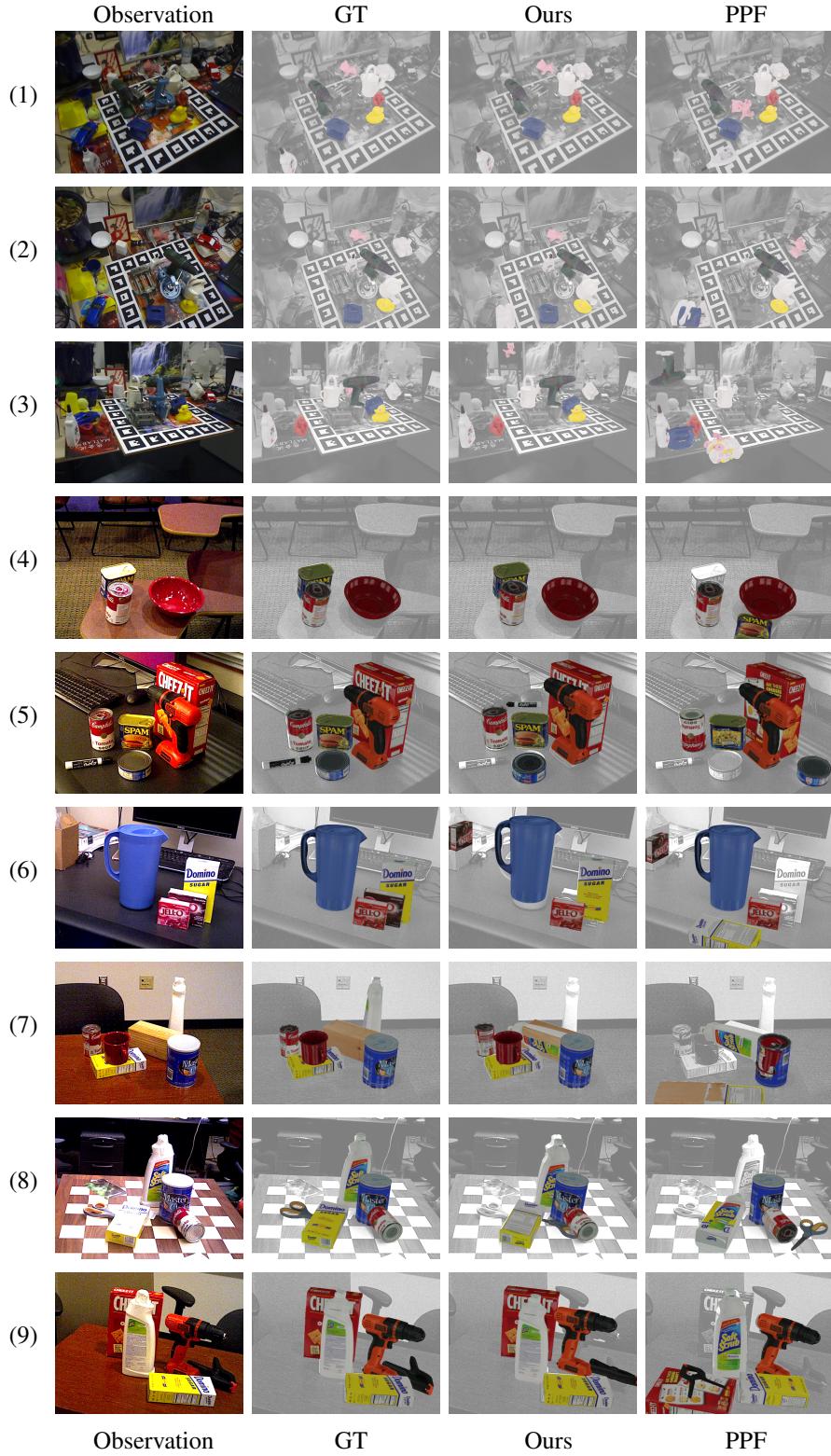
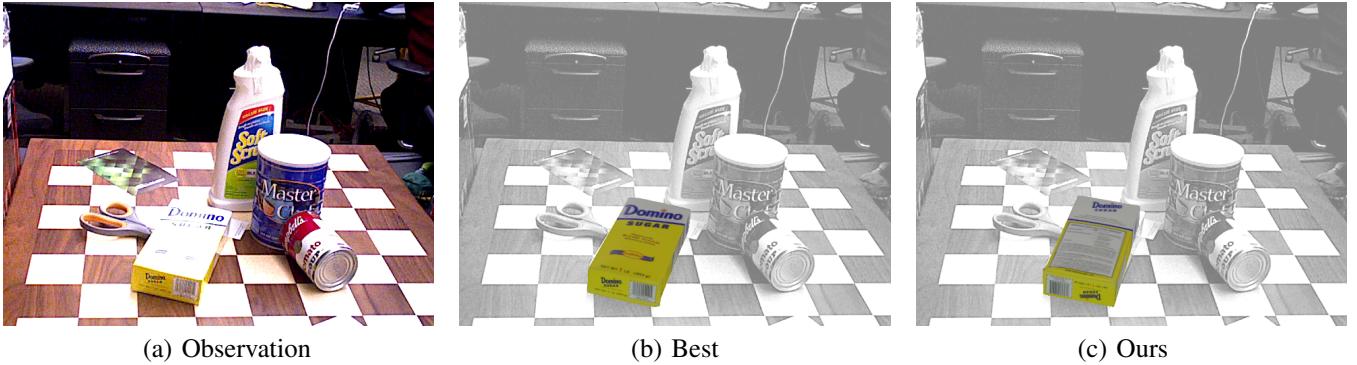


Fig. 5: Qualitative results on LM-O (first 3 rows) and YCB-V (last 6 rows) dataset. Raw input image and ground truth renders shown in the first and second column, respectively. The third and fourth column compare the top results using our scoring pipeline (“Ours”) and the original PPF (“PPF”) hypothesis algorithm [15], respectively.



(a) Observation

(b) Best

(c) Ours

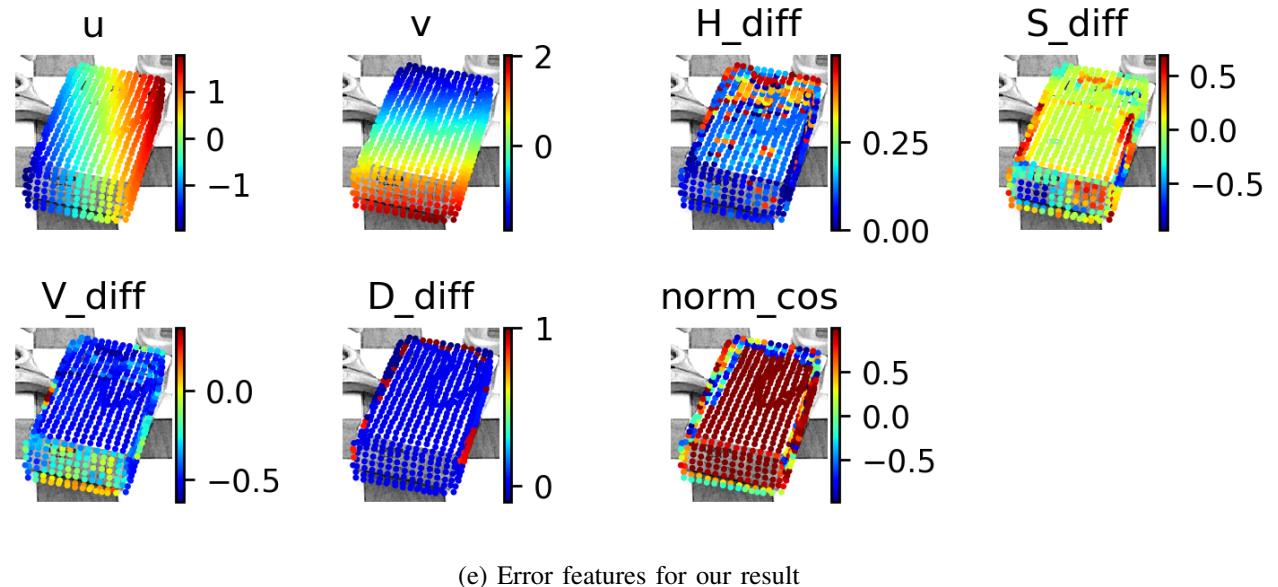
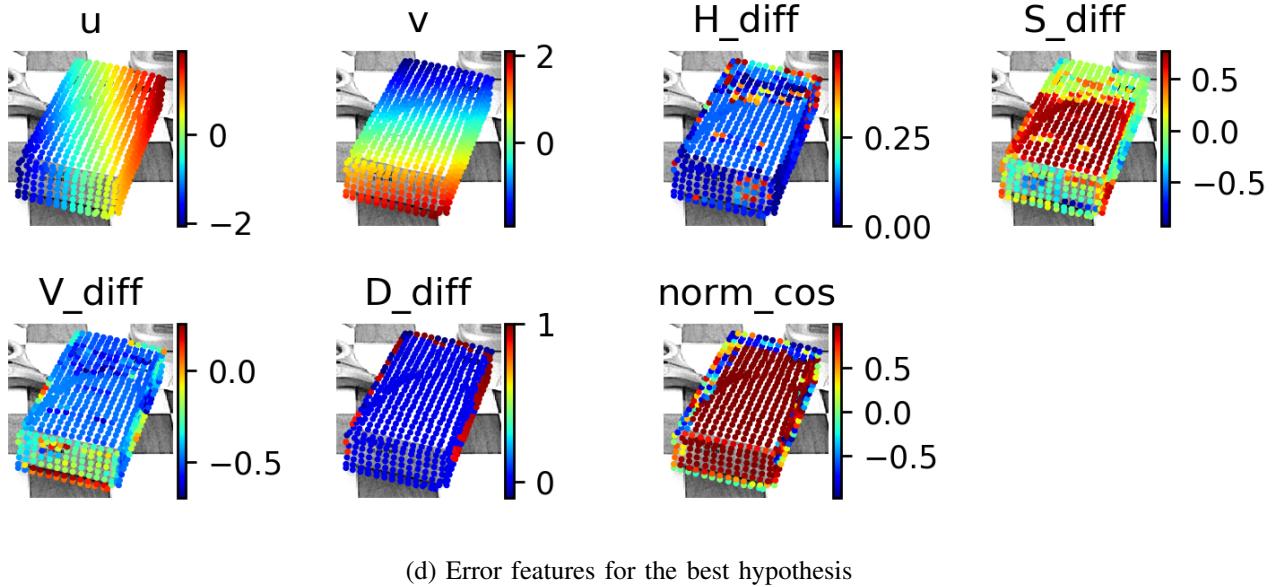


Fig. 6: Failures case of our method. “Best” means the pose that has the lowest ADD error in the pose hypothesis set. “Ours” means the highest scoring hypothesis returned by our method. In plot (d) and (e), “u” and “v” are the normalized projection coordinates. “H\_diff”, “S\_diff”, “V\_diff” and “D\_diff” represent the signed difference of the hue, value, saturation and depth between projected model points and the observation respectively. “norm\_cos” is the cosine of the angle between transformed model normal vectors and observed normal vectors.

Model SD	Scene SD	Ref Pt Rate	Object PC	Refinement	Time (PPF)	BOP score (PPF)	Time (PPF+ZePhyR)	BOP score (PPF+ZePhyR)
0.03	0.03	1	Dense	Dense	2.900	0.527	2.948	<b>0.598</b>
0.03	0.05	1	Dense	Sparse	1.626	0.502	1.674	0.571
0.05	0.05	1	Dense	Sparse	1.388	0.480	1.436	0.550
0.05	0.05	0.5	Dense	Sparse	0.794	0.463	0.842	0.524
0.05	0.07	0.5	Dense	Sparse	0.530	0.349	0.578	0.456
0.03	0.04	0.5	Sparse	Sparse	0.524	0.319	0.572	0.504
0.05	0.07	0.25	Dense	Sparse	0.315	0.303	0.363	0.408
0.03	0.04	0.2	Sparse	Sparse	0.257	0.297	0.305	0.484
0.03	0.05	0.2	Sparse	Sparse	0.219	0.253	0.267	0.441
0.05	0.05	0.2	Sparse	Sparse	0.200	0.213	0.248	0.379

TABLE IX: Inference time and performance on the LM-O dataset of PPF and PPF+ZePhyR using different PPF settings.