Category-agnostic Segmentation for Robotic Grasping

Anas Gouda, Abraham Ghanem, Christopher Reining

Abstract—Robotic grasping in unstructured and non-deterministic environments needs to depend on smart vision processing that is able to segment unseen and arbitrary number of object categories. The usual case of detecting and segmenting objects from training sets makes the work limited to its own setup. This encourages us to develop generic methods for category-agnostic segmentation.

In this work we introduce DoPose, a dataset of highly cluttered and closely stacked objects for segmentation and 6D pose estimation. We show how using careful choice of synthetic data and fine-tuning on our real dataset along with a rational training can boost the performance of already existing CNN architectures to generalize on real data and produce comparable results to SOTA methods even without post-processing or refinements. Our DoPose dataset, network models, pipeline code and ROS driver are available online.

Index Terms—category-agnostic segmentation, robotic grasping

I. INTRODUCTION

3D classical non-learning segmentation methods can reach a limit when dealing with cluttered data or uncertain environment. But when trying to switch from a classical non-learning approach to a learning approach, the first question that comes to mind is: Can the openly available pre-trained models even work for my data? This question encourages us to pursue deep learning method that can be as generic as non-learning ones.

A wider concept of category-agnostic segmentation of unseen objects is class-agnostic segmentation. The class-agnostic segmentation approach is introduced in [1] in the general context of segmentation with few-shot learning, in [2] for salient object detection, and in [3] for video segmentation.

[4], [5], [6] and [7] use the concept of category-agnostic segmentation for unseen objects in the context of robotic grasping. In that work, different input data representations are used (RGB, depth, RGB-D, 3D). Some of these methods also introduced post-processing methods for segmentation mask refinement. For most of that work, RGB data played a big role even for methods that used RGB-D or 3D methods. In our previous work [8], we show preliminary results for training with synthetic data and testing with real. Our previous method generalizes well on simple scenes but faces a huge drop in performance on highly cluttered scenes and problems with background over-segmentation.

In this contribution, we are revisiting the method of category-agnostic segmentation for unseen objects with more focus on the training data and the training process rather than the method. Our goal here is to train a model that is able to segment unseen objects on different scene setups (tabletop, bin, shelf). Figure 1 shows the final results of our model on unseen dataset with unseen objects from NVIDIA HOPE dataset [9].

The key points of this contribution consist of:

- A dataset of highly cluttered and closely stacked objects on two different scene types (tabletop, bin picking) with high resolution images. The dataset offers the largest available collection of high-resolution real images of bin scenes. This dataset is used in this work to enhance the performance of our models.
- An RGB and RGB-D model for category-agnostic unseen object segmentation. As shown later, our RGB model can generalize to different scene types and perform close to the SOTA methods that require 3D input data.
- A multi-step training approach with synthetic and real data that can generalize on real unseen datasets. We show that this method boosts the performance and reduces sim2real gap.
- ROS package for grasping based on our pipeline using our pre-trained models.
II. RELATED WORK

We discuss related work concerned with robotic grasping with deep learning, category-agnostic segmentation approaches, and relevant datasets.

Most grasping methods can be categorized into two categories. First, methods that separate the 3D image processing steps from the grasp computation. Second, methods that try to compute feasible grasps directly from depth images or point clouds. This idea was explained in [8] by splitting the grasping pipeline into generic building blocks.

[10] introduced a large data collection and training pipeline using deep reinforcement learning to merge the whole grasping pipeline into one step.

Methods that focus on the grasp computation include [11] which introduced SOTA class-agnostic grasping method for 2-finger grippers using segmentation from [5]. Grasp Pose Detection (GPD) in [12] also introduced a method that predicts a successful 2-finger gripper grasps from point clouds.

The first work that utilized the category-agnostic segmentation for robotic grasping is SD Mask R-CNN in [13], from which our approach differs in 2 aspects. First, they only use depth data and we use RGB or RGB-D data. Second, their approach focuses simply on a bin picking environment where the camera view is always fixed and looking directly downwards at the object, while ours segment objects in 6D free camera space.

[5] introduced a method that is the closest approach to this work. The approach uses RGB-D feature embeddings followed by a clustering algorithm then a refinement step. The difference compared to our method is that our approach uses CNN for the whole segmentation process.

[7] introduced a network to segment unseen objects using 3D data representation instead of RGB or RGB-D images. The same authors in [6] refined this approach in RICE using a graph based representation to separate over-segmented and join under-segmented objects. RICE showed SOTA performance against other methods.

Several datasets for object segmentation and object pose estimation exist. The published data can be acquired from one of three sources: real-world data, synthetic data, or a data generator.

YCB [14], LM [15], T-LESS [16], ITODD [17], and HOPE [9] are examples of datasets that provide tabletop and random scenes samples of cluttered objects. BlenderProc generator [18] was used during the BOP challenge [19] to generate several photo-realistic synthetic data for these datasets.

Stillleben generator [20] was used to generate SynPick dataset [21] and NVIDIA Dataset Synthesizer [22] was used to generate NVIDIA FAT dataset [23]. Several other works used PyBullet engine [24] to generate synthetic data with lower quality rendering process.

The datasets that introduce work closest to our DoPose dataset are OCID [25] and OSD [26] for tabletop scenes and WISDOM [13] for bin scenes. Our main plus against OCID is the variation in the data, as in each scene objects are rearranged from scratch. OSD dataset provides a few number of scenes. WISDOM dataset is the only dataset that includes bin scenes. Still, it is limited to 300 sample images and a fixed camera view.

Even though many other datasets exist, they are either synthetic, only provide few number of samples, or they provide tabletop scene setups only. This encouraged us to collect and publish our novel DoPose dataset. It covers highly occluded closely stacked objects with different scene setups and it is not limited to tabletop setups. Our dataset is collected in 2 scene setups, tabletop and bin. These setups will cover sufficient data variance so that we our models generalize better on unseen data as discussed in section IV-A. In section IV-B, we will evaluate the fine-tuning performance of our model on real data against the state-of-the-art.

III. METHOD

In this section, we first point our methodology of data collection and our process for creating our DoPose dataset. Second, we explain how the first training step is carried out by training on synthetic data. Third, we look at how the collected real data are utilized in the second training step to fine-tune our RGB and RGB-D models.

A. Dataset and annotation tool

As manual annotations are time consuming and labor intensive, our approach uses a semi-automated annotation process.
TABLE I: The table shows the evaluation of our RGB and RGB-D models on NVIDIA FAT synthetic dataset (validation set) and evaluation on our DoPose dataset. We carried out the evaluation on different scene from DoPose setups (tabletop and bin) separately.

|               | RGB                |            |            | RGB-D               |            |            |
|---------------|--------------------|------------|------------|--------------------|------------|------------|
|               | segmentation       | bounding box |           | segmentation       | bounding box |           |
|               | AP     | AR     | AP     | AR     | AP     | AR     | AP     | AR     |
| Synthetic FAT [23] (validation set) | 63.71 | 68.7  | 52.50 | 60.7  | 61.87 | 65.3  | 56.3  | 60.1  |
| Real (ours)   | tabletop        | 68.6  | 74.2  | 65.8  | 73.5  | 68.4  | 72.6  | 64.1  | 67.0  |
|               | bin             | 44.7  | 58.4  | 52.8  | 66.8  | 40.0  | 49.8  | 44.4  | 53.5  |

For each scene captured, both RGB and depth images along with the camera transformations of different view angles are obtained as shown in figure 2. The dataset is saved in the BOP format [27], which is a standardized format for 6D pose estimation. Using this format allows us to facilitate our dataset creation. For each of our objects we have a 3D mesh model. We use our annotation tool to manually annotate 6D pose of each object of the centered sample (figure 2 center) with our 3D meshes, then the annotation tool using the camera transformation calculates the 6D pose of all objects in all view angles. The BOP toolkit is then used to generate the segmentation masks for all samples and angles along with the COCO format JSON files. The RGB and Depths images are captured with Zivid Two structured light camera. The camera is mounted on a Kuka iiwa LBR 14 robotic manipulator. The interface and data collection scripts are ROS python scripts. The 3D models were made by assembling the point clouds captured by our Zivid Two camera using the CloudCompare software.

We include one extra annotation file that is not standardized in the BOP format, which is the scene transformations between the camera for each view angle and the world frame. We also accompany our code with a script that generates annotated point cloud data. This way our dataset can be used for multiple purposes: Instance segmentation, 6D Pose estimation, and point cloud segmentation.

For the bin setup, the data contains 183 scenes with 2150 image views. In those 183 scenes, 35 scenes contain 2 views, 20 contain 3 views, and 128 contains 16 views. And for the tabletop setup, the data contains 118 scenes with 1175 image views. In Those 118 scenes, 20 scenes contain 3 views, 50 scenes with 6 views and 48 scenes, with 17 views. So in total our data contains 301 scenes and 3325 view images. Most of the scenes contain mixed objects. The dataset consists of 18 objects in total. 5 among the 18 objects were chosen to look a bit similar (not the same object) to YCB objects on purpose to be able to study the effect of generalization on similar and totally different objects as several existing datasets use the YCB objects. Figure 3 shows the distribution of the objects in the dataset.

**B. RGB and RGBD category-agnostics segmentation**

We train the RGB model and the RGB-D model using a two-step training process. The RGB model is a pretrained on COCO dataset [28] before running through our two-step training. The RGB model uses Mask R-CNN [29] implementation from detectron2 [30]. The RGB-D model uses the same network but we extended the input to 4 Channels instead of 3. As There is no pre-trained RGB-D model, the RGB-D model is trained from scratch.

Our first-step training uses NVIDIA FAT [23] synthetic dataset for both of our models (RGB and RGB-D). The main point to keep in mind when training models for category-agnostic segmentation is under-training. This is important because of 2 reasons. First, the concept of understanding an object should not be bounded to objects similar to the dataset objects. The second reason is that we are training on synthetic data, so we should avoid over-fitting our model to synthetic data. Figure 4, shows the training and validation loss for training the RGB model. Our DoPose dataset (which will be used for fine-tuning in the next section) is used as the validation set during this first training step. This is required because NVIDIA dataset scenes are very similar, so the validation loss would keep decreasing along with the training loss endlessly which means that we would not be able to know when to exactly stop the training. The red line presents...
where our first-step training ends and the model at that point would be used for the second step training. After analysing the output segmentation masks of the models produced at different training steps, we noticed that the training process affects the confidence of the masks more than the mask quality. Overfitting would lead the model to have over-confidence in random background mask batches and highly occluded over-segmented object masks.

The RGB model (pre-trained) is trained for 1.7k iterations, and the RGB-D model (trained from scratch) is trained for 100k iterations. Figure 5, shows the segmentation using our RGB-D CNN. Table I shows the COCO metrics of our CNNs on the synthetic validation set and our DoPose real data. From the evaluation table, we observe that the RGB-D model achieved slightly better average precision and slightly less in the average recall. But both models were able to generalize on real data. From this we can observe that pre-training greatly affected the amount of data needed but didn’t affect the performance between our RGB and RGB-D models. We can also perceive that both models achieved higher average precision on real tabletop data than real bin data, mainly because the table data looks more similar to the synthetic training data and it is also less occluded. To enhance our model performance on occlusion we do the second-step training on real data.

C. Fine tuning with our real data

The second-step training is done for 500 iterations (less than 2 epochs) for both our RGB and RGB-D model. While this number of iterations seems low, we were able to notice from our qualitative analysis that the longer the training, the more the masks will overfit. Consequently, over-fitting would lead to over-segment or under-segment the objects or obtain false positive segmentation masks. We use only the bin samples from our DoPose dataset for the fine-tuning, so we need to evaluate our models on a different dataset. In the next section IV, we evaluate our RGB model on three unseen datasets with different objects and different metrics.

IV. Results

In this section, we evaluate our models on unseen datasets with unseen objects in different environment setups, compare our RGB model to SOTA. Finally we show how we can compute suction grasp points from our segmentation masks.

A. Evaluation on other datasets

We evaluate our RGB model on the NVIDIA HOPE dataset [9] which includes unseen objects on many random setups (tabletop, floor, shelf, couch, ..). We also evaluate our model on a proprietary dataset from an industrial partner made for evaluating bin picking segmentation methods.

1) Evaluation on HOPE dataset: Table II shows the evaluation of our RGB model on NVIDIA HOPE dataset. Figure 1, shows a sample of the evaluation with our RGB model, where good performance could be observed. The table also shows the effect of fine tuning on slightly increasing the average precision.

2) Evaluation on proprietary dataset for bin-picking: Table III shows the evaluation results of our fine-tuned RGB model. Figure 6 shows 2 segmented samples of the proprietary dataset. What makes our RGB model practical for such industrial cases is that our mask confidence is associated with
occlusion. So high confidence masks represent the objects that are easier to reach and lower confidence represent objects that are occluded and may have been over-segmented. So a simple grasping approach would be to aim for segmentation masks with the highest confidences.

TABLE III: Evaluation on proprietary dataset for bin picking with our fine-tuned RGB model

| Method       | AP  | AR  | AP  | AR  |
|--------------|-----|-----|-----|-----|
| segmentation | 51.518 | 0.572 | 49.685 | 0.553 |

Fig. 6: Segmentated samples of the proprietary dataset.

B. Comparing to state-of-the-art

We compare our RGB model to two SOTA methods. The first one with an RGB-D approach from [5] with refinement included in the same work, and the second method (UOIS [7]) which uses the data in 3D format with refinements implemented in [6]. Images in figure 7 and 8 used for comparison are obtained from the methods’ example code available online.

Figure 7, shows the comparison against the first method. We can observe from the figure that our method clearly recognizes sharp object edges against that method even though our model used RGB data only. Also, our method was able to recognize objects placed inside other objects as shown in the third row samples.

Figure 8, shows the comparison against the second method (UOIS). As our method includes no background removal we notice that some of the background objects are segmented as well but this can be easily excluded with post processing from the depth image. But in general our method achieved comparable qualitative results.

We are mainly interested in comparing with the methods before any refinement steps. Nevertheless table IV shows that our model was able to achieve comparable results to the state-of-the-art methods even without any refinement steps. Also, it should be considered that our model uses RGB data only in contrast to the other methods that used depth or 3D representation data. This shows that our data and approach for training can boost the performance of the already existing architectures before adding any refinement steps. This evaluation was carried out using Overlap P/R/F metrics introduced in [7] as evaluated in [6]. We carried out the same evaluation metric on our RGB model using evaluation scripts from [6].

TABLE IV: Comparing to SOTA methods as benchmarked in [6] using overlap P/R/F metrics introduced in [7]

| Method              | P  | R  | F  |
|---------------------|----|----|----|
| Mask R CNN          | 80.3 | 79.8 | 79.3 |
| Mask R CNN + RICE   | 92.3 | 91.8 | 92.0 |
| UOIS-Net-3D         | 86.3 | 88.6 | 87.5 |
| UOIS-Net-3D+RICE    | 89.7 | 91.9 | 90.7 |
| UCN                 | 91.6 | 92.8 | 91.9 |
| UCN+RICE            | 92.5 | 93.2 | 92.5 |
| ours                | 90.4 | 89.3 | 89.8 |

C. Grasping and pipeline implementation

As our main goal is oriented towards robotic grasping, we show how to compute suction grasp points using our CNN algorithm. I explains the step of calculating the grasp position and orientation. The position is the center of the biggest plane calculated by RANSAC. The orientation is the halfway vector of the normal at the center of such plane. The orientation uses the halfway vector fixing the Y-axis, as the normal is a vector and not an orientation. Figure 9 shows the grasps computed for a sample from NVIDIA HOPE dataset. Using work in [11] or [12] grasping can be extended to 2-Finger grippers.

Algorithm 1 Computing suction grasp point from mask segment

1: \( m_{rgb} \leftarrow \text{mask}(rgb) \)  \( \triangleright \) Masked RGB
2: \( m_{depth} \leftarrow \text{mask}(depth) \)  \( \triangleright \) Masked depth
3: \( \text{point}_\text{cloud} \leftarrow \text{point}_\text{cloud}(m_{rgb}, m_{depth}, \text{matrix}) \)
4: \( \text{plane} \leftarrow \text{compute}_\text{planes}(\text{point}_\text{cloud}) \)  \( \triangleright \) biggest plane
5: \( \text{normals} \leftarrow \text{compute}_\text{normals}(\text{plane}) \)
6: \( \text{center} \leftarrow \text{center}(\text{plane}) \)
7: \( \text{orientation} \leftarrow \text{halfway}_\text{vector}(\text{normal}[\text{center}]) \)
8: \( \text{grasp} \leftarrow (\text{center}, \text{orientation}) \)
9: \( \text{return} \text{grasp} \)

V. Conclusion and future work

In this work we showed how our multi-step training along with the careful choice of the synthetic training data and fine-tuning with our DoPose real dataset can reduce the sim2real gap.

Our quantitative and qualitative analysis shows that our model is able to produce comparable results to other approaches using RGB data only and without any fine tuning or refinement. While RGB models are much easier to deploy, reusing our data to train 3D CNN based approaches should lead to better results than training with unrealistic synthetic data without fine-tuning. Thus, the future extension of our work is to use the same data but for 3D approaches rather than 2D images. There is also the possibility to use our model to do category-agnostic 6D pose estimation.
Fig. 7: Comparing our RGB model to the method of Xiang et al. [5] with their initial and refined segmentation.

Fig. 8: Comparing our RGB model to UOIS [7]
