Detection and Recognition of Dinning Table Objects Using RGB-D Camera for Robotic Applications

Jakob Baumgartner
Faculty of Electrical Engineering,
University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
jakob.baumgartner@fe.uni-lj.si

Abstract

In this paper we present a pipeline for 3D segmentation and classification of dining table objects. The created pipeline can be used in real robot applications where we work in an unstructured environment. The algorithm created is able to correctly segment previously seen or unseen objects that might appear on a dining table and classify known common objects. The input to our pipeline is a single RGBD image. Special attention was paid to correctly segment transparent objects, which are usually not well detected by commercial depth sensors. Such objects are much better detected on the RGB image that we use to enhance the depth image for transparent objects. We validate our pipeline on an object segmentation database on the table as well as on our own images captured with the Intel Realsense camera.

1. Introduction

RGBD (red, green, blue, depth) sensors [20] use a combination of traditional RGB (red, green, blue) cameras that capture colour information about an object and depth sensors that measure the distance between the sensor and the object. In the context of a dining table, RGBD sensors can be used to allow the robot to recognise and classify a variety of objects, such as dishes, glasses, utensils and food. By combining the colour information from the RGB cameras with the depth information from the depth sensors, the robot is able to create a detailed 3D model of the objects on the table so that it can recognise their shape, size and orientation.

RGBD object detection and classification algorithms have a number of advantages over traditional object detection algorithms that rely solely on colour information[1]. One advantage of using RGBD information is that the algorithm can better handle occlusion and clutter in the scene. Occlusion occurs when an object blocks the view of another object, and clutter refers to the presence of multiple objects in the scene that may be difficult to distinguish. By using depth information, the algorithm can better understand the spatial relationships between objects and more accurately detect and classify objects even when they are partially occluded or surrounded by clutter. Example of occlusion on a dining table is when a plate is partially hidden behind a glass. Another benefit of using RGBD information is that it enables the algorithm to better distinguish between objects that are similar in colour but different in shape or size. This can be particularly useful when there are multiple objects of the same colour in the scene, as the algorithm can use the depth information to distinguish them based on their shape and size. In our case it can help us distinguish between colourful tablecloths as a background and objects sitting on them.

However, the use of RGBD object recognition is still less common than the use of RGB object recognition. One reason for this is that RGB sensors are more widespread and less expensive than RGBD sensors. Another reason is the greater difficulty in labelling and especially segmenting such data sets due to the added dimensionality of data. As a result, there are far fewer and less comprehensive RGBD datasets than RGB datasets. For this reason, researchers are trying to train new algorithms on simulated datasets. Such datasets exhibit perfect segmentation, but there is an open problem in transferring the trained models from simulation to working with real data. While synthetic depth trained algorithms usually transfers quite well to real world, simulated RGB images lack the textures and lighting conditions of the real world, so algorithms trained on synthetic RGB data do not transfer well to the real world. This has changed to some extent with the advent of commercial sensors such as Microsoft Kinect and Intel Real-Sense. In addition, algorithms that use RGBD data often require more computing power and resources to analyse the data, which can make

them difficult to implement in real-time applications or on devices with limited resources.

In this paper, we aim to create an image processing pipeline that could be used on a robot in the hospitality industry to segment objects on a dining table. We can call this task partially structured because it contains a number of known and unknown objects. A typical input image contains both known objects, such as glasses and plates, and unknown objects, such as items left behind by customers in a restaurant. In order for our robot to clear the table after the guests, it must be able to recognise each object on the table so that it can remove it from the table. We imagine that the robot has three different compartments for the collected items. One compartment for the dirty dishes, one compartment for the rubbish left behind and one compartment for other items, where guests' forgotten items such as wallets and mobile phones are kept, as well as for the tips that guests leave.

In developing this object recognition pipeline, we have paid particular attention to solving challenges that may arise in the dining table environment. In addition to a variety of known and unknown objects, we also deal with objects such as cutlery, napkins and coins that are too thin or too small to be detected in the depth image. Another challenge for this pipeline is the large number of transparent objects such as cups and bottles that commercially available depth sensors cannot detect properly.

2. Related Work

In recent years, there has been significant progress in the field of unseen object instance segmentation, which aims to detect and segment instances of objects that have not been seen during training. This is a challenging task that requires the ability to generalize to new objects while still accurately segmenting instances.

Several methods have been proposed to address this problem, including Mask R-CNN by He et al. (2017) [10] [9], which extends the Faster R-CNN framework to predict instance masks in addition to object bounding boxes. Panoptic FPN by Kirillov et al. (2019) [13] also builds upon the FPN architecture to generate panoptic segmentation, including both object instances and stuff segments. YOLACT by Bolya et al. (2019) [4] uses a fully-convolutional model, that trades some of the segmentation performance for being able to run real-time. SolO-instance by Wang et al. (2020) [21] uses a set of prototypes to represent object instances, which are then refined in a manner that allows for instance-specific predictions. PointRend by Kirillov et al. (2020) [14] uses a neural network module that performs point-based segmentation predictions at adaptively selected locations based on an iterative subdivision algorithm. PointRend achieves significantly better segmentation results on several popular datasets compared to previous approaches. PolarMask by Xie et al. (2020) [25] introduces an anchor-box free and single-shot instance segmentation method that can be used as a mask prediction module for instance segmentation. PointGroup by Jiang et al. (2020) [12] presents a new end to-end bottom-up architecture, specifically focused on better grouping the points by exploring the void space between objects.

The following methods focus specifically on the segmentation of unseen instances. In the absence of sufficiently large datasets for segmenting instances in cluttered environments, these approaches use synthetic data and generalise the algorithms so that they are applicable to the detection of real-world data [23]. These methods are most comparable to our method of choice, which is also trained on synthetic data. Being state-of-the-art, these methods offer similar performance to the method we chose for segmentation, while working at the same speed or slower. UCN by Xiang et al. (2021) [22] uses RGB-D feature embeddings from synthetic data to perform instance segmentation. MSMFormer by Lu et al. (2022) [15] uses a new transformer architecture that simulates the von Mises-Fisher (vMF) mean shift clustering algorithm.

These methods demonstrate the progress that has been made in unseen object instance segmentation and highlight the potential for future developments in this field.

3. Methods

Our pipeline consists of three separate components, as can be seen in the figure 1. Together, they allow us to solve the problem of instance segmentation and classification of objects on a dinner table in a real-world environment. The input to our pipeline is a single RGBD image. Our method returns a list of segmented instances with segmentation masks and the positions of the objects. Our method also returns classifications for objects that are common in the given environment.

The first component in our pipeline is a depth estimator for transparent objects [18]. Transparent objects are common elements in our application environment. One of the most common categories on dinner table are transparent glasses for sodas and juice. Transparent bottles and salad bowls are also common. Transparent objects possess unique visual properties that make it incredibly difficult for standard 3D sensors to make accurate depth estimates. They do not adhere to the geometric light part assumptions of classic stereo vision algorithms, the consequence of which is that commercially aveliable sensors producing noisy or distorted approximations of the surfaces behind them. To solve this problem, we use the ClearGrasp algorithm to derive the depth data of the transparent objects from the RGB data and the surrounding non-transparent surfaces.

The second component in our pipeline is an Unseen Object Instante Segmentator (UOIS) [24]. Due to the unstruc-

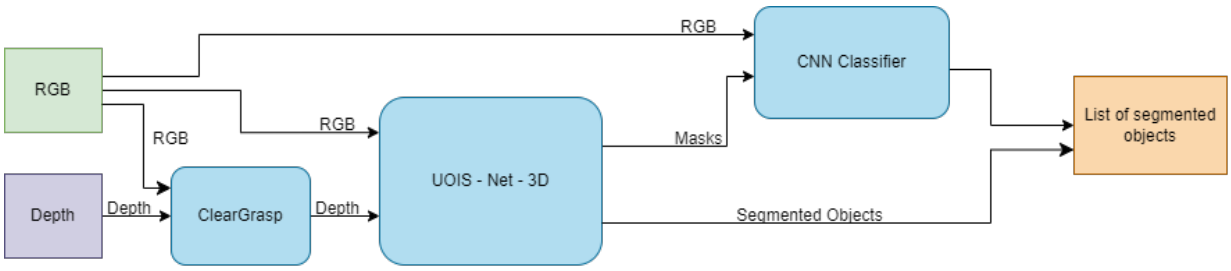

Figure 1. Full proposed pipeline.

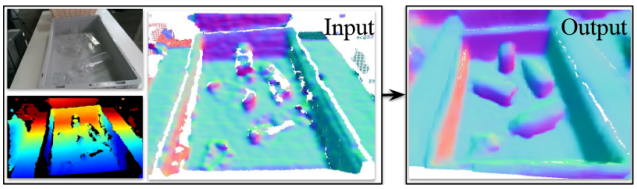

Figure 2. We use ClearGrasp algorithm to estimate missing depth data of transparent objects.

tured nature of our task, it is infeasible and impractical to model every possible object in our environment. For a robot to successfully manipulate objects on a dining table, it must be able to recognise and segment instances of previously seen and unseen objects [23]. This component takes as input RGB and point cloud data and provides segmentation masks and positions for each known and unknown object on the dining table. It is able to segment objects even if they are cluttered together or partially occluded. Before we can use this stage, we need to backproject Depth image back onto the point cloud data [5], for this we need to know the intrinsic matrix of the camera used.

The third component of our pipeline is a CNN classifier. UOIS provides instance segmentation masks. With these, we can cut out individual objects and then perform a classification for the common objects on a dining table. These detections can be passed on to the robot so that it can separate kitchen utensils such as plates, cups and cutlery from other objects and the rubbish left on the table.

3.1. Transparent objects depth estimator

Transparent objects do not adhere to the geometric assumptions about the light path made in classical stereo vision algorithms. This makes it difficult for commercial 3D sensors to correctly detect depth information. ClearGrasp is an algorithm that leverages deep learning with synthetic training data to create accurate depth profiles of transparent objects.

3.1.1 Method

ClearGrasp algorithm [18] is a modified version of completion pipeline proposed by Zhang and Funkhouser [26]. Algorithm consists of four stages.

Transparent object segmentation At this stage we use an image segmentator to predict pixel-wise masks of transparent objects. For this we use Deeplabv3+ [7] [8] with a DRN-D-54 [27] backbone to remove all depth pixels corresponding to transparent surfaces.

Surface normal estimation In this stage we also use a combination of Deeplabv3+ with a DRN-D-54 backbone, this time to predict surface normals on the RGB input image.

Boundary Detection In the next stage, we again use a combination of Deeplabv3+ and DRN-D-54, this time to label each pixel of the RGB image into one of three categories. The categories are Non-Edge, Occlusion Boundary (object behind another) and Contact Edges (objects sticking together).

3.1.2 Global Optimisation for Depth

Finally, we can use the depth image with the removed pixels of the transparent objects, the surface normal predictions, the occlusion edges and the contour edges obtained in the previous steps to reconstruct the missing depth regions of the transparent objects.

$$E = \lambda_D E_D + \lambda_S E_S + \lambda_N E_N$$

At this stage, we solve a system of equations with the aim of minimising the weighted sum of the squared errors of three terms: E_D - distance between estimated and observed depth, E_S - difference of distance between depths of neighbouring pixels and E_N - measure of difference between estimated depth and predicted surface normals. λ are the weights of the different terms.

An example of the prediction method is shown in figure 2.

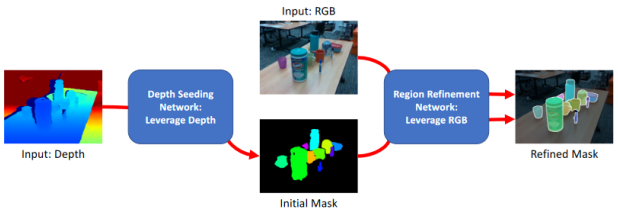

Figure 3. UOIS-Net-3D consists of two neural networks, that process Depth and RGB data seperately, to produce instance segmentation masks.

Figure 4. We use UOIS-Net-3D algorithm to segment all objects in the image.

3.1.3 Dataset

The algorithm was trained on a synthetic dataset with nearly 50,000 distinct instances. Blender's physics engine with the ray tracing rendering engine Blender Cycles was used to create the dataset. 9 different CAD models of transparent objects, many different background textures and different simulated lighting conditions were used. Each instance of the dataset contains an RGB image, a depth image, semantic segmentations of all transparent objects, the camera position, the poses of all objects and the surface normals image.

3.2. Unseen Object Instance Segmentator

To divide cluttered objects on the dining table into individual instances, we use Unseen Object Instance Segmentator (UOIS-Net-3D) [24]. The algorithm takes as input a single RGBD image and outputs object instance segmentation masks for all previously seen or unseen objects. The output of the algorithm does not contain specific categories of objects, only background, table and object instance.

The algorithm consists of separately trained neural networks and multiple stages. A simple representation of the algorithm can be seen in the figure 3.

3.2.1 Method

Depth Seeding Network - DSN The first stage of the instance segmentation network takes as input an organised point cloud D with XYZ coordinates. The reason why we use depth data first is because of the better generalisation of neural networks trained on simulated depth data to real-world scenarios, compared to networks trained on synthetic RGB data. The organised point cloud is passed through the DSN network, which returns a segmentation mask for each pixel for the given image and the 3D offsets to the object centres. $V' \in \Re^{H \times W \times 3}$.

Initial Mask Processing Module - IMP In this phase of the algorithm, we apply some basic image techniques to the initial segmentation mask. First, we apply an opening operation (erosion+dilation) to each mask instance to remove salt/pepper noise. Then we apply a closing operation (dilation+erosion) to remove small holes in the mask. Finally, we remove minor unconnected components of each mask.

Region Refinement Network - RRN At this stage, we use another neural network to refine the segmentation mask obtained from the depth image with additional RGB information. The input of the RRN is a 4-channel cropped image consisting of a concatenated cropped RGB and a single object mask. We use a U-Net [17] network architecture. During the training process, we first train the DSN network and then use the output segmentations for RRN training. During training, we first use the segmentation masks before input to the RRN. This allows us to simulate the noise and errors that occur when using data from a real depth sensor but are not seen on the clear synthetic images. For pertrubation, we use augmentation techniques such as translation, rotation, addition, cutting, morphological operations and random elipses. For an example of a segmentation result, see image 4.

3.2.2 Dataset

The method was trained on a synthetic dataset called the Tabletop Object Dataset (TOD). The dataset consists of 40,000 synthetic scenes. Each scene consists of a SUNCG [19] home environment. In this environment is a ShapeNet [6] model of a table with 5 to 25 ShapeNet [6] models of objects randomly placed or stacked on the table. For each object, the constalation dataset contains seven different views. Due to the limitations of the simulator used (PyBullet), RGB textures do not look photorealistic. The depth data, while clear compared to data from an actual commercial sensor, is closer to a realistic representation.A synthetic dataset was used as there is no dataset collected and annotated in the real world that meets the requirements for training such an application.

3.3. Classifier

The final component of our pipeline is an image classification network. We use segmentation masks obtained from UOIS to cut out each object from the RGB image and then use the ResNet [11] convolutional model to classify each object into one of the categories listed. The categories are:

Cup, Plate, Bowl, Glass, Cutlery, Purse, Napkin, Tin, Bottle, Other. The idea behind this is that a robot working in the hospitality industry can divide the objects into a section for laundry and a section for the bin / other objects.

Dataset To train our classifier, we used images from the ImageNet dataset for the specified categories. ImageNet consists of more than 20,000 categories, but we selected for training only images from the categories we chose and a set of images with random objects to train the so-called "other" category. Half of the images we selected are from objects we want to classify and the other half of the images are from a number of different uncategorised objects.

4. Experiments

In order for our pipeline to be useful for robotic applications, it must be able to work with a high success rate in each part of the pipeline and execute at a reasonable speed. In the experimental part of this article, we therefore focus on testing parts of the pipeline individually and in conjunction with others.

To perform the experiments, we have created the pipeline of algorithms described. We use PyTorch [16] implementations [3] [2] of each of the methods described above. To avoid problems with different versions of PyTorch as well as other dependencies, we run each of the components in a separate Docker container.

The experiments were conducted on a computer with Intel® Core™ i7-12700 CPU, GEFORCE GTX 1080 Ti GPU and 32GB of RAM performed.

4.1. Object Segmentation Database (OSD)

Our experiments can be divided into two parts. First, we tested our system with the Object Segmentation Database (OSD).

We perform an ablation experiment to see how removing the augmentation network for transparent objects at the beginning of the pipeline affects the segmentation and classification results. The purpose of this test was to see if using augmented data affects the segmentation of non-transparent objects in any way.

We test how accurate our classifier is.

We also measure the time it takes each of the pipeline components to perform computations. We pay attention to how the number of opaque objects affects the computation times.

4.2. Intel RealSense D435

In the second part of the experiments we use the Intel RealSense D435 camera to capture a series of RGBD images. The goal of this experiment is to see if our pipeline works on new data. Besides general examples, we use the camera to capture several RGBD images of situations that are not included in the OSD database.

We take 25 different images that contain transparent objects. We then compare the results of segmenting transparent objects with and without using the ClearGrip component to enhance the depth images.

Then we take a series of images of small and thin objects. The RGBD camera has a resolution of 2% error at distances of less than 1m. This means that certain small or thin objects are not visible on the depth image. Examples of such objects are cutlery, paper pages or menus. We want to test whether the segmentation algorithm used can recognise these objects based on RGB data alone. Since the first detection step in the UOIS uses depth information, we expect that these objects will not be detected.

References

[1] *RGB-D Image Analysis and Processing*. Advances in Computer Vision and Pattern Recognition. Springer International Publishing, Cham, 2019.

[2] Cleargrasp. `https://github.com/Shreeyak/cleargrasp`, 2021.

[3] Unseen object instance segmentation for robotic environments. `https://github.com/chrisdxie/uois`, 2021.

[4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. (arXiv:1904.02689), Oct 2019. arXiv:1904.02689 [cs].

[5] Gazi Erkan Bostanci, Nadia Kanwal, and Adrian Clark. Augmented reality applications for cultural heritage using kinect. *Human-centric Computing and Information Sciences*, 5:1–18, Jul 2015.

[6] Angel Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. Dec 2015.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. (arXiv:1802.02611), Aug 2018. arXiv:1802.02611 [cs].

[9] Michael Danielczuk, Matthew Matl, Saurabh Gupta, Andrew Li, Andrew Lee, Jeffrey Mahler, and Ken Goldberg. Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data. (arXiv:1809.05825), Mar 2019. 19 citations (Semantic Scholar/arXiv) [2023-01-30] arXiv:1809.05825 [cs].

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. (arXiv:1703.06870), Jan 2018. arXiv:1703.06870 [cs].

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. (arXiv:1512.03385), Dec 2015. arXiv:1512.03385 [cs].

[12] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. (arXiv:2004.01658), Apr 2020. 158 citations (Semantic Scholar/arXiv) [2023-01-28] arXiv:2004.01658 [cs].

[13] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollar. Panoptic feature pyramid networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 6392–6401, Long Beach, CA, USA, Jun 2019. IEEE.

[14] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 9796–9805, Seattle, WA, USA, Jun 2020. IEEE.

[15] Yangxiao Lu, Yuqiao Chen, Nicholas Ruozzi, and Yu Xiang. Mean shift mask transformer for unseen object instance segmentation. (arXiv:2211.11679), Nov 2022. 0 citations (Semantic Scholar/arXiv) [2023-01-31] arXiv:2211.11679 [cs] version: 1.

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. (arXiv:1912.01703), Dec 2019. arXiv:1912.01703 [cs, stat].

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] Shreeyak S. Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. Clear-grasp: 3d shape estimation of transparent objects for manipulation. (arXiv:1910.02550), Oct 2019. 100 citations (Semantic Scholar/arXiv) [2023-01-28] arXiv:1910.02550 [cs, eess] version: 2.

[19] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. (arXiv:1611.08974), Nov 2016. arXiv:1611.08974 [cs].

[20] Kyriaki A. Tychola, Ioannis Tsimperidis, and George A. Papakostas. On 3d reconstruction using rgb-d cameras. *Digital*, 2(33):401–421, Sep 2022.

[21] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. (arXiv:1912.04488), Jul 2020. arXiv:1912.04488 [cs].

[22] Yu Xiang, Christopher Xie, Arsalan Mousavian, and Dieter Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. (arXiv:2007.15157), Mar 2021. 47 citations (Semantic Scholar/arXiv) [2023-01-28] arXiv:2007.15157 [cs].

[23] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. (arXiv:1907.13236), Jul 2020. 58 citations (Semantic Scholar/arXiv) [2023-01-30] arXiv:1907.13236 [cs].

[24] Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. Unseen object instance segmentation for robotic environments. (arXiv:2007.08073), Oct 2021. 48 citations (Semantic Scholar/arXiv) [2023-01-28] arXiv:2007.08073 [cs].

[25] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. (arXiv:1909.13226), Feb 2020.

[26] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[27] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 175–185, Jun 2018.