Da-TRASH: Depth-appended Tabletop Recycling Algorithm for Segmenting Havoc

Ashwin Saxena
Computer Engineering
University of Michigan
Ann Arbor, USA
ashwinsa@umich.edu

Andrew Scheffer
Computer Science and Engineering
University of Michigan
Ann Arbor, USA
drewskis@umich.edu

***Abstract*—In this work, our team aims to reproduce and extend upon the paper *Learning RGB-D Feature Embeddings for Unseen Object Instance Segmentation* by Y. Xiang et. al [1]. This work makes use of non-photorealistic, synthetic RGB + Depth data to produce surprisingly accurate instance segmentation masks of unknown objects. In this report, our team will validate the specific result of the paper suggesting that combining RGB and depth feature vectors elementwise is most effective for this task. Additionally, our team will attempt to extend upon this work by adapting the model to produce adequate results on simple RGB images by first predicting the corresponding depth image with another machine learning model. This extension allows this instance segmentation model to be run on simpler cameras, without depth-sensing capabilities. We show improved segmentation accuracy of our new model, *Da-TRASH*, on trash segmentation datasets as well as traditional tabletop datasets.**

I. Introduction

This work presents a novel method for Unseen Object Instance Segmentation (UOIS) for robotic applications [1]. Segmenting unseen objects in cluttered scenes is an important perception task in robotics, especially in environments where the types of objects are potentially unknown (i.e. kitchens, machine shops, etc). Previous methods to solve this problem have been limited due to an absence of large-scale datasets of real images that contain objects in robotic manipulation scenes. Moreover, using simulation to create RGB images does not work well due to the sim-to-real domain gap.

This paper aims to utilize a dataset of completely synthetic RGB-D images to directly learn feature embeddings for every pixel in an image. These embeddings can then be produced for realistic images and clustered to segment unseen objects. To produce these feature embeddings for every pixel of an input image, the authors use a fully convolutional network with the goal of creating similar embeddings for pixels that belong to the same object and differing embeddings (defined by cosine distance) for pixels belonging to different objects. The question of how to incorporate depth data into the main convolutional network was also investigated, suggesting that training two networks (one for depth and one for RGB) and then adding the outputted feature maps elementwise was the most effective.

The main conclusion of the method provided in [1] is that it is possible to train feature embedding networks directly from non-photorealistic, synthetic RGB-D datasets. Additionally, the performance of this method is exceptional when optimized to a new clustering loss function that rewards embeddings that are similar when the pixels are a part of the same object and incentivizes different objects to have drastically different cluster centers in feature-space.

Our extension, *Da-TRASH*, extends upon this idea by not relying on accurate depth data, but instead predicting the metric depth data using a transformer-based encoder-decoder model named *ZoeDepth*. This extension allows the instance segmentation model to be run more accurately on cheaper hardware, making it more accessible. Fig. I depicts an example of the output of *Da-TRASH* method on a self-staged scene. Later, we will show that this new model proves particularly effective at segmenting cluttered waste scenes without depth data.

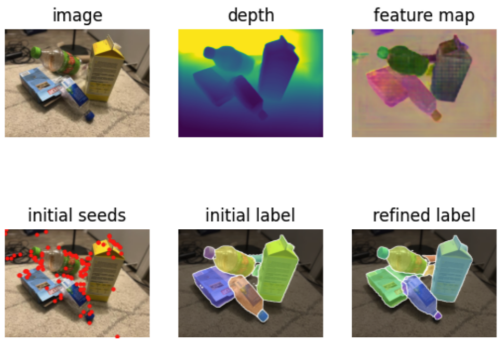

Fig. 1. Instance segmentation of a self-staged scene of miscellaneous recycling items using predicted depth.

II. Related Work

The task of unseen object instance segmentation has been an active area of research in the computer vision community recently. The majority of methods for object instance segmentation are based on deep learning techniques. A popular learning-based approach is to generate a plethora of synthetic data using object CAD models, then use this data as input to train neural networks [2]; however, these approaches often suffer with the ”sim-to-real gap” when the computer-generated

images are not realistic. This work shows that simulated, non-photorealistic RGB and depth images can be sufficient for UOIS.

Deep metric learning has also been a popular method for generating generalized feature embeddings for many tasks. For example the model, Facenet, by Schroff et. al generates [3] face embeddings (produced by different networks) that are compatible to each other and allow for direct comparison between each other. This transition from RGB colorspace to some arbitrary embedding space (with the mapping learned by neural networks) allows for state-of-the-art facial clustering and segmentation. The method presented for unseen object instance segmentation expands upon these ideas to cluster pixels that likely are from the same object.

III. Method

In essence, the main method of this paper can be broken into two steps:

1) Extract feature embeddings for every pixel in the RGB and Depth data
2) Group these feature embeddings into k object clusters using mean-shift clustering

A high-level depiction of this process can be seen in Fig. 2.

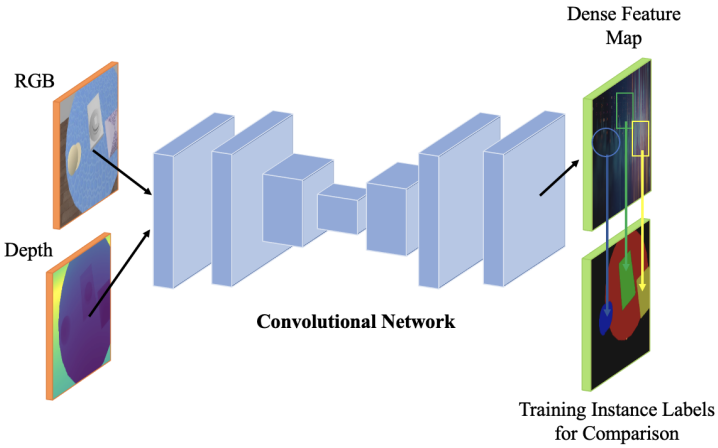

Fig. 2. Simplified illustration of the method for learning RGB-D feature embeddings using a fully convolutional network and a metric learning loss.

A. Feature Embeddings

The first step of this method is to back-project the depth data into an organized point cloud using the camera intrinsics. Next, as seen in Fig. 2, the input to the model is RGB and depth data (of the same size) which are fed into a pretrained fully convolutional network modeled after the 34 layer ResNet-8s model [4]. As stated in [1], various backbone architectures can be used such as the U-Net and VGG. The output matrix of this model has the exact same width and height as the input images but has depth C, where C is the length of the feature embedding for each pixel. In this paper, $C = 64$.

The goal of training this model is to ensure pixels from the same object are close together in embedding space whereas pixels from different objects are far from each other in embedding space. To ensure this, the authors apply a metric learning loss function to a subset of each image's pixels [1].

B. Combining RGB and Depth Data

One interesting aspect of this paper is how they investigated combining the RGB data and depth data for generating pixel-wise feature embeddings. This is an important step in the model because the paper is attempting to show that the *combination* of RGB and depth data improves results over past research which only utilizes depth data. This result of the paper is what our team will attempt to reproduce. To start, the authors present three different ways of fusing RGB and Depth (point cloud) data.

1) **Early Fusion** - the RGB image I and the point cloud image P are concatenated before feeding them into the network
2) **Late Fusion Addition** - I and P are fed into identically structured convolutional networks to compute two feature maps, then the two feature maps are added elementwise
3) **Late Fusion Concatenation** - similar to above, except the two feature maps are concatenated

The authors performed experiments that suggested that the Late Fusion Addition approach is most effective, and our main goal will be to reproduce this result.

IV. Validation Experiments

In this section of the report, our team will outline the experiments that we performed to validate the methodology described in [1].

A. Overfitting the Model

First, our team overfit the RGB-Depth Add architecture on a small subset of the dataset presented in [1]. Initially, we intended to retrain the model from scratch on all 40,000 scenes (with 7 images per scene); however, with our limited compute resources we discovered that this goal was infeasible.

Instead, we have selected 14 scenes (for a total of 98 training examples) to overfit the model on. To perform this experiment, we ran the computation on Google Colab with 4 images per batch (requiring about 8 GB of GPU RAM). We decided to run 64 epochs over the small training set to overfit the model. In Fig. 3 the average overall loss per epoch is plotted along with its two components: intra-cluster-loss and inter-cluster loss.

As seen in Fig. 3, the overall loss decreases steadily over time, increasing our confidence that the model is learning useful feature embeddings. Additionally, we can see the intra-cluster-loss does not decrease initially like the inter-cluster-loss, but decreases more steadily over a longer time period. This indicates that when the model is first initialized with random weights, it first prioritizes pushing cluster centers away from each other in the embedding space. The total training time was 1 hour.

	Overlap			Boundary			
	P	R	F1	P	R	F1	%
RGB	0.5739	0.7379	0.6354	0.3508	0.4972	0.3929	0.5220
Depth	0.8314	0.8658	0.8465	0.5619	0.5706	0.5611	0.8152
RGBD early	0.7935	0.8027	0.7944	0.5657	0.5912	0.5682	0.6879
RGBD add	**0.8667**	**0.8745**	**0.8702**	**0.6679**	**0.7126**	**0.6821**	**0.8336**
RGBD concat	0.8217	0.8549	0.8375	0.5670	0.6023	0.5764	0.6751

TABLE I: average recall, precision, and F1 score for overlap and boundary pixels in segmentation output of all OSD images

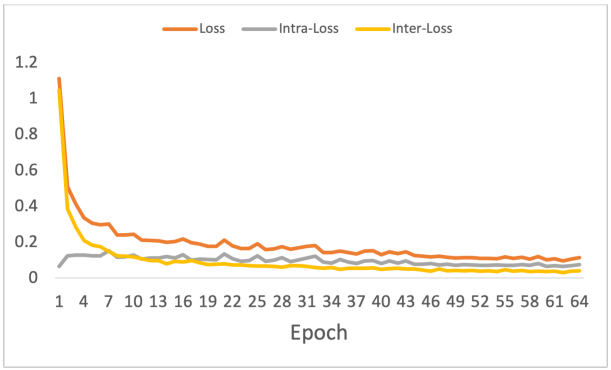

Fig. 3. Overfit model history for 64 epochs on a total of 14 scenes (98 images) from the training set.

Fig. 4. Projected point cloud data from OSD

B. Reproducing RGB + Depth Fusion Results

In this next section of the report, the methods and results related to the fusion of RGB and Depth data in [1] will be reproduced. The Object Segmentation Database (OSD) [5] will be used to evaluate the performance of the three methods of fusion discussed in the previous section. OSD contains 111 unique RGB-D images with up to 15 objects in any single scene with varying levels of object occlusion. Following [1], we will use precision and recall along with the corresponding macro F1-Score across all classes in the case that there is class imbalance in the segmentation task. These three metrics will be calculated once for the overlap of all pixels in the ground truth and predicted segmentation image, and once more for the overlap of decision-boundary pixels in the ground truth and predicted segmentation image. This is in an effort to obtain insight into the effectiveness of the models at predicting object boundaries.

Our team had particular issues with dealing with outdated Python packages at this step of the testing process. It turns out that the Point Cloud Library (PCL) wrapper used by the original authors is no longer maintained, requiring our team to fully understand the back projection of the depth maps into 3D space using the camera intrinsics. Fig. 4 shows an intermediate result of producing one of these depth point clouds with the updated code.

Once depth point clouds were being efficiently created, Table 1 was produced by evaluating the discussed metrics on five different model architectures: an RGB-only model, a Depth-only model, and the three proposed RGB+D fusion models. For each model, we also included the average percent of ground truth objects that had at least 75% overlap with a unique prediction. As seen in Table 1, the RGBD-add methodology outperforms all other proposed models. This not only shows that adding RGB and Depth data elementwise after being passed through their own networks is most effective but also shows that useful RGB features can be created using non-photorealistic synthetic data. This is shown in how the RGBD-add model outperforms the simplistic Depth-only model. Once this part of the method proposed in [1] was reproduced, we started working on the extension of developing *Da-TRASH*.

V. Algorithmic Extension

A. The Problem

Segmenting and classifying cluttered and unseen objects is problem relevant in several industries. Recycling and trash sorting is one domain where this technology can be leveraged to improve existing trash sorting algorithms. Only about 32.1 percent of recyclable waste is actually recycled [6]. At the heart of the problem are the inefficiencies of the waste sorting process (separating paper, plastic, metal, glass, etc.) due to the extremely complex and cluttered nature of the waste stream. Recyclable waste detection poses a unique computer vision challenge as it requires detection of highly deformable and often translucent objects in cluttered scenes with little to no context information usually present in human-centric datasets. For our algorithmic extension, we decided to tackle this challenge of segmenting waste by using the UOIS method presented in the paper and combining it with zero-shot depth estimation to make *Da-TRASH*, a waste segmentation algorithm using only RGB images.

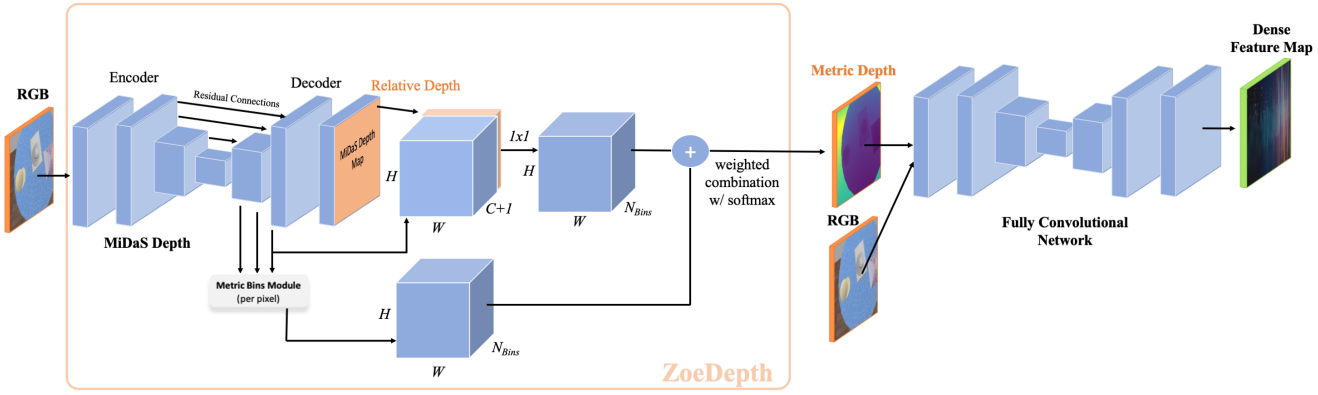

Fig. 5. Da-TRASH architecture. First, *ZoeDepth* is used to generate a depth map. An RGB image is fed into the MiDaS depth estimation framework. The bottleneck and succeeding four hierarchy levels of the MiDaS decoder (at 1/32, 1/16, 1/8, 1/4 and 1/2 of the MiDaS in- and output resolution) are hooked into the metric bins module. The metric bins module computes the per-pixel depth bin centers that are linearly combined to output the metric depth [7]. The predicted metric depth image, along with the RGB image, then pass through the fully convolutional network to generate a feature map that can be used to create segmentation labels.

B. Dataset

The dataset we used to test was the ZeroWaste dataset which contains in-the-wild industrial-grade waste pictures on a conveyor belt. ZeroWaste is the largest public dataset for waste detection. The creators of the ZeroWaste dataset concluded that current state-of-the-art detection and segmentation methods cannot efficiently handle this complex in-the-wild domain. In addition to the ZeroWaste dataset, we also used some pictures of trash that we took to see how generalizable our pipeline is.

Fig. 6. Example images from the ZeroWaste dataset

C. Depth Estimation

Since the ZeroWaste dataset contains no depth images, we decided to use depth estimation algorithms to construct *Da-TRASH* that would run using just RGB images. We decided to use *ZoeDepth* (Zero-shot Transfer by Combining Relative and Metric Depth) for depth estimation. *ZoeDepth* is a new state-of-the-art depth estimation algorithm which needs no pre-training to produce good results on unseen images. Before *ZoeDepth*, existing work either focused on generalization performance disregarding metric scale (relative depth estimation) or state-of-the-art results on specific datasets (limited metric depth estimation). *ZoeDepth* leverages an encoder-decoder model to compute a metric depth map. *ZoeDepth's* flagship model, ZoeD-M12-NK, is pre-trained on 12 datasets using relative depth and fine-tuned on two datasets using metric depth. The model uses a lightweight head with a novel bin adjustment design called the metric bins module for each domain. During inference, each input image is automatically routed to the appropriate head using a latent classifier. The framework admits multiple configurations depending on the datasets used for relative depth pre-training and metric fine-tuning. Fig. 5 shows the architecture of the *Da-TRASH* pipeline as it integrates *ZoeDepth* with UOIS.

VI. Extension Setup

Since we did not have access to GPUs on our local machines, we had to use Google Colab for running all our experiments. Google Colab does not exactly tell us what kind of GPUs it uses, but we used widely available libraries, so our experiments should be replicable by anyone with Google Colab access. Most of the experiments do not take much time besides the depth prediction step. Based on the size of the image, generating the depth map can take anywhere between 30 seconds to 2 minutes to create. When doing it on a large scale, this could be prohibitively slow.

VII. Extension Results and Discussion

A. OSD Dataset

When we first tried using *ZoeDepth* generated depth maps with the RGB images, we did not get good results. This is partly because of the unscaled values of the produced depth maps. We were using UOIS pretrained models which assumed that closer objects had lower values in the depth map and distant objects had higher values in the depth map. However, *ZoeDepth* generated depth maps that had inverse maps such that closer objects had higher values. Once this was accounted for, we began receiving promising segmentation results as seen in Fig. 7.

	Overlap			Boundary			
	P	R	F1	P	R	F1	75%
RGB	0.5739	**0.7379**	0.6354	0.3508	**0.4972**	0.3929	0.5220
Predicted Depth	0.7087	0.7182	0.6916	0.4726	0.4194	0.4231	**0.5485**
Da-TRASH	**0.7281**	0.7208	**0.7045**	**0.5235**	0.4446	**0.4592**	0.5263

TABLE II: OCID recall precision, and F1 score for overlap and boundary pixels in segmentation output

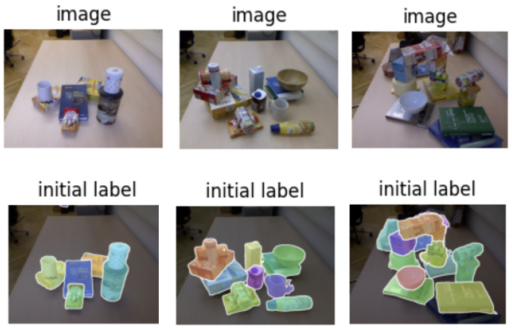

Fig. 7. Segmentations Results of Da-TRASH on OSD Dataset

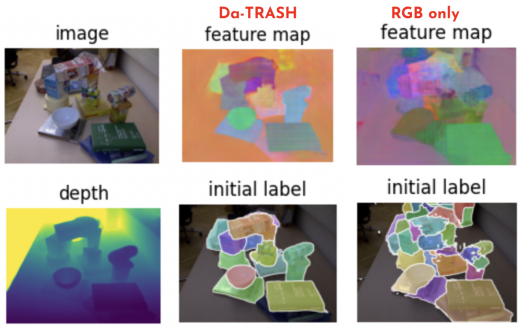

Fig. 8. Da-TRASH Segmentation compared to RGB-only segmentation

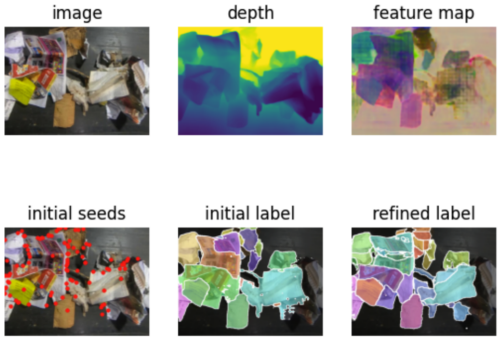

Fig. 9. Instance segmentation of a dense cluttered scene from ZeroWaste

To evaluate the performance of our new pipeline (which uses both RGB features and predicted-depth features) we compared the segmentation results to segmentations generated based on pure RGB images, and segmentations generated based on only the predicted depth images from the *ZoeDepth* model. These benchmarks can be seen in Table II. As seen, *Da-TRASH* performs significantly better compared to the RGB-only UOIS model in terms of precision and F1-score for both overlap and boundary datasets. However, the recall was slightly lower when compared to running the model on just RGB data. One way of interpreting the slightly lower recall but higher precision is that many of the predicted boxes are more accurate, but slightly more of the ground truth objects have been misclassified in our pipeline. The values for the 75% metric are not statistically significant and more data is needed to evaluate which model is most effective in this regard.

Overall, The precision and F1-score of the *Da-TRASH* pipeline significantly improved upon the Only-RGB method. Additionally, one can see by combining RGB feature embeddings with the predicted depth embeddings, the segmentation F1-scores and precisions are improved in most cases.

When *Da-TRASH* is compared to the RGB-only pipeline, it can be seen from Fig. 8, that for RGB-Only, the resulting feature maps are much more blurry. Consequently, the RGB-Only segmentation is much worse when compared to the segmentation produced by *Da-TRASH* pipeline. This comparison illustrates the efficacy of using *Da-TRASH* in instances when depth images are unavailable.

B. ZeroWaste Results

We were able to get promising results by running *Da-TRASH* pipeline on the ZeroWaste dataset, both in cases of sparse and dense distributions of trash as shown in Fig. 9 and 10. We tried to generate useful metrics like precision and recall, however, due to the limitations of the ZeroWaste dataset, we were unable to do so. The ZeroWaste dataset only contains ground-truth information about material classification and not ground-truth information about instance segmentation, so any metrics calculated from those would have been inaccurate. Nevertheless, the qualitative results we generated looked encouraging.

C. Preprocessing Depth

Our team hypothesized that our prediction boundaries could be made more accurate by discretizing the values of the depth map outputted by *ZoeDepth*. In theory, this would make clearer distinctions as to what objects were in the foreground/background, hopefully allowing the algorithm to more easily infer the boundaries of different objects. We propose the use of a k-means clustering algorithm for this task. The output of this algorithm is depicted in Fig. 11 where the image to the right is the discretized image where $k = 4$.

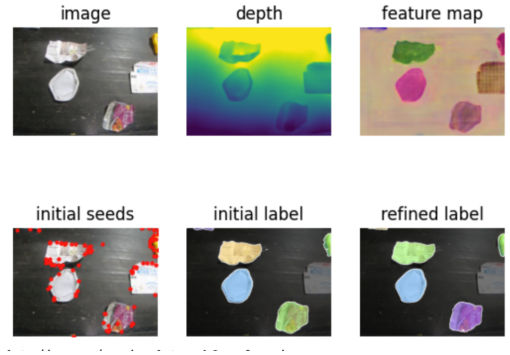

Fig. 10. Instance segmentation of a sparse scene from ZeroWaste

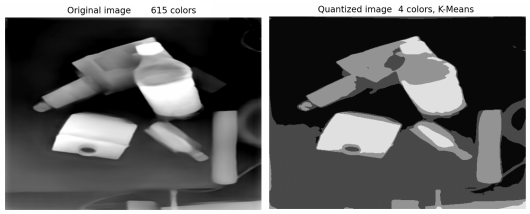

Fig. 11. Depiction of the output of the k-means clustering on depth data

This discretized depth image can directly be backprojected into x, y, z space using the camera intrinsics. This creates a nice, discrete pointcloud that can be visualized in Fig. 12.

Fig. 12. Discrete point cloud partial visualization with $k = 4$ (only k=1 and k=2 are shown.)

Unfortunately, through testing, we were not able to get the discretized depth images to perform anywhere near at the level of the standard predicted depth image. Given more computational resources, a new model could be trained from scratch using discretized depth which may yield better results.

VIII. Conclusion

Unseen Object Instance Segmentation is a huge problem in the field of computer vision and robotics. *Da-TRASH* leverages state-of-the-art depth estimation technologies to segment waste images, a notoriously difficult task to accomplish due to waste's variable qualities like shape, opacity, and color. *Da-TRASH's* ability to be used with just RGB images, while improving upon the results, makes *Da-TRASH* extremely versatile and handy to use.

References

[1] Y. Xiang, C. Xie, A. Mousavian, and D. Fox, "Learning rgb-d feature embeddings for unseen object instance segmentation," in *Conference on Robot Learning (CoRL)*, 2020.

[2] L. Shao, Y. Tian, and J. Bohg, "Clusternet: 3d instance segmentation in rgb-d images," Sep 2018. [Online]. Available: https://arxiv.org/abs/1807.08894

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," Jun 2015. [Online]. Available: https://arxiv.org/abs/1503.03832

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Dec 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[5] A. Richtsfeld, T. Mörwald, J. Prankl, M. Zillich, and M. Vincze, "Segmentation of unknown objects in indoor environments," 10 2012, pp. 4791–4796.

[6] Z. Z. J. A. F. A. P. H. V. A. B. C. S. A. B. Dina Bashkirova, Mohamed Abdelfattah and K. Saenko, "Zerowaste dataset: Towards deformable object segmentation in cluttered scenes," 2022.

[7] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," 2023. [Online]. Available: https://arxiv.org/abs/2302.12288