3D Object Recognition By Corresponding and Quantizing Neural 3D Scene Representations

Mihir Prabhudesai*, Shamit Lal*, Hsiao-Yu Fish Tung,
Adam W. Harley, Shubhankar Potdar, Katerina Fragkiadaki

Carnegie Mellon University
{mprabhud, shamitl, htung, aharley, smpotdar, katef}@cs.cmu.edu

Abstract

We propose a system that learns to detect objects and infer their 3D poses in RGB-D images. Many existing systems can identify objects and infer 3D poses, but they heavily rely on human labels and 3D annotations. The challenge here is to achieve this without relying on strong supervision signals. To address this challenge, we propose a model that maps RGB-D images to a set of 3D visual feature maps in a differentiable fully-convolutional manner, supervised by predicting views. The 3D feature maps correspond to a featurization of the 3D world scene depicted in the images. The object 3D feature representations are invariant to camera viewpoint changes or zooms, which means feature matching can identify similar objects under different camera viewpoints. We can compare the 3D feature maps of two objects by searching alignment across scales and 3D rotations, and, as a result of the operation, we can estimate pose and scale changes without the need for 3D pose annotations. We cluster object feature maps into a set of 3D prototypes that represent familiar objects in canonical scales and orientations. We then parse images by inferring the prototype identity and 3D pose for each detected object. We compare our method to numerous baselines that do not learn 3D feature visual representations or do not attempt to correspond features across scenes, and outperform them by a large margin in the tasks of object retrieval and object pose estimation. Thanks to the 3D nature of the object-centric feature maps, the visual similarity cues are invariant to 3D pose changes or small scale changes, which gives our method an advantage over 2D and 1D methods.

1 Introduction

The goal of this paper is detecting objects and inferring their 3D poses in RGBD images, with minimal human supervision. The ability to recognize objects under varying poses, sizes, lighting conditions, and camera viewpoints is fundamental for humans and other animals to track and interact with diverse objects. While humans and animals acquire this ability through evolution and interacting with the world under a moving visual sensor—their eyes—, most existing computer vision models are trained from labelled images, acquired from stylized camera viewpoints (He et al. 2017; Tulsiani et al. 2017a).

Recognizing familiar objects and detecting their 3D locations, poses and scales in images without 3D annotations remains elusive. In robotics, many works assume a closed world of predefined 3D object models, e.g., 3D object meshes, instead of discovering those from images (Narayanan and Likhachev 2017), and the fitting of the models to images is trained mostly supervised (Sundermeyer et al. 2018; Manhardt et al. 2018; Sucar, Wada, and Davison 2020). Few-shot object detection methods (Koch, Zemel, and Salakhutdinov 2015; Vinyals et al. 2016; Snell, Swersky, and Zemel 2017) use a support sample to quickly classify a query sample, but remain in 2D image space and do not infer 3D object orientation, rather object label.

Our key intuition in this work is to represent objects in terms of **3D feature representations** inferred from the input RGBD images, and infer alignment between two objects by explicitly rotating and scaling their representations during matching. While current state-of-the-art (SOTA) models for object detection and pose estimation represent an object as a feature vector or 2D feature maps (Rad, Oberweger, and Lepetit 2018; Mehta et al. 2018; He et al. 2017), our model represents objects as a 3D feature representation inferred from 2.5D (RGBD) input images, which can be explicitly scaled, rotated and compared in 3D. Different from methods in robotics research that infer explicit 3D geometry of an object in terms of meshes or poinclouds from multi-view data (Narayanan and Likhachev 2017; ten Pas and Platt 2018; Pinto and Gupta 2016) and depend heavily on a sufficient number of views, our model learns to infer the 3D object feature representation from a single view upon self-training.

We propose 3D quantized-Networks (3DQ-Nets), a model that can detect objects in 3D and that can iteratively establish accurate object correspondences without human labels or 3D annotations. We initialize its feature representations by pre-training on self-supervised view prediction task (Tung, Cheng, and Fragkiadaki 2019). To predict views, our model maps 2.5D images to 3D feature maps and project those to novel viewpoints to predict 2.5D alternative views. This task is unsupervised, since to collect the data for training, we only need to put in the scene a moving agent that can freely move and observe the scene from varying viewpoint. The inferred 3D visual feature map is view-invariant and thus unaffected by image variations caused by changes in camera viewpoint. In other words, an object will have the same

*Equal contribution
Project page:
https://mihirp1998.github.io/project_pages/3dq/

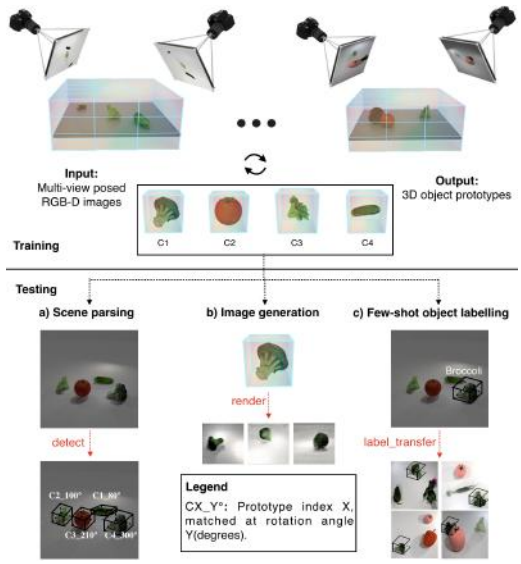

Figure 1: **Top: Model overview.** Our model takes as input posed RGB-D images of scenes, and outputs 3D prototypes of the objects. **Bottom: Evaluation tasks.** (a) Scene parsing: Given a new scene, we match each detected object against the prototypes using a rotation-aware check to infer its identity and pose. (b) Image generation: We visualize prototypes with a pre-trained 3D-to-2D image renderer. (c) Few shot object labelling: Assigning a label to a prototype automatically transfers this label to its assigned instances.

representation when viewed from different camera distances and angles. 3DQ-Nets further improve the features through automated cross-scene correspondence mining. The step is critical for establishing more accurate correspondence between objects. Our model cluster objects in a pose-aware manner into several clusters of similar-looking objects. We call the learned cluster centers *prototypes*, since they correspond to aggregates of object instances across 3D poses and scales. Given a scene, our model learns to parse the scene in terms of objects associated to prototype identities and their corresponding 3D poses (see Figure 1 (a)). The learned prototypes can be explicitly rotated, and can be rendered into images through a learned neural decoder (see Figure 1 (b)). We demonstrate the usefulness of our framework in few-shot learning: our model can recognize and name objects from one or a few samples (see Figure 1 (c)). Once given a labelled instance, the model propagates the label to all the instances in the same cluster.

Whether the model can infer correct correspondence from the object-centric 3D feature representation depends on the quality of two key components: the learned visual features and the 3D object detector. The weights of the encoder, decoder, 3D object detector, and prototypes are optimized using a mix of end-to-end backpropagation and expectation-maximization (EM) steps, and we show 3D object detection and prototype learning improve over time and help one another.

We empirically show that the modules of our model benefit one another and are essential for learning to recognize objects and their 3D orientation without supervision: the 3D object detector benefits from 3D visual prototypes by discarding bounding boxes not matching to prototypes; learning better object detection results in more accurate inference of finding object correspondences; better inferred object correspondences result in better learning of visual feature representations; and better visual feature improves clustering by inferring accurate pose-equivariant alignment of objects to prototypes.

We test our model in diverse environments including photo-realistic simulators and real world videos captured by a Kinect camera. We empirically show our model can effective learn to name new objects in a few-shot setting by propagating provided labels through the learned clusters. Our model outperforms by a large margin numerous baselines that do not infer a 3D feature space, rather, detect and cluster objects in a 2D feature space using CNN feature representations pretrained on ImageNet and finetuned with the few supplied labels, or do not mine cross-scene correspondences. We ablate each module of the proposed model and quantify its contribution in the performance of our full model.

The main contribution of this work is matching objects in a 3D-aware representation space inferred from images, supervised by view prediction and automated correspondence mining, without any 3D annotations. Objects are clustered into 3D prototypes which form then the basis for recognition: prototype identity inference and 3D pose with respect to the prototype's orientation. To the best of our knowledge, this is the first system that demonstrates that pose-aware 3D object recognition emerges without any 3D annotations in RGB-D images. Our code will be made available upon publication.

2 Related work

Self-supervised visual representation learning using pretext tasks. Self-supervising visual feature representation with a variety of pretext tasks has shown to deliver useful visual representations for downstream visual recognition tasks. Pretext tasks that have been considered are predicting views of static scenes (Eslami et al. 2018; Tung, Cheng, and Fragkiadaki 2019; Harley et al. 2020), predicting frame ordering (Lee et al. 2017), predicting spatial context (Doersch, Gupta, and Efros 2015; Pathak et al. 2016), predicting color of grayscale images (Zhang, Isola, and Efros 2016), predicting color of future video frames (Vondrick et al. 2018), predicting egomotion (Jayaraman and Grauman 2015; Agrawal, Carreira, and Malik 2015), and many others. Our 2D-to-3D image encoder builds upon works that train 3D visual representations—instead of 2D—using view regression and contrastive view prediction as the pretext tasks (Tung, Cheng, and Fragkiadaki 2019; Harley et al. 2020). While Harley et al. (2020) demonstrates the usefulness of such pretraining for 3D object detection, our work shows we can use the learned features to detect and associate objects, and infer 3D poses between objects. The work of Florence, Manuelli, and Tedrake (2018) used intra-scene correspondences provided by triangulation to train 2D CNNs for point

feature matching. Pot, Toshev, and Kosecka (2018) uses supervision from depth, egomotion and a vanilla 2D object detector to collect multiview images of the same object and learns 2D feature representations that cluster into discrete object identities. We consider a supervision setup similar to Pot, Toshev, and Kosecka (2018), but we pursue 3D feature representations. We learn 3D object detection and pose estimation of objects, as opposed to solely 2D object detection.

Inverse graphics, analysis-by-synthesis. Approaches on inverse graphics or analysis-by-synthesis attempt to map 2D images to complete 3D scene representations in terms of object 3D meshes, camera pose and scene layout (Kulkarni et al. 2015; Romaszko et al. 2017; Izadinia, Shan, and Seitz 2016). Many works show they can recover parametrized 3D meshes or binary voxel occupancies of objects from videos and scenes (Tulsiani et al. 2017b; Novotný, Larlus, and Vedaldi 2017; Wu et al. 2018; Tung et al. 2017a) by unsupervised rendering and matching to input depth maps. Our work differs in that we pursue feature-based 3D representations instead. As such, we do not need to know a low parametric model of the object mesh ahead of time as in recent works (Tung et al. 2017b; Kulkarni et al. 2015), and we do not need a predefined set of 3D object shapes as in recent works (Romaszko et al. 2017; Izadinia, Shan, and Seitz 2016).

Few-Shot Object Recognition. Existing work has proposed models that can learn to detect new objects with one or a few samples. However, these models cannot estimate 3D pose. Metric-based few-shot learning approaches (Koch, Zemel, and Salakhutdinov 2015; Vinyals et al. 2016; Snell, Swersky, and Zemel 2017) learn an embedding space in which objects of the same category are clustered together in the latent space. After training, the model can infer the most similar instance from the support samples by comparing instances in the learned embedding space. However, there is no obvious method to directly use the embedding space to infer the relative object poses between the query instance and the support sample. Moreover, the learning of these models also relies on human labels: all approaches require standard few-shot learning dataset which consists of multiple sub-groups of images that are labelled as "belonging to the same category". The recently proposed method of Tian et al. (2020) learns a classifier on top of supervised or self-supervised representations with few labels, and this outperforms previous few-shot learning approaches, but again it remains unclear how we can use the learned representations or the classifier to infer relative object poses.

Self-paced learning. Many techniques in semi-supervised or unsupervised visual learning iterate between pseudo-label inference and classifier/feature update using the inferred labels, in an Expectation-Maximization (EM) style algorithm (Soviany et al. 2019; Zou et al. 2018; Xie, Girshick, and Farhadi 2015; Shen, Efros, and Aubry 2019), yet existing work focus on improving 2D detection without considering detection in 3D. For example, successful recent methods for domain adaptation (Soviany et al. 2019; Zou et al. 2018) iterate between pseudo pixel label inference in the target domain and updating the pixel labellers. These methods show the classifiers or detectors can improve without drifting. While our work self-infers cross-scene 3D correspondences to improve the features and infers pseudo 3D box labels to improve the 3D object detector, previous works operating in 2D image space do not consider this.

3 3D Quantized-Networks (3DQ-Nets)

We depict the architecture of our model in Fig. 2. Given a set of posed RGB-D images of a static scene, our model constructs a 3D scene feature representation by neurally lifting and registering features extracted from each frame using geometry-aware inverse graphics networks (GIGNs) (Sec. 3.1). Our model detects objects in the inferred 3D scene representation (Sec. 3.1) and matches the 3D object feature tensors against a set of 3D prototypes by searching over 3D rotations (Sec. 3.2). Concurrently, our model uses the detected 3D boxes to improve the 3D visual feature representation by iteratively inferring 3D part correspondences across objects detected in different scenes, and using metric learning to supervise the feature representation to reinforce the inferred correspondences (Sec. 3.3).

Our model iteratively optimizes over weights of the encoder, decoder, 3D detector module and prototypes, and uses individual modules to bootstrap the learning of the others. We pretrain the weights of the encoder and decoder of GIGNs by view prediction. We detail each module in their respective section and present the learning of the model in Sec. 3.4.

3.1 2.5D-to-3D lifting using Geometry-aware Inverse Graphics Networks (GIGNs)

Geometry-aware Inverse Graphics Networks (GIGNs) (Tung, Cheng, and Fragkiadaki 2019; Harley et al. 2020) "lift" RGB-D images of static world scenes to 3D scene feature maps. The networks can be optimized end-to-end for a downstream task, such as supervised 3D object detection or unsupervised view prediction. To obtain the 3D scene feature maps, GIGNs are equipped with a differentiable 2D-to-3D inverse projection operation that can transform 2D feature maps into 3D feature maps. We will denote the 3D feature map inferred from an input RGB-D image I as $\mathbf{M} = \text{Enc}(I) \in \mathbb{R}^{w \times h \times d \times c}$ where w, h, d, c denote the width, height, depth and number of channels, respectively. Our experiments use $(w, h, d, c) = (72, 72, 72, 32)$. GIGNs explicitly rotate and translate the feature maps inferred from different RGB-D views using their corresponding ground truth camera poses. As a result, feature maps from different views are all aligned to a common coordinate system.

3D feature learning by predicting views We pre-train the encoder and decoder of GIGNs by predicting views using our posed RGB-D multiview image set.

Following the work of (Tung, Cheng, and Fragkiadaki 2019; Harley et al. 2020), we train GIGNs to predict a query view given a single view input, which enforces the model to complete the missing or occluded information from the image. Specifically, to predict a novel view, the scene feature map $\mathbf{M}$ is oriented to a sampled query viewpoint v_q and decoded to an RGB image and occupancy grid, and then com-

pared with the ground truth RGB (I_q) and occupancy (O_q) respectively:

$$\begin{aligned}\mathcal{L}^v =& \|\mathrm{Dec}^{\mathrm{RGB}}(\mathbf{M}, v_q) - I_q\|_1 \\ &+ \log(1 + \exp(-O_q \cdot \mathrm{Dec}^{\mathrm{occ}}(\mathbf{M}, v_q))),\end{aligned} \tag{1}$$

The RGB output is trained with a regression loss, and the occupancy is trained with a logistic classification loss. Occupancy labels are computed through raycasting, similar to Harley et al. (2020). Please refer the supplementary material for more details.

3D object detection A 3D detector operates on the output of the geometric encoder Enc and predicts a variable number of object boxes with associated confidences: $\mathcal{O} = \mathrm{Det}(\mathbf{M}) \in \{(\hat{b}^o_{loc}, c^o) | \hat{b}^o_{loc} \in \mathbb{R}^6, c^o \in [0, 1]\}$. We follow the architecture used in the work of Tung, Cheng, and Fragkiadaki (2019) for our detector. We provide our detector with a "warm start" by pre-training it with 3D box annotations computed from triangulated 2D category-agnostic proposals from a publicly-available 2D objectness detector (Wu et al. 2019). A detector trained with noisy annotations obtained from triangulation is expected to perform poorly, but it is sufficient for our system to start learning something useful. In Sec. 3.4 we describe our method for self-training the detector, so that it gradually learns to outperform its initialization.

3.2 Quantizing objects into prototypes

Our model learns a set of 3-dimensional prototypes $\mathbf{e}_k \in \mathbb{R}^{w_p \times h_p \times d_p \times c}, k \in \mathcal{K} = \{1, \ldots, K\}$ by clustering cropped 3D feature maps. Each prototype represents a set of similar objects. The prototype serves as the cluster center of the set. To learn them, our model clusters objects in the scene in a pose-equivariant and scale-equivariant manner: similar object instances that vary in scale and pose are mapped to the same prototype. We crop the 3D scene feature map $\mathbf{M}$ given a detected box to obtain object 3D feature tensors, and resize it to match the common size of the 3D prototypes $\mathbf{M}^o = \mathrm{resize}(\mathrm{crop}(\mathbf{M}, b^o), [w_p, h_p, d_p])$. Our experiments use $(w_p, h_p, d_p) = (16, 16, 16)$. We match detected objects' 3D feature tensors to prototypes using a rotation-aware feature matching. Specifically, we exhaustively search across rotations $\mathcal{R}$, in a parallel manner, considering increments of $10°$ along the vertical axis:

$$(z^o_{id}, z^o_R) = \underset{k \in \mathcal{K}, \mathrm{R} \in \mathcal{R}}{\mathrm{argmin}} \|\mathbf{e}_k - \mathrm{Rot}\left(\mathbf{M}^o, \mathrm{R}\right)\|_2, \forall o \in \mathcal{O}, \tag{2}$$

where $\mathrm{Rot}\left(\mathbf{M}, \mathrm{R}\right)$ explicitly rotates the content in feature map $\mathbf{M}$ with angle R through trilinear interpolation. Having assigned objects to oriented prototypes, we update our prototypes to minimize their Euclidean distance to the assigned oriented and scaled object tensors:

$$\mathcal{L}^{3DQ}(\mathbf{e}) = \sum_{o=1}^{|\mathcal{O}|} \|\mathbf{e}_{z^o_{id}} - \mathrm{Rot}(\mathbf{M}^o, z^o_R)\|_2 \tag{3}$$

We initialize our prototype dictionary with a set of exemplars. To ensure prototype diversity at this initial stage, we build the dictionary incrementally, and only use an exemplar as a prototype if its feature distance to the already-initialized prototypes is higher than a threshold. Equations 2 and 3 can be seen as expectation maximization steps iterating between exemplars-to-prototypes assignment and prototype updates.

3.3 Cross-scene 3D correspondence mining

Whether the model can establish the correct correspondence between objects and learn meaningful clusters relies on the quality of the visual features. To improve the visual features our model exploits visual similarity not only within scenes, but also across scenes. While the view prediction objective of Eq. 1 exploits different views of the *same* scene to learn the features, our model further exploits part-based correspondence between objects in *different* scenes to further improve the learned features. We adopt the correspondence mining method of ArtMiner (Shen, Efros, and Aubry 2019) to operate in 3D as opposed to 2D: Part based correspondences are hypothesized within detected objects and are verified by voting of their surrounding context voxels. If the original match is verified, hard-positive matches are then suggested in the surrounding of the match. Using the mined hard positive matches and randomly sampled negatives, we finetune the weights of our encoder Enc using metric learning. We empirically found that training with such cross-scene part-based correspondences helps improve the features. We provide the implementation details in the supplementary material.

3.4 Iterative learning of object detection, visual features, and clustering

Since the initialized object detector is sub-optimal due to the lack of groundtruth 3D boxes and can affect the rest of the modules, it is critical that we have a mechanism to improve it over time. To achieve this, we iterate our model over the following steps: (i) 3D object detection (Section 3.1). This generates a set of 3D object proposals. (ii) Cross-scene object part correspondence mining and learning (Section 3.3). This updates the encoder weights Enc using metric learning on inferred cross-scene correspondence on the detected objects. (iii) Prototype update (Section 3.2). This assigns detected object instances to prototypes and updates the prototypes $\mathbf{e}$ by backpropagating the clustering loss in Eq. 3. (iv) Object detector update. We label 3D object proposals as positives or negatives using a combination of 3D center-surround saliency score and matching to prototypes score. After the object detector is updated, we can iterate from step one to improve the rest of the modules.

Specifically, we keep the 3D object proposals that have a good matching score against the learned prototypes and discard the 3D object proposals whose 3D center-surround feature match score is below a threshold. The intuition is trust detection that either detects something that occurs often or has high saliency score. Center-surround saliency heuristic is used by numerous works for 2D and 3D object detection (Klein and Frintrop 2011; Ju et al. 2015). We then train the 3D object detector module to emulate such labels through standard gradient descent. In Fig. 4-(a), we visualize the self

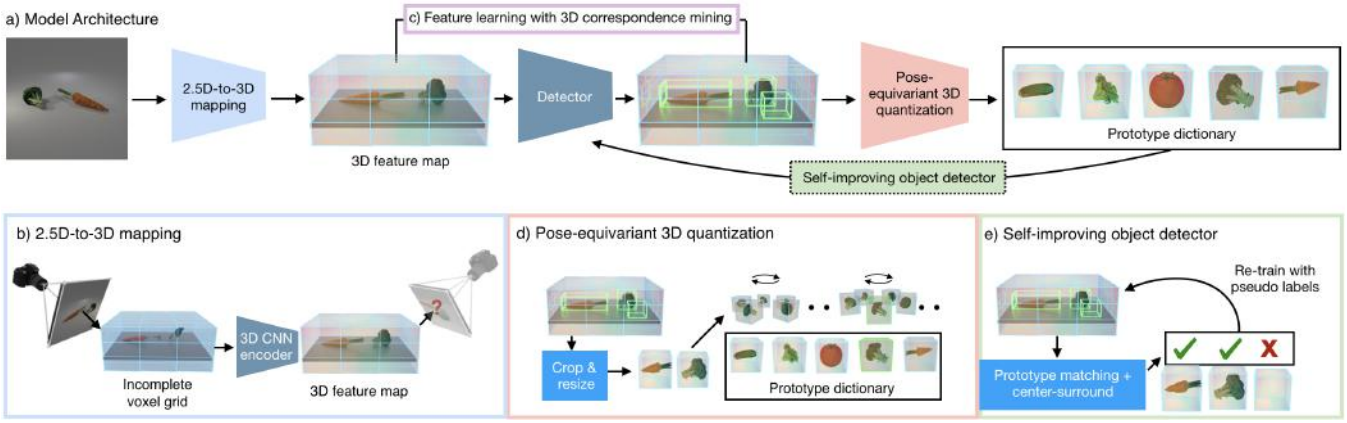

Figure 2: Architecture for **3D Quantized-Networks (3DQ-Nets)**. Given multi-view posed RGB-D images of scenes as input during training, our model learns to map a single RGB-D image to a completed scene 3D feature map at test time, by training for view prediction (b). The model additionally uses cross-scene and cross-object 3D correspondence mining and metric learning, to make the features more discriminative (c). Finally, using these learned features, our model quantizes object instances into a set of pose-canonical 3D prototypes using rotation-aware matching (d). These learned prototypes help improve our object detector by providing confident positive 3D object box labels (e) .

annotations and improvement made by our self-improving detector over 4 iterations.

4 Experiments

We test our framework in a variety of simulated environments and real world scenes. In simulation, RGB, depth and egomotion are provided by the simulator, whereas in the real world, RGB and depth are provided by Kinect sensors and egomotion is computed using camera calibration. Our experiments aim to answer the following questions:

1. Do 3DQ-Nets recognize objects better than CNN models pretrained on large labelled image datasets?
2. How does the proposed pose-aware 3D clustering compare against 2.5D pose-aware clustering, 3D pose-unaware clustering, or raw 3D point cloud registration?
3. Does cross-scene 3D correspondence mining improves features over view-predictive training, and how much?
4. In 3DQ-Nets, do feature learning, object clustering to prototypes, and 3D object detection improve over training iterations?

We benchmark our model on three datasets: (i) **CLEVR veggie** dataset: we build upon the CLEVR dataset (Johnson et al. 2017) and add 17 vegetable object models bought from Turbosquid. (ii) **CARLA** dataset: we created scenes using all 26 vehicle categories available in the CARLA simulator of Dosovitskiy *et al.* (Dosovitskiy et al. 2017) (iii) **BigBIRD (Singh et al. 2014)**: a publicly available dataset that contains multiview shots for 125 different objects rotating on a table. We assign the objects to 41 different object categories, combining similar objects into a single category.

We further qualitatively evaluate our model on two datasets: (iv) **Replica (Straub et al. 2019)** dataset: we render images from the indoor meshes provided by Replica in AI Habitat simulator (Savva et al. 2019). The views are selected by moving the agent around randomly selected objects. (v) **Real world desk scenes dataset**: training setup consists of 8 Kinect sensors surrounding the table to capture multiview RGB-D data. During test time, we only use a single Kinect sensor. More details on our dataset collection are included in the supplementary material.

4.1 Few-shot object category labelling

In this experiment, we use ground-truth 3D bounding boxes during training of our model to isolate errors caused by the 3D object detection module. Out task is to classify object-centric image crops into object categories, when supplied with only two labelled object-image crop per category. This means, that e.g., in the CARLA dataset, we use 52 labelled object image crops. Note, the objects can be at any orientation. We evaluate the ability of our model and baselines to retrieve objects of the same category when supplied with these few labelled examples.

Given an annotated instance, our model finds the prototype that has minimum rotation-aware feature distance to the object instance, and it propagates the label to all the instances that are assigned to the same prototype. If a prototype is matched with more than one label, then the label which has matched the most is assigned to the prototype. Note that the small labelled set is not used to update our features or prototypes. In Table 1, we compare 3DQ-Nets against two 2D baseline models using pretrained ResNet-18 on ImageNet as their backbone: (i) Finetuning the top layer of ResNet-18 with our training examples (ResNetClass), (ii) using the top average pool layer activations of ResNet-18 to retrieve and copy the label of the nearest neighbor instance from the training examples (ResNetRet), i.e., not finetuning at all the weights. We show the results in Table 1. Our model outperforms both ResNetClass and ResNetRet. Despite the fact the ResNet features are pre-trained on a large

Datasets.	ResNetRet	ResNetClass	3DQ-Nets
CARLA	0.27	0.58	**0.71**
CLEVR	**0.80**	0.72	0.75
BigBIRD	0.40	0.67	**0.82**

Table 1: **Few shot object category labelling accuracy**

set of annotated images, our model can self-adapt in the new domain of each dataset, and thus learn more meaningful object distances, captured in the inferred 3D feature representations. On CLEVR-veggie dataset, ResNetRet performs slightly better than 3DQ-Nets. We suspect this is because the object categories in CLEVR-veggie appear in ImageNet, so the ImageNet pertaining likely provides discriminative features for these objects.

4.2 Clustering with 3D pose-aware quantization

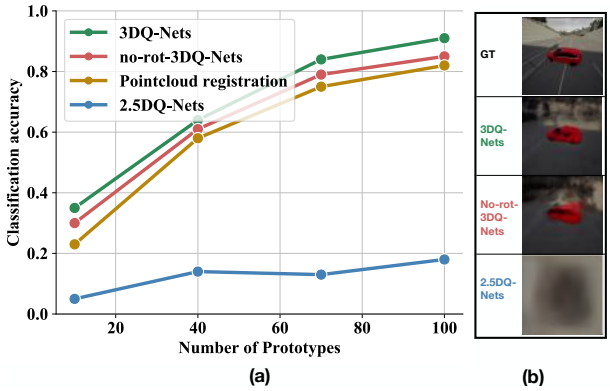

Figure 3: **(a) Unsupervised classification accuracy** with varying length of prototype dictionary in CARLA. **(b) Scene reconstruction** results using the learned prototypes from our model and the baselines.

In this experiment, we evaluate the importance of 3D pose-aware quantization for 3D object clustering. We compare our model against three baselines: (i) 2.5DQ-Nets, a 2D CNN model that takes concatenated RGB and depth as input and quantizes detected 2D image patches into a discrete set of 2D prototypes by optimizing an autoencoding objective. During quantization, the model conducts 2D rotation search. (ii) no-rot-3DQ-Nets, a model similar to ours except that it assigns instances to 3D prototypes without rotation search. (iii) Pointcloud registration (Mitra et al. 2004), a method that uses registered point clouds as prototypes and conducts 3D rotation aware search to infer the identity of the closest 3D poincloud prototype and the 3D pose of the object instance with respect to the prototype. For our model and baselines we consider ground-truth 3D and 2D object boxes to isolate the error from different detectors.

To evaluate the unsupervised classification accuracy using prototypes, we use LIN-MATCH, a bipartite graph matching method (Kuhn 1955), that finds the permutation of prototype indices that minimizes the classification error. We show

Datasets	ResNet	rgbocc	rgbocc+VC	rgbocc+VC*	ours
CARLA	0.49	0.62	0.55	0.67	**0.80**
CLEVR	**0.87**	0.74	0.71	0.74	0.81
BigBIRD	0.47	0.44	0.69	**0.77**	0.73

Table 2: **Retrieval results (precision@10 nearest neighbors)** for different architectures and objectives for 2D and 3D visual representation learning.

these comparisons with varying length of the prototype dictionary in Figure 3 (a). We see in Figure 3 (a) that models that use 3D representation achieve significantly higher accuracy compared to models using 2D representation. Further adding rotation search in 3D during clustering improves the performance since the operation enforces objects with similar appearance but with different poses to be clustered together. We also show that being able to inpaint objects from a single view during inference helps our model in outperforming the Pointcloud registration baseline that needs to handle incomplete input object-centric poinclouds. In Figure 3 (b) we show the scene reconstruction results of our models and the neural baselines after replacing the object in the scene with its best matched prototype under the inferred pose and rendering the 3D feature map through the learned decoder. Please refer the supplementary material for more details on the baselines and results on other datasets.

4.3 3D feature learning with 3D correspondence mining

In this experiment, we evaluate the contribution of 3D mining in feature learning, by evaluating the features in object category few shot retrieval. We compare it against the following feature learning methods: (i) Resnet-18 pretrained on Imagenet dataset (ResNet), where we average-pool features within the projected (ground-truth) 2D object boxes to represent the objects. (ii) GIGNs trained with RGB view and occupancy prediction (rgb-occ) of (Tung, Cheng, and Fragkiadaki 2019). (iii) GIGNs trained with object-centric view contrastive prediction (rgbocc+VC) of (Harley et al. 2020). (iv) We improve (iii) by using the same metric learning loss function (He et al. 2019) as our model (rgbocc+VC*). (v) GIGNs trained additionally with cross-scene 3D mining (ours). For (ii),(iii),(iv),(v), we use the cropped 3D feature maps from 3D object boxes to represent the objects. We randomly sample 1000 objects and retrieve their nearest neighbors by considering the maximum inner product across 36 rotations against a pool of another 1000 objects. For (i), we consider 2D rotation search as opposed to 3D. We show category-level retrieval precision within the first 10 retrieved nearest neighbors (i.e., precision@10) in Table 2.

As shown in Table 2, cross-scene correspondence mining improves the retrieval results. In the CLEVR dataset, ResNet outperforms our model. Our model performs the best among the unsupervised methods.

Task \ Iterations	Iter 0	Iter 1	Iter 2	Iter 3
Feature Learning	0.72	0.76	0.79	0.79
Quantization	0.51	0.63	0.65	0.66
Detection	0.43	0.48	0.51	0.52

Table 3: Performance across training EM iterations of our model in CLEVR. Feature learning is measured using the same technique as Table 2. Object quantization uses the same measurement technique as Fig. 3 (a). Detection performance is measured by meanAP at IoU $= 0.5$.

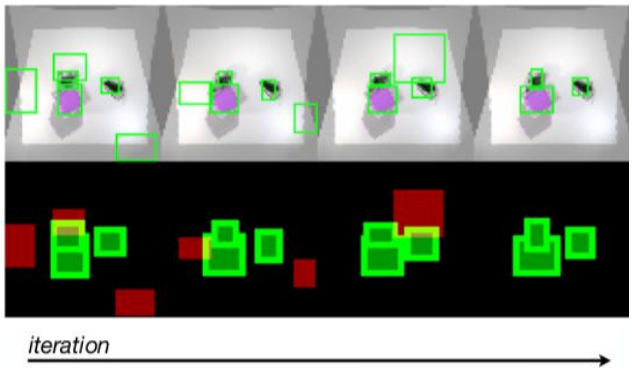

Figure 4: **Detection improvement over 4 iterations.** The first row shows the input image and the proposals of the object detector. The second row shows the annotations assigned to the proposals using the 3D prototype distance and 3D center-surround score. We show that our detector improves over time without any ground truth 3D proposals.

4.4 Joint training of 3D object detection, feature learning and clustering

Table 3 shows evaluations of our different modules during 4 iterations of EM. We see that the performance of all our modules improves over iterations. To initialize the weights for the modules (Iteration 0), we warm-start the 3D scene features using RGB view and occupancy prediction (rgbocc), and use the 3D object proposals provided by triangulated 2D boxes from 2D objectness detector to train the detector, visual features and prototypes. From Iteration 1 onwards, we use the 3D detected boxes from the trained detector as inputs, and use 3D mining to update the features. We subsequently improve the detector and the rest of the modules iteratively. We show that all modules can boostrap one another and continually improve over iterations. We further show our detector improvement over time in Figure 4.

4.5 Scene parsing using prototypes

Our learnt prototypes capture each object instance in its canonical pose. We use these prototypes for task of scene parsing. Given a new scene, we first detect all the objects and extract their features from the scene. Then, we match the object-centric feature maps with all the prototypes using a rotation aware similarity check explained in Section 3.2.

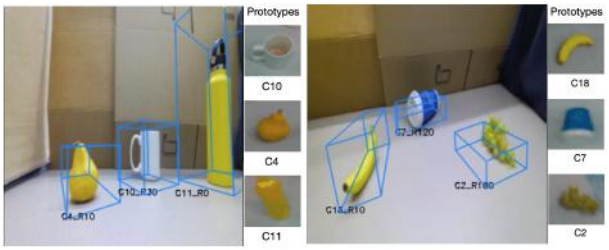

Figure 5: **Real world scene parsing results**. More results are available on the project webpage.

For each detected object instance in the scene, we visualize the matched prototype number (C) and the respective rotation angle along vertical axis (R) as seen in Figure 5. We also visualize the respective prototypes by neurally rendering them to images. Please refer supplementary material for the video on real world scene parsing. Results on Replica dataset are provided in the supplementary material.

4.6 Limitations

The presented framework currently has the following limitations: **(i)** Prototypes do not deform. A prototype is rotated and scaled, but not non-rigidly deformed. Adding such deformable parametrization (Kurenkov et al. 2017) would increase the expressiveness of the prototype dictionary. **(ii)** Prototypes cannot be stylized. Allowing prototypes to be stylized by changing their appearance using predicted stylization parameters (Huang and Belongie 2017), would again increase their expressiveness dramatically. **(iii)** The model cannot learn from videos of dynamic scenes, i.e., scenes with independently moving or deforming objects. Overcoming this limitation would require tracking the moving objects over time.

5 Conclusion

We presented a system that given multi-view posed 2.5D images learns to detect the objects in 3D and organizes them into a set of 3D prototypes in their canonical poses and scales. We applied our method to various datasets in simulation and in the real world. We demonstrated the usefulness of our framework in few shot learning, where prototypes propagate a small number of semantic labels to object instances. In that task, our model outperforms ImageNet-pretrained and 2.5D feature learning baselines. We further empirically showed that the modules of our framework improve over time, the proposed 3D prototypes are more expressive than 2.5D and registered point cloud equivalents, and 3D cross-scene correspondence mining dramatically improves retrieval accuracy, compared to view prediction objectives alone. Learning from videos of dynamic scenes and incorporating deformability and stylization during prototype matching and learning, as discussed in the limitations section, are clear avenues for future work.

References

Agrawal, P.; Carreira, J.; and Malik, J. 2015. Learning to See by Moving. *CoRR* abs/1505.01596. URL http://arxiv.org/abs/1505.01596.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised Visual Representation Learning by Context Prediction. *CoRR* abs/1505.05192. URL http://arxiv.org/abs/1505.05192.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An Open Urban Driving Simulator. In *CORL*, 1–16.

Eslami, S. M. A.; Jimenez Rezende, D.; Besse, F.; Viola, F.; Morcos, A. S.; Garnelo, M.; Ruderman, A.; Rusu, A. A.; Danihelka, I.; Gregor, K.; Reichert, D. P.; Buesing, L.; Weber, T.; Vinyals, O.; Rosenbaum, D.; Rabinowitz, N.; King, H.; Hillier, C.; Botvinick, M.; Wierstra, D.; Kavukcuoglu, K.; and Hassabis, D. 2018. Neural scene representation and rendering. *Science* 360(6394): 1204–1210. ISSN 0036-8075. doi:10.1126/science.aar6170.

Florence, P. R.; Manuelli, L.; and Tedrake, R. 2018. Dense Object Nets: Learning Dense Visual Object Descriptors By and For Robotic Manipulation. In Billard, A.; Dragan, A.; Peters, J.; and Morimoto, J., eds., *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, 373–385. PMLR. URL http://proceedings.mlr.press/v87/florence18a.html.

Harley, A. W.; Li, F.; Lakshmikanth, S. K.; Zhou, X.; Tung, H.-Y. F.; and Fragkiadaki, K. 2020. Learning from Unlabelled Videos Using Contrastive Predictive Neural 3D Mapping. In *ICLR*.

Hartley, R.; and Zisserman, A. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2019. Momentum Contrast for Unsupervised Visual Representation Learning.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. B. 2017. Mask R-CNN. *CoRR* abs/1703.06870. URL http://arxiv.org/abs/1703.06870.

https://www.turbosquid.com. 2020 .

http://wiki.ros.org/camera_calibration/. 2020 .

Huang, X.; and Belongie, S. J. 2017. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *CoRR* abs/1703.06868. URL http://arxiv.org/abs/1703.06868.

Izadinia, H.; Shan, Q.; and Seitz, S. M. 2016. IM2CAD. *CoRR* abs/1608.05137. URL http://arxiv.org/abs/1608.05137.

Jayaraman, D.; and Grauman, K. 2015. Learning image representations equivariant to ego-motion. *CoRR* abs/1505.02206. URL http://arxiv.org/abs/1505.02206.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Lawrence Zitnick, C.; and Girshick, R. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2901–2910.

Ju, R.; Liu, Y.; Ren, T.; Ge, L.; and Wu, G. 2015. Depth-aware salient object detection using anisotropic center-surround difference. *Signal Processing: Image Communication* 38: 115–126.

Klein, D. A.; and Frintrop, S. 2011. Center-surround divergence of feature statistics for salient object detection. In *2011 International Conference on Computer Vision*, 2214–2219. IEEE.

Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese Neural Networks for One-shot Image Recognition.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1-2): 83–97.

Kulkarni, T. D.; Kohli, P.; Tenenbaum, J. B.; and Mansinghka, V. 2015. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 4390–4399.

Kurenkov, A.; Ji, J.; Garg, A.; Mehta, V.; Gwak, J.; Choy, C. B.; and Savarese, S. 2017. DeformNet: Free-Form Deformation Network for 3D Shape Reconstruction from a Single Image. *CoRR* abs/1708.04672. URL http://arxiv.org/abs/1708.04672.

Lee, H.-Y.; Huang, J.-B.; Singh, M.; and Yang, M.-H. 2017. Unsupervised representation learning by sorting sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, 667–676.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Manhardt, F.; Kehl, W.; Navab, N.; and Tombari, F. 2018. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 800–815.

Mehta, D.; Sotnychenko, O.; Mueller, F.; Xu, W.; Sridhar, S.; Pons-Moll, G.; and Theobalt, C. 2018. Single-shot multi-person 3d pose estimation from monocular rgb. In *2018 International Conference on 3D Vision (3DV)*, 120–130. IEEE.

Mitra, N. J.; Gelfand, N.; Pottmann, H.; and Guibas, L. 2004. Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 22–31.

Narayanan, V.; and Likhachev, M. 2017. Deliberative object pose estimation in clutter. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3125–3130. IEEE.

Novotný, D.; Larlus, D.; and Vedaldi, A. 2017. Learning 3D Object Categories by Looking Around Them. *CoRR* abs/1705.03951. URL http://arxiv.org/abs/1705.03951.

Park, J.; Zhou, Q.-Y.; and Koltun, V. 2017. Colored point cloud registration revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, 143–152.

Pathak, D.; Krähenbühl, P.; Donahue, J.; Darrell, T.; and Efros, A. 2016. Context Encoders: Feature Learning by Inpainting. In *CVPR*.

Pinto, L.; and Gupta, A. 2016. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, 3406–3413. IEEE.

Pot, E.; Toshev, A.; and Kosecka, J. 2018. Self-supervisory Signals for Object Discovery and Detection.

Rad, M.; Oberweger, M.; and Lepetit, V. 2018. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4663–4672.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Romaszko, L.; Williams, C. K. I.; Moreno, P.; and Kohli, P. 2017. Vision-As-Inverse-Graphics: Obtaining a Rich 3D Explanation of a Scene From a Single Image. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*.

Savva, M.; Kadian, A.; Maksymets, O.; Zhao, Y.; Wijmans, E.; Jain, B.; Straub, J.; Liu, J.; Koltun, V.; Malik, J.; Parikh, D.; and Batra, D. 2019. Habitat: A Platform for Embodied AI Research. *CoRR* abs/1904.01201. URL http://arxiv.org/abs/1904.01201.

Shen, X.; Efros, A. A.; and Aubry, M. 2019. Discovering Visual Patterns in Art Collections With Spatially-Consistent Feature Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Singh, A.; Sha, J.; Narayan, K. S.; Achim, T.; and Abbeel, P. 2014. BigBIRD: A large-scale 3D database of object instances. *2014 IEEE International Conference on Robotics and Automation (ICRA)* 509–516.

Snell, J.; Swersky, K.; and Zemel, R. S. 2017. Prototypical Networks for Few-shot Learning. *CoRR* abs/1703.05175. URL http://arxiv.org/abs/1703.05175.

Soviany, P.; Ionescu, R. T.; Rota, P.; and Sebe, N. 2019. Curriculum Self-Paced Learning for Cross-Domain Object Detection.

Straub, J.; Whelan, T.; Ma, L.; Chen, Y.; Wijmans, E.; Green, S.; Engel, J. J.; Mur-Artal, R.; Ren, C.; Verma, S.; Clarkson, A.; Yan, M.; Budge, B.; Yan, Y.; Pan, X.; Yon, J.; Zou, Y.; Leon, K.; Carter, N.; Briales, J.; Gillingham, T.; Mueggler, E.; Pesqueira, L.; Savva, M.; Batra, D.; Strasdat, H. M.; Nardi, R. D.; Goesele, M.; Lovegrove, S.; and Newcombe, R. 2019. The Replica Dataset: A Digital Replica of Indoor Spaces. *arXiv preprint arXiv:1906.05797* .

Sucar, E.; Wada, K.; and Davison, A. 2020. Neural Object Descriptors for Multi-View Shape Reconstruction. *arXiv preprint arXiv:2004.04485* .

Sundermeyer, M.; Marton, Z.-C.; Durner, M.; Brucker, M.; and Triebel, R. 2018. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 699–715.

ten Pas, A.; and Platt, R. 2018. Using geometry to detect grasp poses in 3d point clouds. In *Robotics Research*, 307–324. Springer.

Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J.; and Isola, P. 2020. Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need? *ArXiv* abs/2003.11539.

Tulsiani, S.; Gupta, S.; Fouhey, D. F.; Efros, A. A.; and Malik, J. 2017a. Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene. *CoRR* abs/1712.01812. URL http://arxiv.org/abs/1712.01812.

Tulsiani, S.; Zhou, T.; Efros, A. A.; and Malik, J. 2017b. Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency. *CoRR* abs/1704.06254. URL http://arxiv.org/abs/1704.06254.

Tung, H. F.; Harley, A.; Seto, W.; and Fragkiadaki, K. 2017a. Adversarial Inverse Graphics Networks: Learning 2D-to-3D Lifting and Image-to-Image Translation with Unpaired Supervision. *ICCV* .

Tung, H. F.; Tung, W.; Yumer, E.; and Fragkiadaki, K. 2017b. Self-supervised learning of Motion Capture. *NIPS* .

Tung, H.-Y. F.; Cheng, R.; and Fragkiadaki, K. 2019. Learning Spatial Common Sense with Geometry-Aware Recurrent Networks. In *CVPR*.

Vinyals, O.; Blundell, C.; Lillicrap, T. P.; Kavukcuoglu, K.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. *CoRR* abs/1606.04080. URL http://arxiv.org/abs/1606.04080.

Vondrick, C.; Shrivastava, A.; Fathi, A.; Guadarrama, S.; and Murphy, K. 2018. Tracking Emerges by Colorizing Videos. *CoRR* abs/1806.09594. URL http://arxiv.org/abs/1806.09594.

Wu, J.; Xue, T.; Lim, J. J.; Tian, Y.; Tenenbaum, J. B.; Torralba, A.; and Freeman, W. T. 2018. 3D Interpreter Networks for Viewer-Centered Wireframe Modeling. *International Journal of Computer Vision (IJCV)* .

Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; and Girshick, R. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Xie, J.; Girshick, R. B.; and Farhadi, A. 2015. Unsupervised Deep Embedding for Clustering Analysis. *CoRR* abs/1511.06335. URL http://arxiv.org/abs/1511.06335.

Zhang, R.; Isola, P.; and Efros, A. A. 2016. Colorful Image Colorization. *CoRR* abs/1603.08511. URL http://arxiv.org/abs/1603.08511.

Zhou, Q.-Y.; Park, J.; and Koltun, V. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* .

Zou, Y.; Yu, Z.; Kumar, B. V. K. V.; and Wang, J. 2018. Domain Adaptation for Semantic Segmentation via Class-Balanced Self-Training. *CoRR* abs/1810.07911. URL http://arxiv.org/abs/1810.07911.

6 Appendix Overview

In Section 7, we provide details for all our datasets and their collection process. In Section 8, we provide implementation details for each of our modules and we also provide further implementation details of the baselines. In Section 9, we provide additional qualitative/quantitative results on Replica and other datasets.

7 Dataset preparation

CLEVR Dataset. We build upon the CLEVR Blender simulator (Johnson et al. 2017) and add 17 vegetable object models bought from Turbosquid (https://www.turbosquid.com 2020), in addition to the object models available in CLEVR. So in total our dataset has 41 unique object models. We consider each object model to be a separate object category, this information will be used for evaluation purposes, not at training time. We create scenes as follows: Each object model is randomly rotated (0° to 360° along vertical axis), translated (randomly within a sphere of radius 10.5 units) and scaled (0.75 to 1.25 times the actual size). Each scene contains up to 3 objects. We randomly vary the lightning of each scene.We render each scene by placing 28 RGB-D cameras at elevations ranging from 26° to 80° with 13° increments and azimuths ranging from 0° to 360° with 45° increments. Each camera is placed within a sphere of radius 1.5 metres from the center of the scene.

CARLA Dataset. Our CARLA dataset uses the 26 vehicle classes available in the CARLA simulator. We consider each vehicle model to be a separate object category, again this information will be used for evaluation purposes, not at training time. Each rendered scene consists of either one or two vehicles. Each scene in the training set consists of multi-view RGB and depth images of static vehicles placed at randomly selected spawn points. We generate scenes by randomly selecting a map from the available CARLA maps. Then we perturb the weather conditions randomly by setting cloudiness to a value in [0, 70], precipitation to be within [0, 75], and sun_altitude_angle to be within [30, 90]. For single-vehicle scenes, we randomly select a spawn point and place a vehicle at that spawn point. Then we place 17 RGB-D randomly cameras around the vehicle. The origin of the vehicle serves as the origin with respect to which the extrinsic matrices of all the cameras are calculated. For vehicles in CARLA, the x axis points forward, y axis points to the right and the z axis points upwards. We place the first eight RGB-D cameras on the boundary of a circle centered at the vehicle's origin with radius=3.4m ,height z=1.0m and with yaw angle varying from -40° to -285° with increments of -35° each. The next eight RGB-D cameras again follow the same setup but with z=3.0m. Finally, the last RGB-D camera is placed overhead with z=5.0m and pitch=-90°.
For two-vehicle scenes, we first place the first vehicle at a randomly selected spawn point. We then select another spawn point from nearby spawn points and position the second vehicle there. This is required so that we can have both vehicles in the field of view of majority of the cameras. The origin in two vehicle setup is taken to be the mean of the origins of the two vehicles. All camera extrinsic matrices are calculated with respect to this origin. We again randomly place 17 RGB-D cameras around the origin. The first camera is placed at x=4.5m, z=1m, and yaw=-180. The next seven cameras are placed on the boundary of a circle of radius 7.5m, height 5.5m, and centered at the origin. The yaw angle is varied from -40° to -285° (with the exception of -180°) with increments of -35°. The next seven cameras are placed on the boundary of a circle of radius 4.5m, height 6.5m, and centered at the origin. Each camera has pitch -40°. The yaw angle is varied from -40° to -285° with increments of -35°. The final camera is placed overhead with z=5m and pitch=-90°.

BigBIRD Dataset. BigBIRD dataset consists of 125 objects placed on a rotating table. We use the 'Raw-RGB-D' dataset provided by BigBIRD. The camera setup consists of 5 RGB-D cameras placed in an arc, with the first camera in front of the object and the last camera overhead. The cameras capture the RGB and depth images of the rotating object at every 3° interval. This gives us 600 RGB-D images for each object. For our use case, we treat the object as stationary and instead assume that there are cameras placed at every 3° interval capturing multi-view images of the static object. This setup results in 600 RGB-D cameras placed around a stationary object. We assign the 125 object classes to 41 different object classes, combining similar objects, e.g. clif_crunch_chocolate_chip and clif_crunch_peanut_butter, into a single class, thus satisfying our use-case. Note that category labels are used for evaluation purposes, not at training time.

Replica Dataset. We render our Replica dataset by loading each of the 18 3D indoor meshes provided by Replica (Straub et al. 2019) in AI Habitat (Savva et al. 2019). To generate scenes with objects, we pick some objects appearing in the vicinity to each other, move the agent around those objects and click 6 multi-view RGB-D images. For retrieval, compression, and detection tasks, we only consider objects which satisfy the following conditions: (1) They should be visible in at least 4 out of the 6 views. (2) The 2D bounding box for the object should have an area greater than 1000 $pixel^2$. (3) The ratio of the number of points occupied by an object in the semantic map to the area of the 2D bounding box for that object should be greater than 0.1. Our dataset collected from replica consists of 26 unqiue object categories.

Real world desk scenes dataset. This dataset consists of a set of 18 different objects placed on a table seen by a dome of 8 Microsoft Azure Kinect sensors. We know the intrinsics of the cameras and calculate the extrinsics by calibrating them using OpenCV's (http://wiki.ros.org/camera_calibration/ 2020) checkerboard caliberation technique. Since we cannot have annotations in the real world, we only show qualitative results for the model trained on this dataset. During test time, we collect RGB-D

images of multiple objects placed on a table by moving a single Microsoft Azure Kinect around the scene. To get the camera extrinsics at different points of time, we estimate the trajectory of the camera by calculating its rigid body transformations using consecutive RGB-D images. The transformation is estimated using the point cloud matching technique described in (Park, Zhou, and Koltun 2017). We utilise Open3D's (Zhou, Park, and Koltun 2018) implementation of (Park, Zhou, and Koltun 2017). Since the resolution of the RGB images and the depth maps are different, we create a RGB-D image by mapping the depth map to the RGB image to obtain the depth values for all the RGB pixels. This mapping is done using the linear interpolation module provided as part of the Kinect SDK.

8 Implementation Details

Code, training details and computation complexity. Our model is implemented in Python/Pytorch. We keep our batch size as 2, our learning rate is kept as 10^{-3} for view prediction training and is dropped it down to 10^{-4} when training for all further tasks. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$. Our model takes 24hrs (approx. 100k iterations) of training for convergence and requires 0.8 seconds for an inference step on a single V100 GPU.

Inputs. For all datasets, we resize input RGB images to a resolution of 256x256 pixels. We randomly select images from 2 views (query view and target view) of the multi-view scene as inputs for training our model, while we use a single view for testing.

Geometry-aware Inverse Graphics Networks(GIGNs) Our 2.5D-to-3D lifting, 3D occupancy estimation and 2D RGB estimation modules follow the exact same architecture as (Harley et al. 2020). We explain the implementation details of each of these modules below.

2.5D-to-3D lifting Our 2.5D-to-3D unprojection module takes as input RGB-D images and converts it into a 4D tensor $\mathbf{U} \in \mathbb{R}^{w \times h \times d \times 4}$, where w, h, d is $72, 72, 72$. We use perspective (un)projection to fill the 3D grid with samples from 2D image. Specifically, using pinhole camera model (Hartley and Zisserman 2003), we find the floating-point 2D pixel location that every cell in the 3D grid, indexed by the coordinate (i, j, k), projects onto from the current camera viewpoint. This is given by $[u, v]^T = \mathbb{K}\mathbb{S}[i, j, k]^T$, where $\mathbb{S}$, the similarity transform, converts memory coordinates to camera coordinates and $\mathbb{K}$, the camera intrinsics, convert camera coordinates to pixel coordinates. Bilinear interpolation is applied on pixel values to fill the grid cells. We obtain a binary occupancy grid $\mathbf{O} \in \mathbb{R}^{w \times h \times d \times 1}$ from the depth image $\mathbf{D}$ in a similar way. This occupancy is then concatenated with the unprojected RGB to get a tensor $[\mathbf{U}, \mathbf{O}] \in \mathbb{R}^{w \times h \times d \times 4}$. This tensor is then passed through a 3D encoder-decoder network, the architecture of which is as follows: 4-2-64, 4-2-128, 4-2-256, 4-0.5-128, 4-0.5-64, 1-1-F. Here, we use the notation k-s-c for kernel-stride-channels, and F is the feature dimension, which we set to $F = 32$. We concatenate the output of transposed convolutions in decoder with same resolution feature map output from the encoder. The concatenated tensor is then passed to the next layer in the decoder. We use leaky ReLU activation and batch normalization after every convolution layer, except for the last one in each network. We obtain our 3D feature map $\mathbf{M}$ as the output of this process.

3D occupancy estimation. In this step, we want to estimate whether a voxel in the 3D grid is "occupied" or "free". The input depth image gives us partial labels for this. We voxelize the pointcloud to get sparse "occupied" labels. All voxel cells that are intersected by the ray from the source-camera to each occupied voxel are marked as "free". We give $\mathbf{M}$ as input to the occupancy module. It produces a new tensor $\mathbf{C}$, where each voxel stores the probability of being occupied. We use a 3D convolution layer with a $1 \times 1 \times 1$ filter followed by a sigmoid non-linearity to achieve this. We train this network with the logistic loss, $\mathcal{L}_{\text{occ}} = (1/\sum \hat{\mathbf{I}}) \sum \hat{\mathbf{I}} \log(1 + \exp(-\hat{\mathbf{C}} \cdot \mathbf{C}))$, where $\hat{\mathbf{C}}$ is the label map, and $\hat{\mathbf{I}}$ is an indicator tensor, indicating which labels are valid. Since there are far more "free" voxels than "occupied", we balance this loss across classes within each minibatch.

2D RGB estimation. Given a camera viewpoint v_q, this module projects the 3D feature map $\mathbf{M}$ to "render" 2D feature maps. To achieve this, we first obtain a view-aligned version, $\mathbf{M}_{v_q}$, by resampling $\mathbf{M}$. The view oriented tensor, $\mathbf{M}_{v_q}$, is then warped so that perspective viewing rays become axis-aligned. This gives us the perspective-transformed tensor $\mathbf{M}_{proj_q}$. This tensor is then passed through a CNN to get a 2D feature map v_q. The CNN has the following architecture (using the notation k-s-c for kernel-stride-channels): max-pool along the depth axis with $1 \times 8 \times 1$ kernel and $1 \times 8 \times 1$ stride, to coarsely aggregate along each camera ray, 3D convolution with 3-1-32, reshape to place rays together with the channel axis, 2D convolution with 3-1-32, and finally 2D convolution with 1-1-E, where E is the channel dimension, $E = 3$.

3D Detector. Our detector follows the architecture design of (Tung, Cheng, and Fragkiadaki 2019), which extends the 2D faster RCNN architecture to predict 3D bounding boxes from 3D features maps, as opposed to 2D boxes from 2D feature maps. The detector takes the 3D feature map from the 2D-to-3D lifting as input to predict object bounding boxes. The detector consists of one down-sampling layer and three 3D residual blocks, each having 32 channels. We use 1 anchor box at each grid location in the 3D feature map with a size of 0.12 meters for the CLEVR dataset, and a size of 1.7 meters for the CARLA dataset. The detector will output an objectness score for each anchor box and select boxes that exceeds a threshold. We set the threshold to be 0.9. While training the 3D detector, we freeze the weights of the 2.5D-to-3D lifting module and only finetune the layers after the 3D feature maps. We empirically found that training both the detector and the 2.5D-to-3D features degrades the quality of the features and the detector.

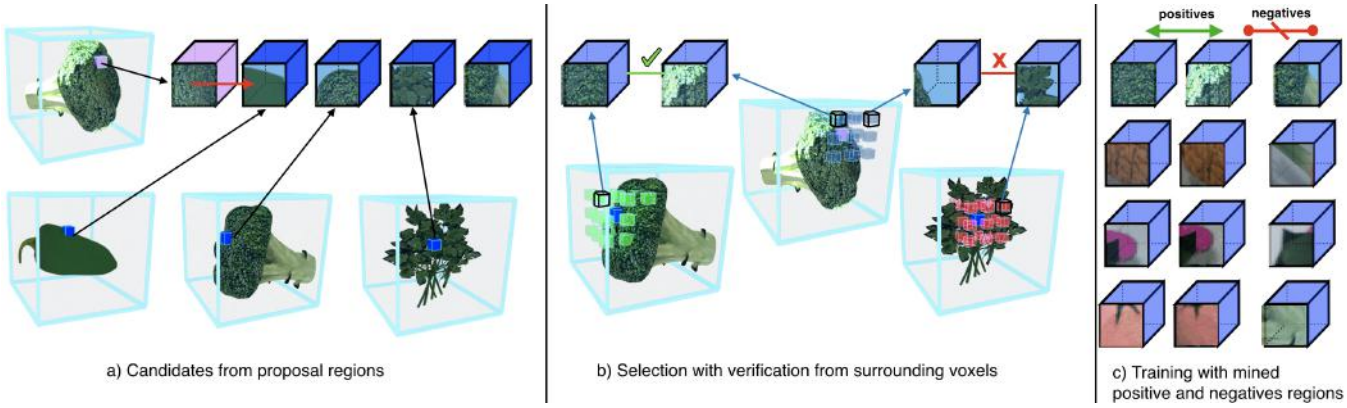

Figure 6: **Cross-scene 3D correspondence mining**. (a) We show that our approach relies on part-level correspondences obtained by matching the features of the query region (in pink) to a pool of object-centric 3D features maps. (b) These part-level correspondences are verified based on how well their surrounding voxels match with one another in a spatially consistent manner. (c) Finally we train our 2.5D-to-3D lifting module by doing metric learning using the verified positive regions and randomly sampled negatives.

3D correspondence mining. We randomly select 2000 object instances from our training data to create two pools (Query Pool & Target Pool) of size 1000 each. Each pool maintains object-centric 3D features of spatial size $16 \times 16 \times 16$ extracted from the 3D feature map using the detected boxes. As shown in Figure 6 for each training iteration, we randomly select a $2 \times 2 \times 2$ patch on an object sampled from the Query Pool, and by doing exhaustive search (across 36 different orientations along the vertical axis) and verification in the target pool object features we mine positive patches for metric learning training.

However, searching over all the possible patches (we extract 4 patches from each object) for all 1000 objects in the target pool with all the 36 poses is computationally inefficient. To reduce computation, we first complete a rough search at the object-level to retrieve objects which are similar to the query object, then we do fine-grained search at the part-level by searching over possible patches from these objects of interest. We do this by ranking objects based on their cosine distance (we take the maximum cosine distance across 36 rotations) with the query object, and take only the top 30 objects to perform fine-grained search on the patch-level.

For each target object, we extract 4 patches to compare with the query patch. For each patch, we conduct a spatial consistency check similar to the work of (Shen, Efros, and Aubry 2019): instead of computing inner product between the patches, we compare the surrounding patches of these patches. We take the patches 6 unit Manhattan distance away from the patch center and compute an inner product on these surrounding patches. The summation of the inner product between all the surrounding patches serve as the final matching score for center patches. We take the top 200 patch retrievals based on the score, and take the 8 corners from their surroundings as positives. We create negatives by randomly selecting a pair of patches from the pool. However, training with naively sampled negatives on the fly is unstable. Following the suggestion from the work of (He et al. 2019), we maintain a dictionary of size 100,000 for the negatives examples, and do momentum update on our 2.5D-to-3D lifting module.

Quantizing objects into prototypes. Each object prototype is a 3D feature tensor of size $16 \times 16 \times 16 \times 32$. We initialize these prototypes incrementally and assign an exemplar as a prototype only if its feature distance to the already-initialized prototypes is lower than a cosine distance of 0.8. This ensures diversity of prototypes during initialization. While associating exemplars to a prototype, we check over 36 different rotations along the vertical axis at 10° increments. We keep our prototype dictionary size K as 50 for all the datasets. Empirically from Figure 3(a)(main paper) we have found that, K should be large enough to cover the object variability in the dataset.

3D object detection supervised by prototypes distance and center-surround score. We initialize our 3D object detector by training it using triangulated 2D class-agnostic bounding box detections obtained from a 2D objectness detector. For our 2D objectness detector we use the publicly available code of (Wu et al. 2019) which uses a Faster R-CNN(Ren et al. 2015) backbone architecture and is trained using lots of 2D bounding box annotations from the COCO dataset (Lin et al. 2014). A 3D detector trained with noisy annotations obtained from triangulation is expected to perform poorly, and thus we found it critical that we have a mechanism to improve it over time.

In order to improve our detector, we first crop the object features from the inferred 3D feature map using the predicted 3D bounding boxes of our detector and resize their spatial dimension to $16 \times 16 \times 16$ to obtain object-centric feature tensors. For every cropped object tensor we calculate the cosine distance which is maximum amongst all the prototypes in the dictionary. If this calculated distance for

a proposal is greater than 0.8 then we keep it as a valid proposal. In-order to find the invalid proposals we use 3D center-surround saliency. Specifically we calculate the average cosine-distance of the cropped object tensor with its surrounding (top, down, left, right, front, behind) across all 3 axes. If the average cosine-distance is above 0.65 then we consider that proposal as invalid. We finally use the valid proposals as pseudo ground truths to further train the detector. We pass our gradients only through the aggregated region of all the valids and invalids, with the fear that there could be a prospective object proposal in the remaining region which was never predicted by the detector. The hope is that via iterative learning our detector learns 3D objectness and is thus able to get rid of it's bad proposals.

Implementation details for all baselines used in experiment subsection 4.2(main paper). Inorder to ablate the learnt latent representation we make sure that the prototypes for our model and all our baselines use the same number of bytes.

Pointcloud Registration(Mitra et al. 2004) We specifically select this as one of our baselines inorder to compare our model against a traditional computer vision method which doesn't use deep learning to learn its features. In this baseline we use registered point clouds as prototypes. We conduct 3D rotation aware search to identify the identity and 3D pose of the new object instances with respect to the prototype. We voxelize the point clouds into a 3D grid while computing the cosine similarity between the two. Since the new object instance will have an incomplete point cloud, we compute similarity only for the occupied points of the new object instance.

2.5DQ-Nets This baseline has the exact architecture as our 3DQ-Nets model, except it uses 2D CNNs instead of 3D and uses 2D rotation aware search instead of a 3D search. The model is trained on autoencoding the same RGB-D view instead of different query view like our model.

no-rot-3DQ-Nets This baseline is an ablation for 3DQ-Nets rotation aware check. In this model we follow the same procedure as 3DQ-Nets but do not do any rotation aware check while matching 3D instances with the prototypes.

9 Additional results

9.1 Quantiative results for clustering with 3D pose-aware quantization

Due to insufficient space in the main paper, in this section we further extend the experiments conducted in Figure 3(a)(main paper) on CARLA datset to all other datasets. In Table 4 we show the unsupervised classification accuracy using the same testing/training setup of Section 4.2(main paper). For this experiment we set the number of prototyes (K) as 50.

Datasets.	2.5DQ-Nets	no-rot-3DQ-Nets	Pointcloud registration	3DQ-Nets
CLEVR	0.23	0.73	0.51	**0.77**
BigBIRD	0.28	0.81	0.57	**0.83**

Table 4: **Unsupervised classification accuracy** with dictionary size of 50 prototypes on CLEVR and BigBIRD datasets.

9.2 Quantiative results for 3D object detection improvement

In this section, we show how the mean average precision (meanAP) of our 3D detector improves over time when supervised by visual compression and 3D center-surround saliency. We consider two initialization schemes for our 3D detector: i) we train our detector with a set of ground-truth 3D bounding boxes in a training set (3D-pretrain), ii) we train our detector by triangulating 2D object proposals from our 2D objectness detector, as described in Section 3.4(main paper), again in a training set (2Dtriang-pretrain). We show results in Table 5. From the results, we see our detector can improve its detection by a large margin after finetuning its weights by learning on the positive examples suggested by the learned object prototypes and negatives examples from the center surround check.

9.3 Qualitative results for 3D feature representation learning

Here, we show the qualitative results for object and patch retrieval using the learned 3D visual feature representations from the proposed cross-scene 3D correspondence mining in Section 3.3(main paper). More implementation details are given in Section 8 and Figure 6 of this Appendix.

Object Level Retrieval. Figure 7 shows the qualitative results for object level retrieval. Here, we compare the object retrieval results on object-centric (cropped and resized) 3D features maps which are learned from the proposed method (rgbocc + 3D correspondence mining) and 2 other baselines: rgbocc and rgbocc+vcdict, which are detailed in Section 4.3(main paper). We show the results on 3 datasets: CARLA, BigBIRD, and CLEVR. For each query image, shown in the first column, we show the top 5 retrievals for the three methods mentioned above. The green box signifies that the retrieved image belongs to the same object category as the query, but is in a different viewpoint of the same scene. Blue box depicts retrieval of the same object category from a completely different scene. As can be seen, our method (rgbocc+3D mining) gives much more accurate retrievals (more number of blue and green boxes) compared to the other two baselines across all datasets. We show the object level retrieval results for this method on Replica dataset in Figure 14.

Patch Based Retrieval. Figure 8 shows the 3D object patch retrieval results using the learnt 3D features from the proposed cross-scene 3D correspondence mining technique. We visualize the top 5 object part retrievals given a query object patch and a pool of target objects. For each query image, we first unproject it in the 3D space, detect objects in the scene, and randomly select a 3D patch on one of the objects. The first column for each dataset represents the query

Datasets	3D pretrain	3DQ-Nets (*final*)	2D triang-pretrain	3DQ-Nets (*final*)
CARLA	0.41	**0.59**	0.32	**0.41**
CLEVR	0.42	**0.61**	0.37	**0.52**

Table 5: **Initial and final 3D detection meanAP** at IoU=0.5 using detected 2D proposal triangulation versus ground-truth 3D bounding boxes in a training set. In both cases, 3DQ-Nets improve the detector over time, supervised by compression and 3D center-surround saliency.

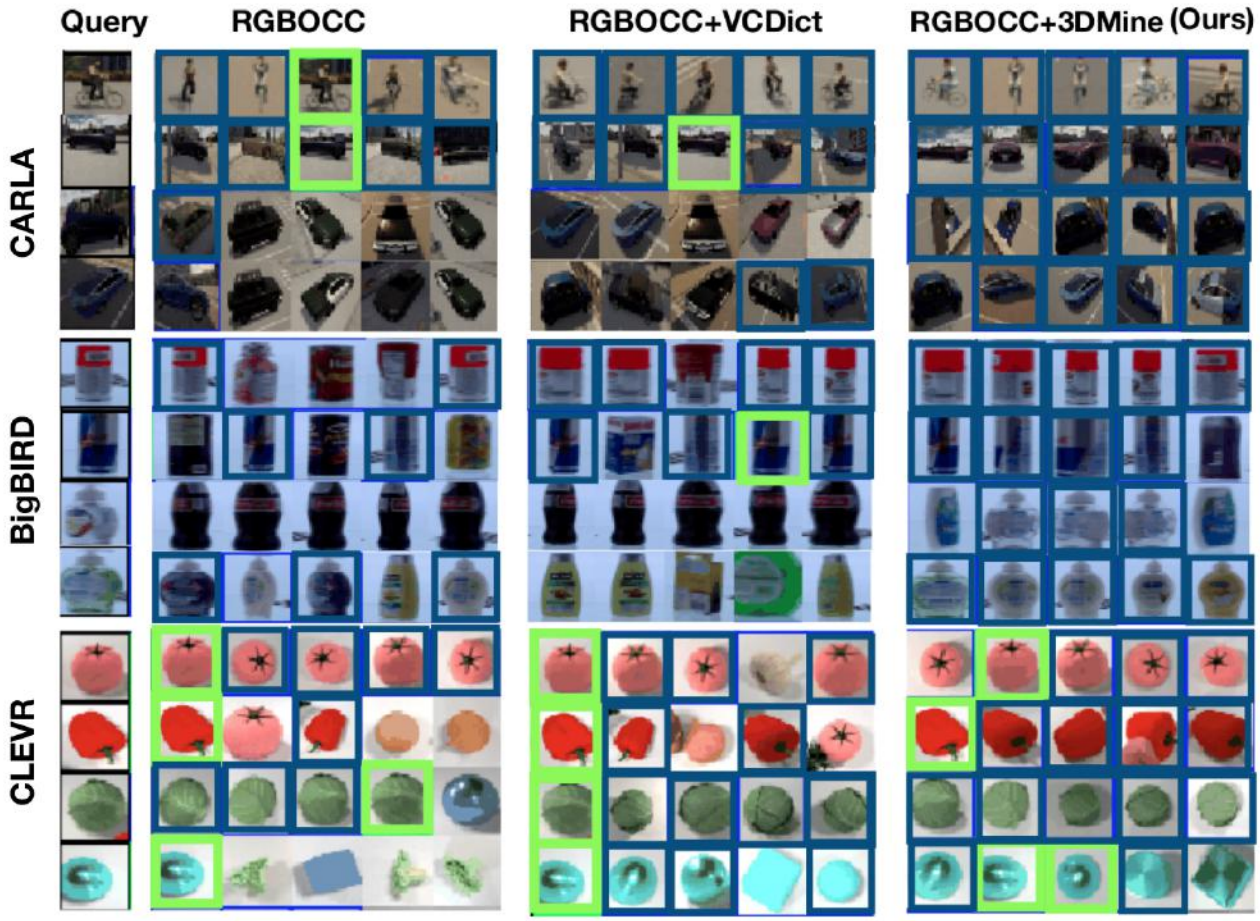

Figure 7: **3D object retrieval results** obtained by retrieving image patch using features learned from different feature learning methods, including rgbocc, rgbocc+vcdict, and rgbocc+3D correspondence mining (3DMine) methods. We visualize the retrieval results on CARLA, BigBIRD, and CLEVR datasets. The green boxes indicate that the retrieved image patches belongs to the same object instance as the query, but is in a different viewpoint. The blue boxes indicate instances with the same ground truth object category labels.

and the next 5 columns show the corresponding top 5 retrieved patches. For each query-prediction row pair, the first row shows the input RGB images and the second row shows bird's eye view of the same RGB images unprojected in 3D space. The blue patches in the bird's eye view visualizations (2nd row) show the 2D projection of the query/retrieved 3D patch. We additionally show patch based retrieval results on Replica dataset in Figure 15. We show the top 5 retrieved 3D patches that best matched the corresponding query patch using verification from surrounding voxels technique described in Figure 6 (b). As can be seen, patch based retrievals seem meaningful when surrounding context is given importance.

Rotation Matching. Finding the rotation transformation between two randomly posed RGB images is a crucial step for our model. As mentioned in Section 3.2(main paper), to do pose-equivariant quantization, we need to first align the input object 3D feature tensors with an object prototype. The quality of our quantization relies on the quality of the features that will yield the correct rotation alignment. We show

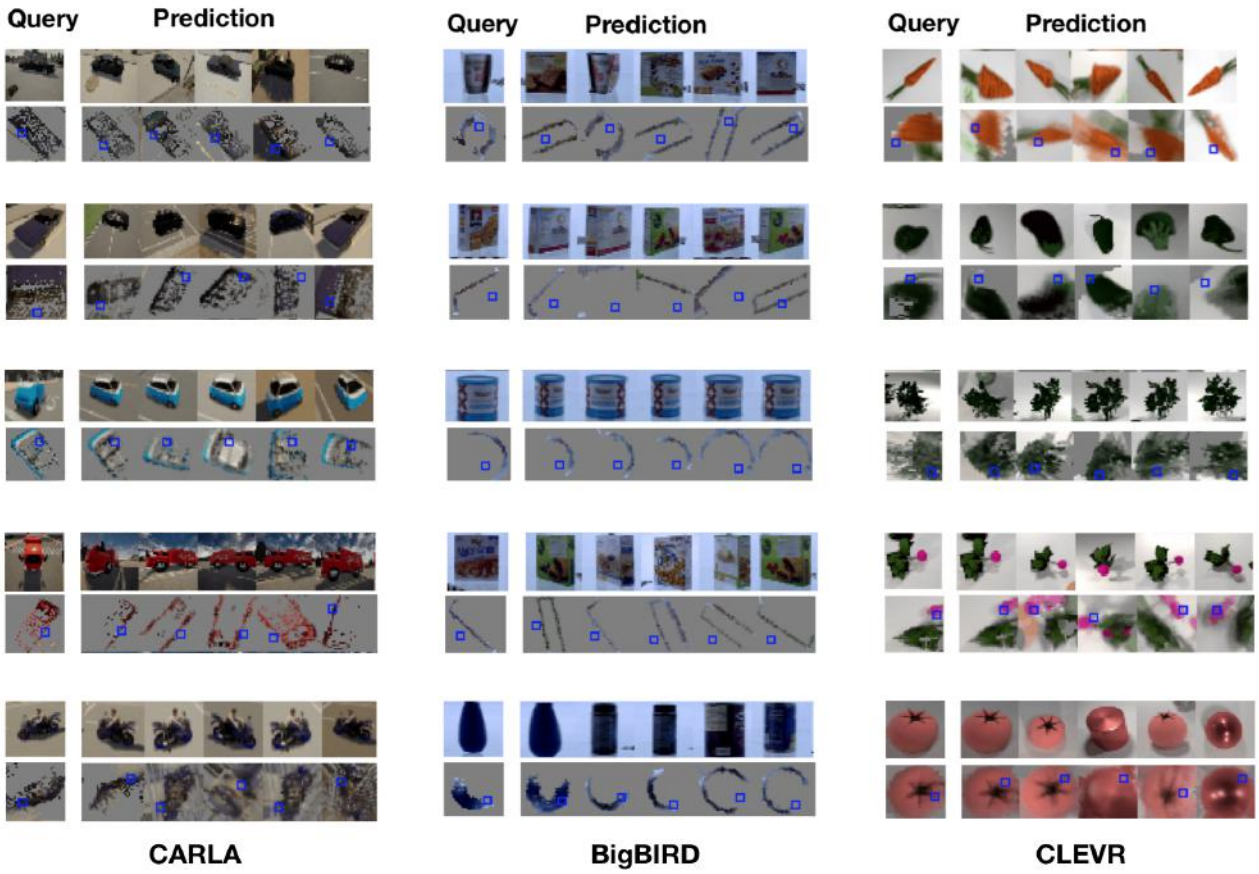

Figure 8: **Patch based 3D object retrieval results** on CARLA, BigBIRD, and CLEVR datasets. For each query-prediction row pair, the first row shows the input RGB images and the second row shows bird's eye view projection of the RGB-D point cloud. The blue patches in the bird's eye view visualizations (2nd row) show the 2D projection of the query/retrieved 3D patch.

the qualitative performance of such rotation assignment on CARLA, BigBIRD and CLEVR datasets in Figure 9. For each of those 3×7 grids, the first row shows the input RGB images of the same object category in different poses, the second row shows the bird's eye view of the same RGBs unprojected in 3D space, and the third row shows the bird's eye view of the same unprojected RGBs but warped to the pose that best matches with the object in the first. We conduct this matching on top of our 3D feature space by doing a rotation aware search. As shown in the visualizations, our model can warp the objects in different orientations to an orientation in the vicinity of the pose of the target object.

9.4 Qualitative results for scene reconstruction using learned 3D object prototypes

Learning 3D prototypes is a fundamental part of our pipeline as it helps us in inferring object associations and poses across different scenes. In this section, we compare the RGB neural reconstruction of a scene after replacing the objects in the scene with the prototypes learned using our model and the 2.5DQ-Nets baseline on both CARLA and CLEVR datasets. In Figure 10, we show the neural scene reconstructions after the object-prototype replacement. For our model we show the 2D neural render of the scene at a different camera view than the input view, whereas for the 2.5DQ-Nets baseline we reconstruct the image at the same camera view.

For each dataset, the first column represents the ground truth RGB render of the scene. Third column represents the target view neural render of the scene using our learned 3D prototypes. This reconstruction is obtained by lifting the 2.5D input to 3D feature space, extracting the object from this feature space, finding the best matching prototype, warping the prototype to the pose of the input object, replacing the object features with the warped 3D prototype features, and finally performing RGB view prediction to the target view with these 3D features. The second column shows the reconstruction results when we follow the same procedure as before but use 2D prototypes and 2D rotation check instead of 3D. The 2D prototypes, because of their inability to be 3D rotation-equivariant while quantization, end up learning the mean representation of objects in different poses, which appears as a circular blur. The 3D prototypes, on the other hand, give sharp reconstructions because the objects in different poses are mapped to the same canonical pose. Results on the Replica dataset are shown in Figure 16.

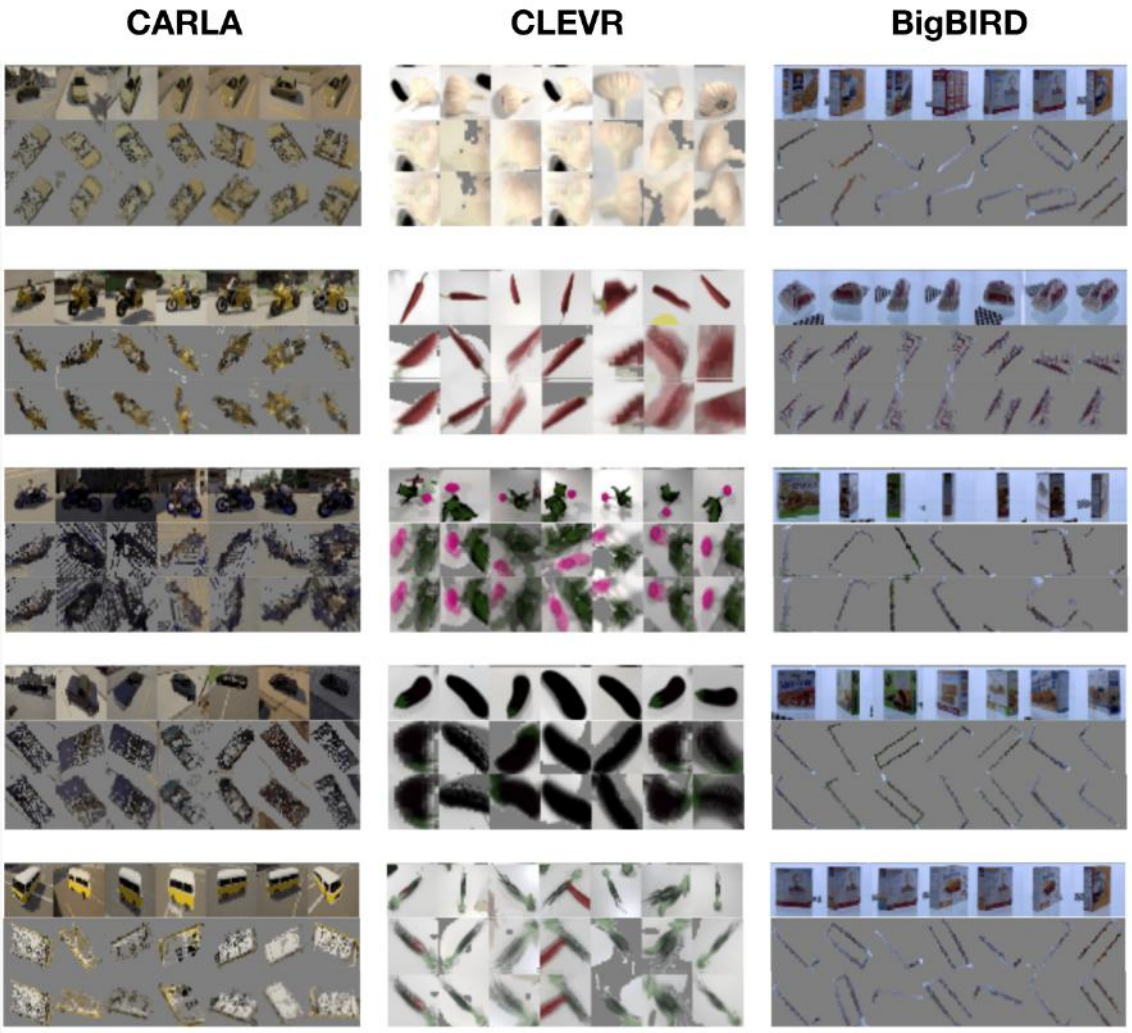

Figure 9: **Rotational alignment results** showing relative pose estimation between two randomly posed RGBs of the same object category. For each of the 3×7 grids, the first row shows 7 input RGB images of the same object category in different poses. The second row shows the projection of the RGB-D point cloud in a bird's eye view. The last row shows the projection of the same RGB-D point but warped to the pose that best matches with the object in the first. Results are shown on CARLA, BigBIRD and CLEVR datasets.

9.5 Qualitative results for the self-improving object detector

As mentioned in Section 3.4(main paper), our model can use the learned 3D object prototypes to self-improve its object detection. The object detector will propose several 3D bounding box proposals. The model will then self label some of these proposals as good proposals if the content inside the proposal can be well-explained by the learned 3D object prototypes, and will label it as bad proposals if the content inside the proposal is not salient and has low 3D center-surround score. The object detector then uses these pseudo labels to refine its weights to achieve better detection. We visualize the detections made by our self-improving detector on CLEVR dataset over 4 iterations in Figure 11. The first row of the figure represents the bounding boxes predicted by our detector at each iteration. The second row shows the self annotated labels generated using the prototype distance and center-surround score for each bounding box at each iteration. The negative boxes which are to be pruned are shown in red, and the ones to be kept are shown in green. As can be seen, our model can propose accurate positive and negative labels to the proposed 3D boxes. Although the object detector performs poorly in the first iteration, the quality of the object proposals made by our object detector can improve over iterations and produce accurate results after 4 iterations.

9.6 Qualitative results for scene parsing

In this section, we show that using our learned prototypes and self improved 3D object detector, we are capable of

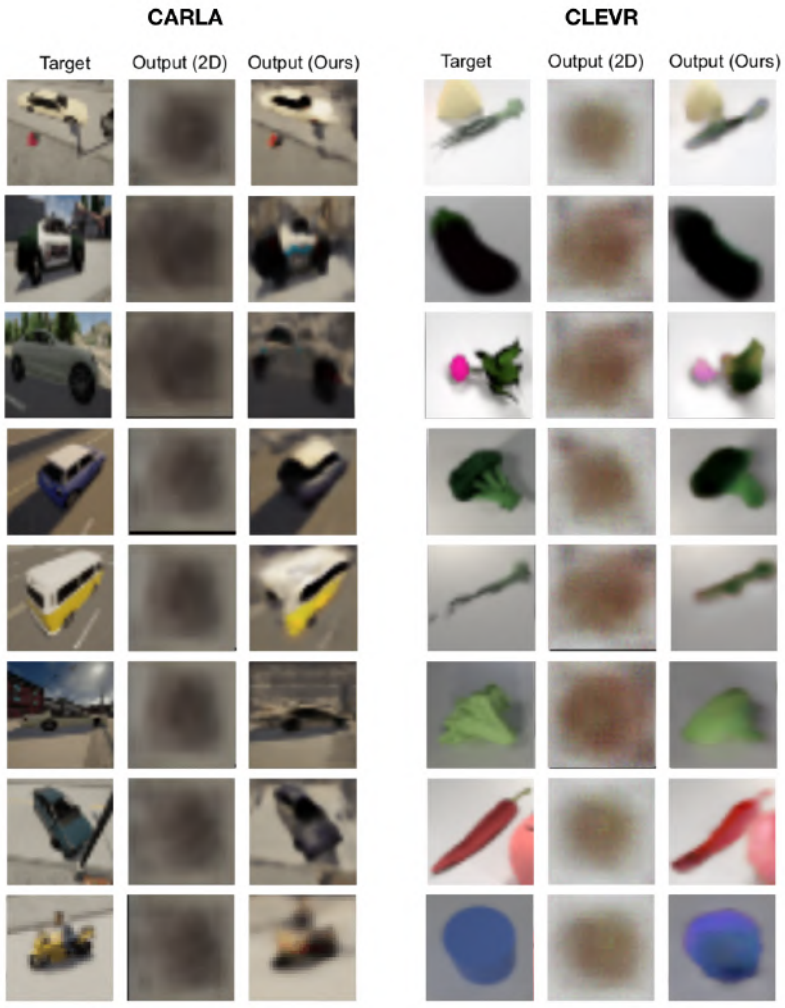

Figure 10: Prototype reconstruction results using the learned 3D prototypes from our model and their comparison with 2D prototypes.

parsing a scene at test time. Figure 12, Figure 13, and Figure 17 show the scene parsing results of our 3DQ-Nets on the CARLA, CLEVR, and Replica datasets, respectively. Our model is able to learn 3D scene parsing without any 3D supervision. As explained in the paper and shown in the figures, 3DQ-Nets can learn to infer 3D scene parsings from the input RGB-D images.

The top row in each visualization in both the figures shows the parsing of a scene by drawing the inferred bounding boxes, along with a text on top of each box mentioning the Prototype Number(C) and Rotation angle(R) in degrees. For example, an object associated to Prototype 5 with a Relative Rotation of 150° with respect to its associated cluster 5 prototype is represented as *C5_R150*. In the next row for each visualization we also show 3 different camera view neural renders of the prototypes, while also mentioning their respective prototype numbers. These neural renders of prototypes are generated by placing the prototype in randomly selected backgrounds. Note that the rotation angles shown in the results are relative to the pose of the reference camera used for recording the scene. Since we randomize the orientation of the reference camera for each scene, it is possible that two objects in seemingly similar poses have different relative rotation angles.

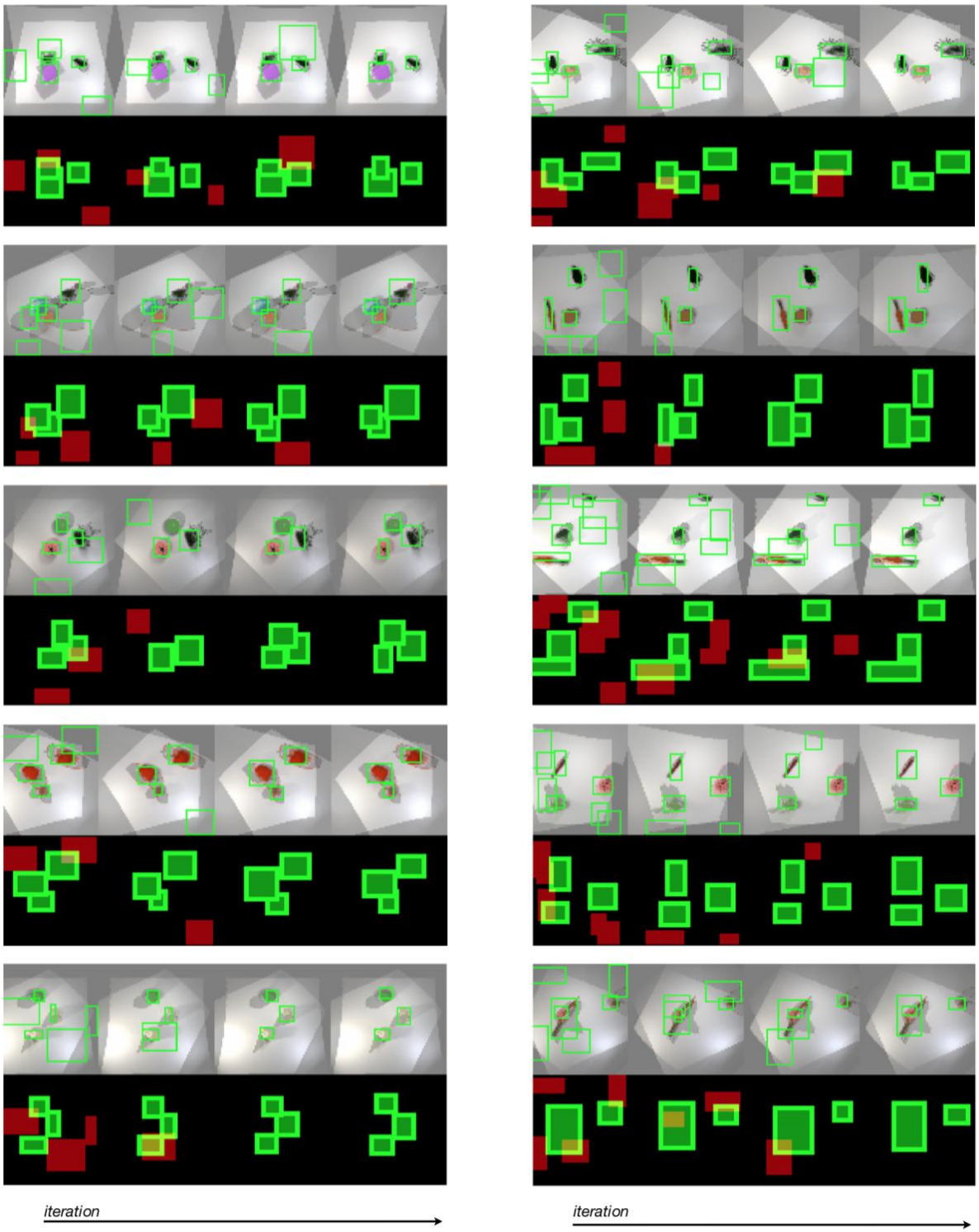

Figure 11: The outputs from the proposed iterative detector improvement for 4 iterations.

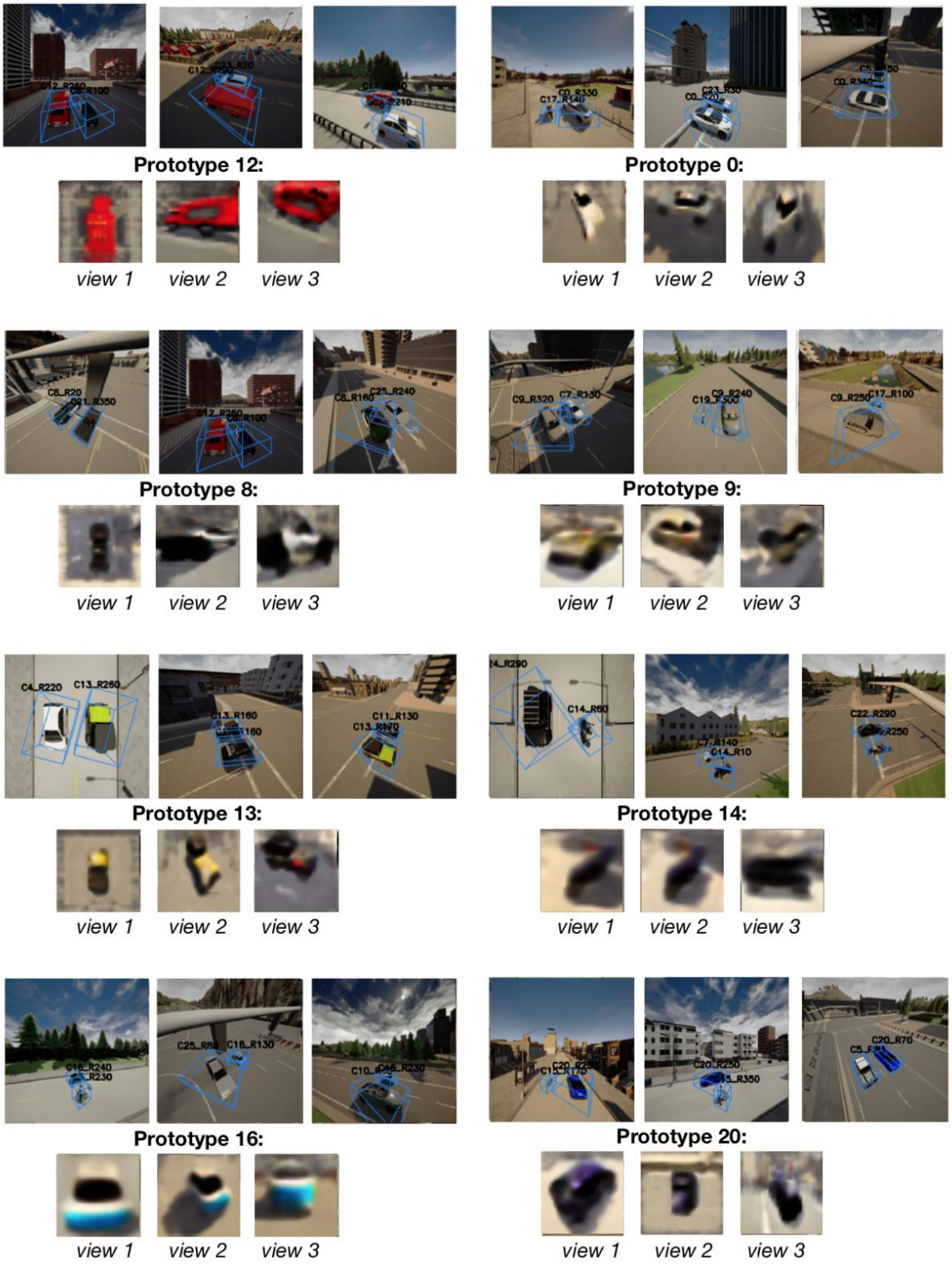

Figure 12: Scene parsing results for CARLA dataset.

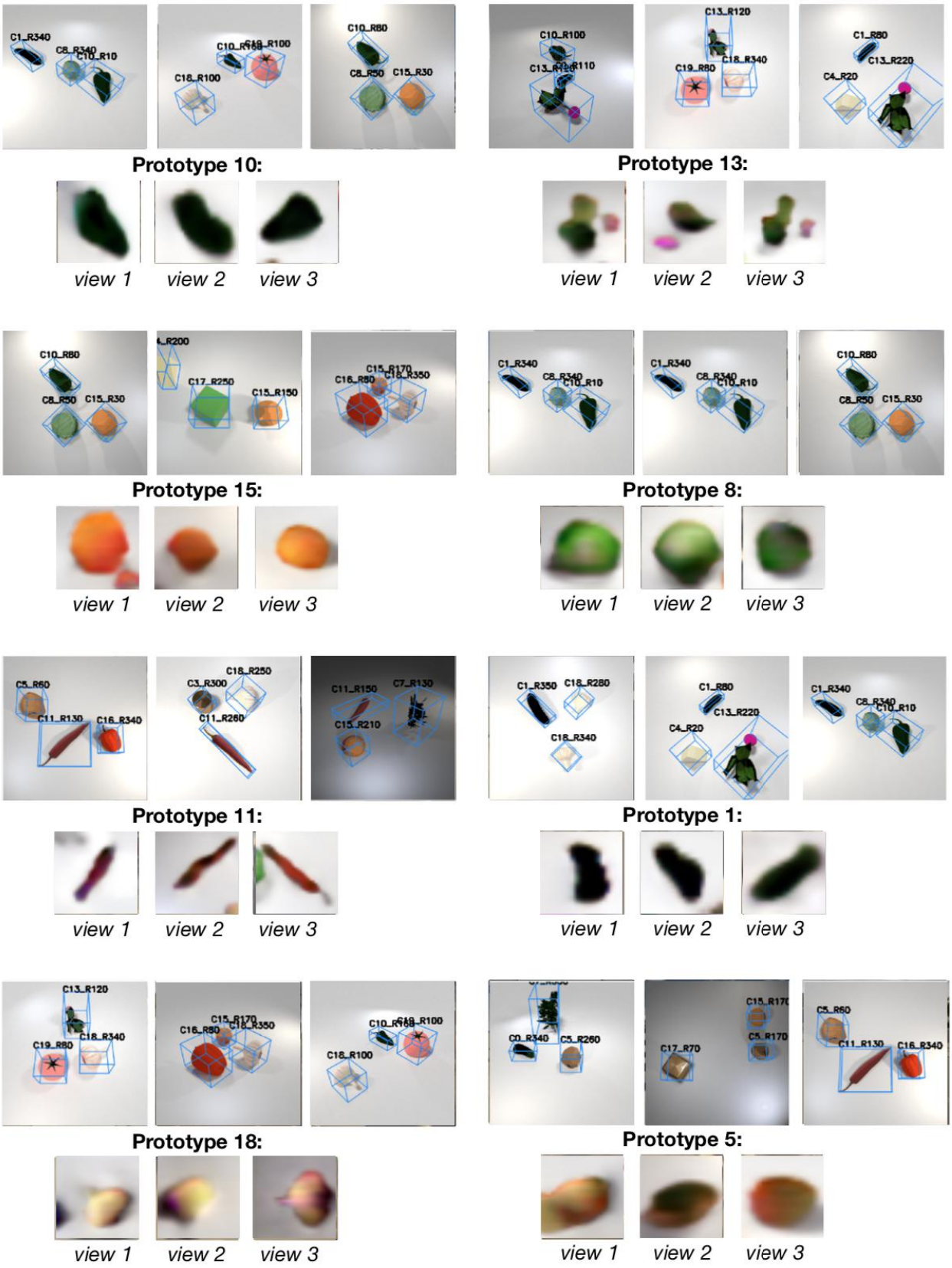

Figure 13: Scene parsing results for CLEVR dataset.

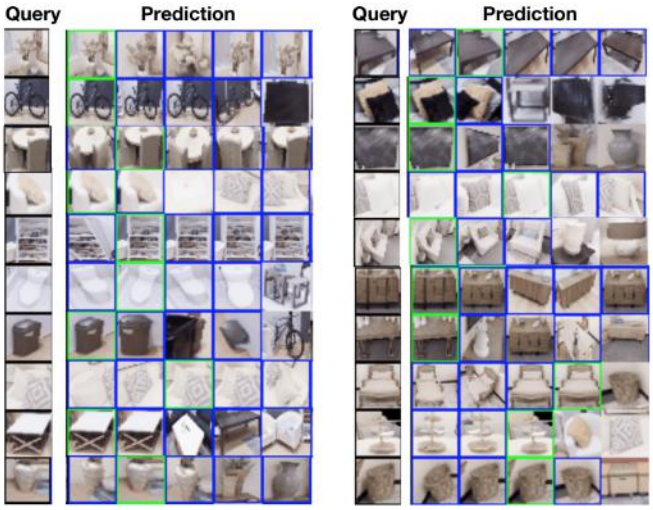

Figure 14: 3D object retrieval results obtained by rgbocc+3D correspondence mining on Replica dataset.

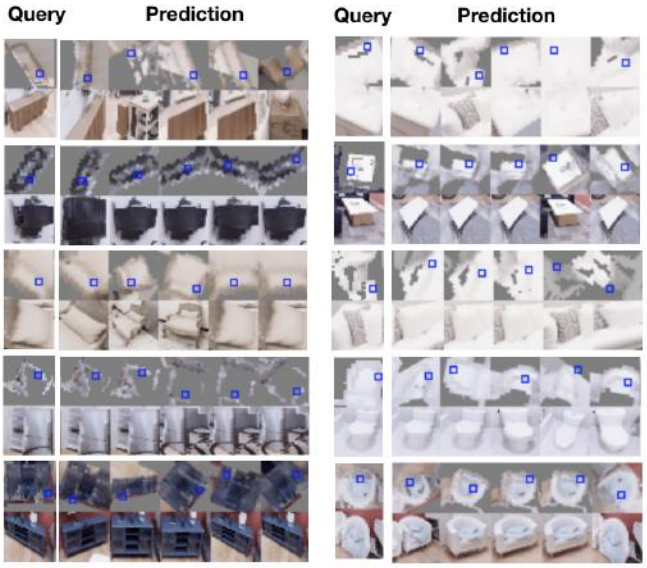

Figure 15: Patch based 3D object retrieval results on Replica dataset.

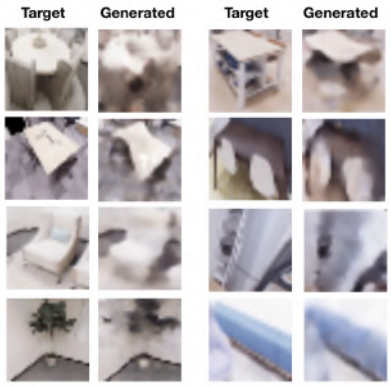

Figure 16: Prototype reconstruction for 3D prototypes learned by our model on Replica dataset.

Figure 17: Scene parsing results for Replica dataset.