Self-Supervised Learning of Object Segmentation from Unlabeled RGB-D Videos

Shiyang Lu[1], Abdeslam Boularias[1], Yunfu Deng[2], Kostas Bekris[1]

***Abstract*— This work proposes a self-supervised learning system for segmenting rigid objects in RGB images. The proposed pipeline is trained on unlabeled RGB-D videos of static objects, which can be captured with a camera carried by a mobile robot. A key feature of the self-supervised training process is a graph matching algorithm that operates over the point cloud over-segmentation output that is reconstructed from each video. The graph matching, along with point cloud registration, is able to find reoccurring object patterns across videos and combine them into 3D object pseudo labels, even under occlusions or different viewing angles. Projected 2D object masks from 3D pseudo labels are used to train a pixel-wise feature extractor through contrastive learning. During online inference, a clustering method uses the learned features to cluster foreground pixels into object segments. Experiments highlight the method's effectiveness on both real and synthetic video datasets, which include cluttered scenes of tabletop objects. The proposed method outperforms existing unsupervised methods for object segmentation by a large margin.**

I. Introduction

Household autonomous robots should be able to reason about objects even when human supervision is not available. This work considers a setup where a robot navigates in a static environment and passively collects RGB-D videos. The environment contains unknown, rigid objects, which rest stably, potentially in cluttered configurations, on supporting surfaces, such as tabletops. The same object instances can reappear in various scenes with arbitrary 6D poses, while being partially occluded by different surrounding objects. This raises the following question: Can a robot learn to correctly segment objects in RGB images given its prior experience of collecting unlabeled videos that contain the involved objects without any human supervision?

While some objects, such as a "coke can" or a "sugar box", can be relatively easily singled out from a scene due to their characteristic texture or simple geometries, other more complex and articulated objects pose a more serious challenge in the absence of prior knowledge. Objects made of multiple parts of different shapes and colors, such as a "lamp" with a concave shape involving a gooseneck and a base, are particularly challenging to be identified as unique objects. For such objects, the robot cannot tell from a single snapshot if the different parts are individual objects or part of the same object. This paper argues that robots can address this problem without prior knowledge or human supervision by simply collecting videos of scenes, which can be extended to a lifelong learning process. The idea is to identify object parts that appear simultaneously in the same spatial arrangement in different scenes. The key contribution of this work towards this idea is a graph-matching algorithm that identifies reoccurring sets of neighboring 3D object parts. The overall approach utilizes the results of the matching algorithm to train a pixel-wise feature extractor through contrastive learning, which can be directly consumed by a clustering method for object segmentation.

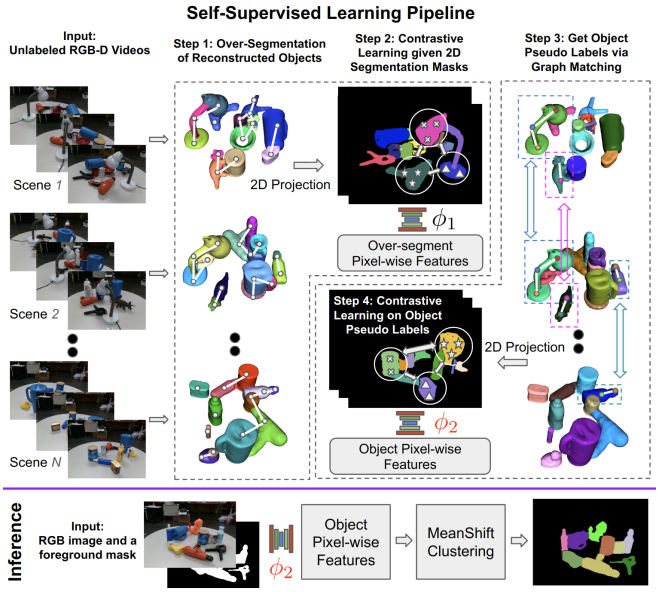

Fig. 1: **Self-Supervised Learning Pipeline.** Top: Step 1. The training system over-segments the 3D scene reconstruction of each video. Step 2. An initial feature detector ϕ_1 is trained using contrastive learning, where positive examples come from pixels belonging to the same 3D segment. Step 3. Graph-matching operates over the 3D segments and their ϕ_1 features to identify sets of segments to be grouped as objects. Step 4. Contrastive learning is used to train detector ϕ_2, which uses as positive examples pixels that belong to the same hypothesized objects. Bottom: Upon inference, given an RGB image and a foreground mask, the network ϕ_2 generates pixel-wise features, which allow a clustering algorithm to segment objects.

The proposed pipeline is shown in Fig. 1. The learning system receives as input a collection of unlabeled RGB-D videos from multiple scenes, and returns a pixel-wise feature extractor ϕ_2 that can generate distinctive object features for segmentation. The system first performs (over-)segmentation of the 3D point cloud for each input video resulting in 3D segments corresponding to simple geometries. This is done by reconstructing each 3D scene by using KinectFusion [1], removing the background using plane detection, and segmenting the scene using existing non-learning, geometry-based methods [2], [3]. The 3D segments are projected to the

[1]The authors are affiliated with the Department of Computer Science at Rutgers University, New Brunswick, NJ, 08901, USA. Email: {shiyang.lu, abdeslam.boularias, kostas.bekris}@rutgers.edu. [2]This author is affiliated with the Department of Electrical and Computer Engineering at Rutgers University, New Brunswick, NJ, 08901

Manuscript 2112 submitted to 2023 IEEE International Conference on Robotics and Automation (ICRA). Received September 15, 2022.

2D frames, and a pixel-wise feature extractor ϕ_1 is learned to generate distinctive features for each 3D segment through contrastive learning, similar to MaskContrast [4]. The 3D segments are then abstracted as nodes of a graph, where edges connect adjacent nodes, and nodes are represented by their features according to ϕ_1. Reoccurring graph patterns are searched across different scenes by a feature-based, subgraph matching algorithm [5]. Graph matching is used here as a mechanism to identify candidate pairs of object instances with multiple parts. The corresponding point clouds of the candidates are registered using RANSAC [6], and matched nodes with low registration scores are pruned out. The remaining matched nodes and the segments represented by those nodes are considered as part of the same object. Then, a second network ϕ_2 is learned using contrastive learning to generate pixel-wise features of object instances. During inference, given an RGB image and a foreground mask, ϕ_2 returns pixel-wise object features, which are then clustered using a variant of the mean-shift algorithm [7], [8] to output the object segmentation of the input RGB image.

The experimental results of Section VI show that the proposed technique significantly outperforms multiple baselines both on real-world and photo-realistic synthetic video datasets. Its performance is close to training with supervision.

II. Related Work

Self-Supervised Learning. Many recent studies on self-supervised learning [9], [10], [11], [12] focus on generating a pre-trained model from a large dataset using pseudo labels without human supervision. Given the pre-trained model, downstream tasks (e.g., object segmentation) can be trained with less annotated data. Many of the existing methods require object region proposals that contain one or very few salient objects. This object localization process is usually performed by 2D region proposals [13] and saliency detection [4] for static images, or object tracking [14], [15] and optical flow [16], [17] for videos. Nevertheless, they are not as suitable for cluttered static scenes where many objects are occluded [18], as the errors introduced during region proposal are often too large. Meanwhile, generating a pretrained model is not ideal as it requires finetuning for downstream tasks, such as segmentation. This issue is more pronounced in instance-level segmentation. This work, on the other hand, focuses on finding object pseudo labels that can be directly used for training.

Object Segmentation without Manual Annotation. Many unsupervised methods have been proposed for object/scene segmentation. Notable non-learning methods are Mean Shift [7] for image segmentation, and LCCP [2], SymSeg [3], CPC [19] for point cloud segmentation. More recently, some efforts have studied RGB-D segmentation for unknown objects, including UOC [20], UOIS [21], RICE [22], SD-MaskRCNN [23]. These methods typically train a neural network on synthetic data of object mesh models using domain randomization, where ground-truth labels are automatically generated. These methods perform well on objects similar to the training data, but often generate sub-optimal results for objects out of the training distribution. Fine-tuning the network on real data is nontrivial without manual annotation.

Unsupervised Object Discovery An object detection method uses part-based matching with bottom-up region proposals [24]. An alternative formulates the problem as ranking amenable to distributed methods available for eigenvalue problems and link analysis [25]. Both methods aim for object detection instead of pixel-wise segmentation.

III. Problem Setup and Notation

This work addresses the problem of object segmentation given an RGB-D image I of a novel scene, which contains various, potentially cluttered objects. Each object is assumed to have appeared before in a collection of m unlabeled raw RGB-D videos $V = \{V_1, V_2, ..., V_m\}$ that were passively recorded by a mobile robot while observing static scenes. There are K unique object instances $O = \{O_1, O_2, ..., O_K\}$ that the robot can observe. Image frame at time t in the training i-th video V_i is denoted as I_i^t, i.e., $V_i = (I_i^1, I_i^2, ..., I_i^{N_i})$, where N_i is the total number of frames in V_i. Each frame I_i^t is composed of a color and a depth image, denoted as $I_i^t.color$ and $I_i^t.depth$ respectively. Each frame I of a video, as well as the test image, may contain only a subset of the object set O, denoted as $O^I \subseteq O$. Background objects, such as furniture, are assumed to be known a priori, and are automatically removed from the RGB-D images. No additional information, such as the number of objects in each scene, or labels that help with segmentation are available.

IV. Object Segmentation during Testing

During online inference, a learned function $\phi_2(x) : \mathcal{X} \rightarrow \mathcal{Z}$ is used to extract a normalized feature vector z for each pixel x in the given test RGB image. A clustering algorithm is applied over the foreground pixels based on their extracted features. This foreground mask can be generated with a saliency detection method, such as BasNet [26] and DeepUSPS [27]. See the example of Fig. 2.

Fig. 2: Example of saliency detection [26] for generating the foreground mask (right) given RGB input (left).

Since the number of objects in a given image is not known, clustering methods, such as K-Means, which require this input, are not applicable. Thus, this work adapts a variant of the von Mises-Fisher mean-shift [8] for feature clustering. This variant not only considers feature similarity of pixels but also 2D proximity of pixels. Furthermore, a simple post-processing step is applied so that pixels in a feature cluster are separated if the corresponding pixels do not form a connected component. Online inference is computationally efficient ($\sim$ 6fps). The detailed training process of ϕ_2 is described in the following section.

V. Self-Supervised Learning Pipeline

A. Video Pre-processing

Given a set of unlabeled RGB-D videos of static objects, KinectFusion [1] is used to reconstruct a 3D scene for each video. Background removal is performed after down-sampling the 3D reconstructed point cloud with a voxel size of $5mm$. The foreground masks during training are generated by removing the known background objects on the 3D reconstructed scenes, and projecting the remaining object point cloud to individual 2D frames. It is important to employ 3D reconstruction, instead of individual depth images from the same scene, for multiple reasons: a) It allows to generate pseudo labels on the reconstructed point cloud once and then acquire consistent labels for segments on all of the frames of each video via 3D to 2D projection, b) It increases visibility of each object given multiple views, which reduces issues due to occlusions that can arise for individual viewpoints, c) It avoids having to find correspondences between segments across frames, which can be error-prone, d) Performing segmentation on each frame is more time-consuming.

B. Object Over-Segmentation

Each reconstructed 3D scene is over-segmented into a set of small segments corresponding to simple geometries. Ideally, this over-segmentation should be consistent across different scenes, where consistency means that the same object surface is segmented in the same way across different scenes. This work first adopts LCCP [2] for this purpose. LCCP is a non-learning segmentation method based on the local convexity of point clouds. It is able to segment convex objects, such as a box, in to a whole, and segment complex objects into small, approximately convex parts. During the development of this work, it is found that LCCP can fail on certain objects with a large concave surface, such as a bowl. To alleviate this issue, SymSeg [3] is introduced to assist with improving segmentation accuracy for symmetric concave objects. SymSeg first detects symmetries in point cloud and then uses them for segmentation. If no symmetry is detected, SymSeg is not applied. In this work, only rotational symmetries are used, since reflection symmetries often lead to under-segmentation in clutter. A simple strategy to fuse the results of LCCP and SymSeg is illustrated in Fig. 3. A 3D segment generated by LCCP is first split into multiple parts if an overlapping symmetric segment is detected by SymSeg. LCCP segments are then merged if they appear in the same symmetric segment of SymSeg, with an exception when the split boundary lies in a 2D plane that is orthogonal to the rotational symmetric axis (e.g. a bottle standing in a bowl). Small segments with less than 200 points are pruned. Note that this work does not argue that this combination is the best for over-segmentation, but they work sufficiently well as a sub-module in the pipeline. Any recent progress in class-agnostic over-segmentation may benefit this work.

C. Contrastive Learning over 3D Segments

The next step is to train a feature extractor ϕ_1 using contrastive learning. This allows learning a distinctive feature

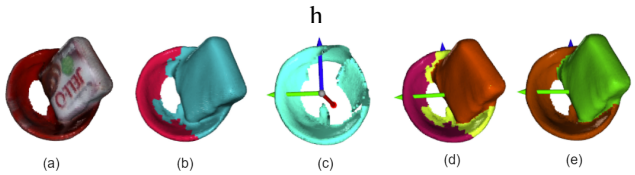

Fig. 3: **Splitting and Merging Segmentation Results of LCCP and SymSeg.** (a) Reconstructed point cloud of a "Jell-O" box in a bowl. (b) Irregular segments generated by LCCP due to notable concavity. (c) Detected rotational axis and segments generated by SymSeg. (d) Splitting segments. (e) Merging segments.

for each 3D segment, which can be used later during graph matching for evaluating the similarity of 3D segments. This contrastive learning is based on the assumption that points in the same segment belong to the same object. Each 3D segment is projected to all video frames using estimated camera poses during reconstruction. If a pair of pixels (i, j) belongs to the same 3D segment, their pixel embeddings (z_i, z_j) are pulled closer in the feature space. If (i, j) belong to different 3D segments, their features are pushed away. In practice, $P \gg N$, where P is the number of foreground pixels per image, and N is the number of segments per scene. The number of pixel pairs is $O(P^2)$, which results in a computationally expensive sampling process. To alleviate this issue, we adopt the strategy from MaskContrast [4] to reduce the computational complexity from $O(P^2)$ to $O(PN)$ as follows. Let M_i be the set of pixels from segment i. Define the mean pixel embedding of M_i as $\bar{z}_i = \frac{1}{|M_i|}\sum_{k=1}^{|M_i|} z_i^k$. The mean feature $\bar{z}_i$ is used to represent the features of all pixels in M_i. Given positive pairs $(z_k, \bar{z}_i)$, for $k \in M_i$, and negative pairs $(z_k, \bar{z}_j)$, for $k \notin M_j$, the contrastive loss for each foreground pixel k is defined as: $\mathcal{L}_k = -log\frac{exp(z_k \cdot \bar{z}_i)/\tau}{\sum_{j=1}^{N} exp(z_k \cdot \bar{z}_j/\tau)}$, where τ is a temperature hyperparameter that has a constant value of 0.07 in all our experiments (as in MoCo [10]). The mean feature $\bar{z}_i$ obtained from contrastive learning is used to represent the 3D segment i.

D. Graph Matching

For each scene, a graph is generated by assigning each 3D segment to a node and defining an edge between every two segments that share a boundary in the 3D space. Re-occurring subgraphs across multiple graphs (i.e., scenes) are hypothesized objects, since it is unlikely for two separate objects to always appear in the same poses relative to each other in all of the scenes encountered by the robot. To find these reoccurring subgraphs, this work adapts an algorithm based on error-tolerant, minimum-cost subgraph matching [5]. Given a source graph $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and a target graph $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the algorithm identifies a subgraph $S \subseteq G_2$, that minimizes the cost of matching S to a subgraph in G_1 given both structural and feature distortions. This optimization problem is formulated into a binary linear program (BLP) with the following objective function:

$$J = \min_{x,y,\alpha,\beta} \Big(\sum_{i\in\mathcal{V}_1}\sum_{k\in\mathcal{V}_2} x_{i,k}\cdot c(i\to k)) + \sum_{i\in\mathcal{V}_1}\alpha_i \cdot c(i\to\epsilon) + \sum_{ij\in\mathcal{E}_1}\sum_{kl\in\mathcal{E}_2} y_{ij,kl}\cdot c(ij\to kl) + \sum_{ij\in\mathcal{E}_1}\beta_{ij}\cdot c(ij\to\epsilon)\Big) \tag{1}$$

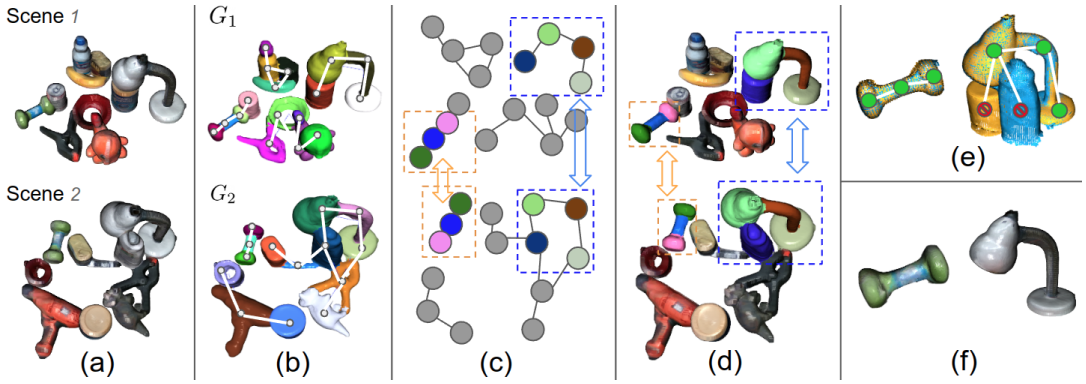

Fig. 4: **Graph Matching.** (a) Colored 3D reconstruction of two scenes. (b) 3D segments, and corresponding graphs, resulting from over-segmentation of the point clouds. (c) Matched nodes from graph matching are highlighted. Parts of the "dumbbell" and the "lamp" are correctly matched, while the "bleach cleanser" and the "master chef can" are incorrectly matched. (d) Matched connected components are shown in the same color. (e) 3D point cloud registration of matched components. (f) Segments/nodes that do not register well (such as the "bleach cleanser" and the "master chef can") are rejected before merging the remaining segments into objects.

where $x_{i,k}$, $y_{ij,kl}$, α_i and β_{ij} are binary variables to optimize: $x_{i,k}$ expresses the mapping of node i from source graph G_1 to node k from target graph G_2; $y_{ij,kl}$ expresses the mapping of edge (i,j) from source graph G_1 to edge (k,l) from target graph G_2; deletion variables α_i and β_{ij} are set to be 1, if node i and edge (i,j) are deleted, respectively, according to the match. This can happen when the 3D segment corresponding to node i in the source scene does not appear anywhere in the target scene. The cost of node mapping $c(i \to k)$ is defined as the cosine distance between the corresponding segment features $\bar{z}_i$ and $\bar{z}_k$. The cost of edge mapping $c(ij \to kl)$ is set to 1. The cost $c(i \to \epsilon)$ (or $c(ij \to \epsilon)$) expresses the cost of not assigning a vertex (or edge) of the source graph to a vertex (or edge) in the target graph. Both costs are set to 0.1, which can be viewed as thresholds for rejection.

$$\sum_{k\in\mathcal{V}_2} x_{i,k} \le 1, \forall i \in \mathcal{V}_1; \quad \sum_{i\in\mathcal{V}_1} x_{i,k} \le 1, \forall k \in \mathcal{V}_2 \tag{2a}$$

$$\alpha_i = 1 - \sum_{k\in\mathcal{V}_2} x_{i,k}, \forall i \in \mathcal{V}_1; \quad \beta_{ij} = 1 - \sum_{kl\in\mathcal{V}_2} y_{ij,kl}, \forall ij \in \mathcal{E}_1 \tag{2b}$$

$$\begin{aligned} \sum_{l\in\mathcal{V}_2 \text{ s.t. } kl\in\mathcal{E}_2} y_{ij,kl} \le x_{i,k}, \forall ij \in \mathcal{E}_1, \forall k \in \mathcal{V}_2; \\ \sum_{k\in\mathcal{V}_2 \text{ s.t. } kl\in\mathcal{E}_2} y_{ij,kl} \le x_{j,l}, \forall ij \in \mathcal{E}_1, \forall l \in \mathcal{V}_2 \end{aligned} \tag{2c}$$

$$\begin{aligned} x_{i,k} \in \{0,1\}, \forall i \in \mathcal{V}_1, \forall k \in \mathcal{V}_2; \\ y_{ij,kl} \in \{0,1\}, \forall ij \in \mathcal{E}_1, \forall kl \in \mathcal{E}_2 \end{aligned} \tag{2d}$$

The objective function is subject to constraints 2a-2d. Constraints 2a indicate that each node from either graph can be mapped to at most one node from the other graph. Constraint 2b indicates that if a node in G_1 is not matched to any vertex in G_2, then it must be deleted. Constraints 2c indicate that an edge should be mapped only if the corresponding vertices are also mapped. The constraints 2d ensure that the variables to be optimized are binary variables. Note that constraints 2b are implicitly respected given constraints 2a and 2c. Thus, the deletion variables α_i and β_{ij} in equation 1 can be removed to reduce the search space resulting in the following objective function:

$$\begin{aligned} J = \min_{x,y} \Big(& \sum_{i\in\mathcal{V}_1} \sum_{k\in\mathcal{V}_2} x_{i,k}\big(c(i \to k)) - c(i \to \epsilon)\big) \\ & + \sum_{ij\in\mathcal{E}_1} \sum_{kl\in\mathcal{E}_2} y_{ij,kl}\big(c(ij \to kl) - c(ij \to \epsilon)\big) \\ & + \sum_{i\in\mathcal{V}_1} c(i \to \epsilon) + \sum_{ij\in\mathcal{E}_1} c(ij \to \epsilon)\Big) \end{aligned} \tag{3}$$

Furthermore, since the 3D segment graphs are undirected, i.e., $(ij \in \mathcal{E}) \Leftrightarrow (ji \in \mathcal{E}), \forall i,j \in \mathcal{V} \times \mathcal{V}$, constraints 2c can be further combined into a single one, i.e.

$$\sum_{l\in\mathcal{V}_2 \text{ s.t. } kl\in\mathcal{E}_2} y_{ij,kl} \le x_{i,k} + x_{j,k}, \quad \forall ij \in \mathcal{E}_1, \forall k \in \mathcal{V}_2 \tag{4}$$

The final formulation is to minimize the objective function according to Eq. 3 given the constraints of Eqs. 2a, 4 and 2d. Each weakly connected component in every 3D scene graph is matched against all the weakly connected components in all other scenes. Note that source and target graphs are not commutative, i.e., $J(G_1, G_2) \neq J(G_2, G_1)$.

E. False Matches Pruning

This optimization returns matched node pairs with minimum cost, which are used to find matched subgraphs. Three criteria are used to prune falsely matched subgraphs.

1. A threshold T_1 is set so that only pairs of subgraphs with a total matching cost J below this threshold are considered valid matches. The implementation sets T_1 to be proportional to $N = |\mathcal{V}_1| + |\mathcal{E}_1|$, i.e., $T_1 = \alpha N$, because the total cost is a sum over both node and edge costs. A large graph is likely to generate a higher cost than a small one. The empirical ratio α is set to be 0.1.

2. The matching pairs of subgraphs that satisfy criteria 1 are filtered by RANSAC-based point cloud registration [6]. An empirical threshold $T_2 = 0.9$ is set for both precision and recall of the registered point cloud of each pair of matched nodes. If a pair of nodes does not meet this threshold, the nodes will not be merged with other nodes in their respective subgraphs, as shown in Fig. 4.

3. If an object is found to be isolated in a scene, which is performed automatically, then other segments should not be merged with this object. To achieve this, if a source graph $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ is matched with a subgraph S of the target graph G_2, such that $S = (\mathcal{V}', \mathcal{E}'), |\mathcal{V}_1| = |\mathcal{V}'|$, then any

matched subgraph $\hat{S} = (\hat{\mathcal{V}}, \hat{\mathcal{E}}) \subseteq G_2$ that contains all the nodes of S as a proper subset, i.e., $\mathcal{V}' \subset \hat{\mathcal{V}}$, will be filtered.

Once pruning is completed, the remaining nodes in the matched graphs are merged. The merged segments are considered objects, and projected back to each frame of the video as object masks for training. The training process for the network ϕ_2 is the same as in Sec. V-C.

VI. Experimental Results

A. Dataset

Two video datasets, shown in Fig. 5, are used for experiments and evaluation. The first one is a physics-aware photo-realistic video dataset generated by BlenderProc [28]. All the 17 toys from the HomeBrewedDB dataset [29] are selected to create random scenes in simulation. For each scene, 12 unique toys are randomly selected and dropped on a surface. A virtual camera captures a video of each scene by rotating around it with a fluctuating height. In total, 36 videos are collected with 800 frames each. The second dataset is collected in real environments with 20 unique objects. 13 objects among them, such as the "power drill", are from the YCB dataset [30]. Since some of the YCB objects have simple geometries, seven additional objects are included: "lamp", "dumbbell", "toy dinosaur", "toy duck", "Clorox bottle", "detergent", and "toy Charmander". Each of these non-convex objects contains several sub-parts of varying shapes. The real dataset contains 30 videos with 10 unique objects and ∼1000 frames each. The ground-truth segmentation for the real dataset is manually annotated. Background removal in these scenes where objects are resting stably on a tabletop is performed using RANSAC [6] for plane detection and removal of the support surface.

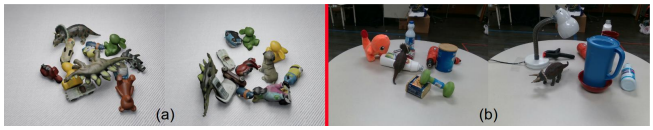

Fig. 5: **Datasets.** (a) A physics-aware photo-realistic synthetic video dataset generated by BlenderProc [28]. (b) A manually collected real dataset with 30 videos and 20 unique objects.

B. Evaluation Metrics

Standard metrics for object segmentation are used, i.e., average precision, recall and Intersection over Union (IoU). The association of predicted and ground-truth segments is performed by using the Hungarian algorithm [31] where the cost of association is based on the IoU of two segments. Ground-truth foreground masks are provided for all methods, and only segmentation of foreground pixels are evaluated. The segmentation results are tested on every 10 frames selected from the testing videos, as neighboring frames in the 30 FPS videos are very similar. The testing frames are selected from videos that are different from the training ones.

C. Baseline Methods

A set of baseline methods are considered for comparison. **A.** DeepLabV3+ [32] is a widely-used segmentation model for RGB images. This baseline is trained with supervision given ground-truth labels, which is considered as an upperbound. Note that the proposed method also uses DeepLabV3+ as a backbone network for pixel-wise feature extraction. **B & C.** UOIS [21] is an RGB-D segmentation method designed for unseen tabletop objects. RICE [22] is a follow-up work over UOIS, which refines segmentation results. For these methods, instead of training from scratch given unlabeled videos, they are trained on a large amount of synthetic data with ground-truth labels. They cannot finetune on new datasets without annotations, and weights provided by the authors are directly used for testing. **D.** LCCP [2] performs segmentation on the object point cloud converted from each depth image and back-project to 2D segmentation. **E.** SD-MaskRCNN [23] is a method for unknown object segmentation similar to UOIS. It is also trained with large simulated data with ground truth but only takes depth images as input. **F.** PiCIE [33] is a state-of-the-art unsupervised semantic segmentation method. **G.** DeepCluster [34] was originally designed for image classification but modified by the author of PiCIE for image segmentation as a baseline. These two methods are trained with foreground RGB images and the background pixels are set to zero. The proposed method, **E**, and **F** are trained from scratch, and tested using 3-fold cross-validation on the unlabeled videos.

	Real Dataset			Synthetic Dataset		
Method	**Prec.**	**Recall**	**IoU**	**Prec.**	**Recall**	**IoU**
A. [2]	0.946	0.920	0.883	0.977	0.932	0.915
B. [21]	0.816	0.727	0.671	0.953	0.867	0.838
C. [21]+[22]	0.869	0.750	0.709	0.961	0.844	0.828
D. [2]	0.893	0.754	0.715	**0.992**	0.806	0.800
E. [23]	0.782	0.773	0.662	0.812	0.751	0.663
F. [33]	0.549	0.525	0.367	0.556	0.614	0.412
G. [34]	0.444	0.477	0.299	0.502	0.526	0.337
Ours	**0.925**	**0.911**	**0.870**	0.938	**0.925**	**0.880**

TABLE I: Segmentation results for different methods. Method A is a supervised solution provided as an upper bound of efficiency. Best results among unsupervised methods in bold.

D. Quantitative Results

Quantitative results of the proposed and baseline methods are shown in Table I. The proposed method works well on both the real and synthetic dataset. It outperforms all the baseline methods except **A**, which was trained with supervision using ground truth labels. Although it may be unfair to compare with methods like **B**, **C** and **E** given that they are designed for unknown object segmentation and are only trained in simulation, it does show that those methods may not work well on arbitrary scenes, especially when the testing objects are out of the training distribution. This work argues that unsupervised learning is useful and necessary on previously seen objects when deploying a robot in real environments. It is a little surprising that the clustering based unsupervised segmentation methods **F, G** performed poorly on this task.

E. Intermediate Point Cloud Segmentation Results

To assess the advantage of using graph matching for object pseudo-label generation, intermediate point cloud segmentation results are provided. The proposed method is compared against LCCP [2] and SymSeg [3] on the reconstructed object point cloud of each video. Intermediate quantitative

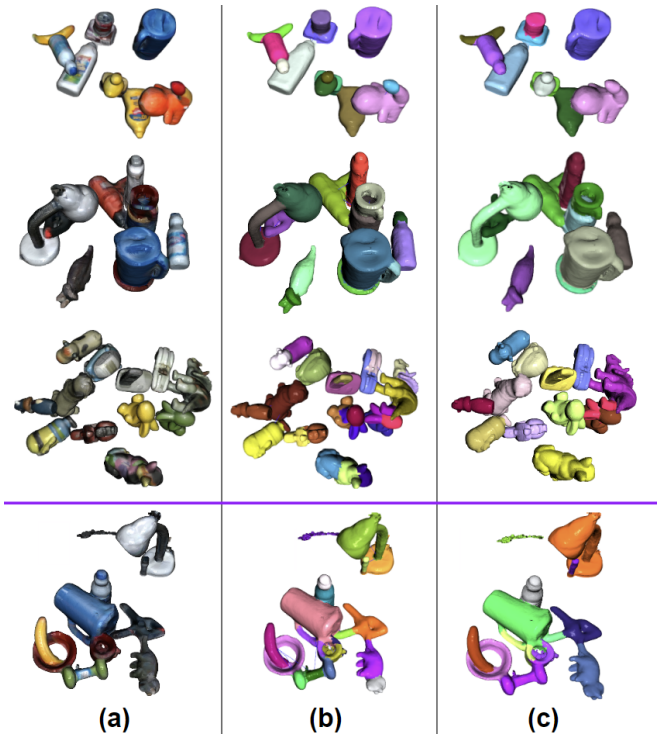

Fig. 6: **Intermediate Qualitative Results of Point Cloud Segmentation.** **(a)** Reconstructed object point clouds in their original color. **(b)** Segmentation results from LCCP + SymSeg. **(c)** Object pseudo labels after graph matching and pruning given over-segmentation results as inputs. A failure case of over-segmentation is shown in the last row, where part of the "pitcher" and "cup" are not separated and the "cup" is segmented into inconsistent pieces. This cannot be addressed during graph matching.

and qualitative point cloud segmentation results are provided in Table II and Fig. 6 respectively. The benefits of combining LCCP with SymSeg are not entirely shown in numbers, as SymSeg sometimes also introduces errors, such as over-segmenting a round cup and its handle. Nevertheless, it makes the 3D segments more consistent for certain cases like in Fig. 3. The incorrectly over-segmented objects, however, can be recovered by the graph matching algorithm. The improvement brought by graph matching is even more pronounced over the more challenging, irregular and non-convex geometries present in the synthetic dataset.

	Real Dataset			Synthetic Dataset		
	Prec.	**Recall**	**IoU**	**Prec.**	**Recall**	**IoU**
[2]	**0.969**	0.858	0.831	0.984	0.771	0.761
[2] & [3]	0.967	0.861	0.833	**0.987**	0.768	0.758
Ours	0.957	**0.964**	**0.923**	0.978	**0.966**	**0.950**

TABLE II: Point Cloud Segmentation Results. "Ours" here corresponds to the object segmentation results achieved after graph matching given the input of over-segmentation from [2] & [3]. Best results in bold.

F. Ablation Study

The results of an ablation study are shown in Table III. All the alternatives use the same feature extractor (DeepLabV3+ [32]) and clustering method (MeanShift [7]). The ablations use different object pseudo-labels for training, and are tested given ground truth foreground masks similar to Table I. **V1.** The first one is pretrained on the COCO dataset [35]. Its final prediction layer is removed and 256-dim pixel features are used for clustering. It has the lowest performance. **V2.** The second one is trained using the output of LCCP [2] without graph matching. Its performance is similar to that of LCCP-based 3D segmentation as expected. **V3.** The third one is trained using pseudo-labels with graph matching, but without pruning. The performance is better than not using graph matching but not as good as the proposed approach since nodes can be incorrectly merged resulting in erroneous object pseudo-labels. **V4.** The last alternative tries to exhaustively register point clouds of connected components without using the candidates from graph matching. If the precision, recall and feature cosine similarity of two registered nodes are greater than 0.9, then these two nodes are considered as a match. This takes significantly more time to find object patterns ($\sim$1.5hr compared to $\sim$10mins of the proposed method for 36 scenes). And since point cloud registration are more likely to fail due to clutter of objects, it underperforms the proposed method.

	Real Dataset			Synthetic Dataset		
Ablation	**Prec.**	**Recall**	**IoU**	**Prec.**	**Recall**	**IoU**
V1.	0.716	0.665	0.547	0.806	0.775	0.665
V2.	0.915	0.815	0.776	0.935	0.746	0.711
V3.	0.904	0.832	0.790	0.934	0.800	0.762
V4.	0.915	0.883	0.842	0.931	0.823	0.782
Ours	**0.925**	**0.911**	**0.870**	**0.938**	**0.925**	**0.880**

TABLE III: Ablation results. Best results in bold.

G. Running Time

Binary linear programming (BLP) is an NP-hard problem, but in practice solutions can be acquired very fast given advanced solvers, such as Gurobi [36], which is used in the companion implementation. The processes of graph matching (Sec.V-D) and pruning (Sec.V-E) on 36 videos can be performed within $10mins$ on a AMD Ryzen 5900 CPU.

VII. Limitations and Conclusion

This work proposes a self-supervised learning pipeline given unlabeled RGB-D videos captured with a moving camera observing static scenes, which is a common scenario for household mobile robots. A graph-matching algorithm is adapted to find object patterns across videos and generate object pseudo-labels for learning. One limitation of the proposed approach is that it relies on the initial object over-segmentation to be consistent across different scenes. This may not always be achieved due to noise despite using the the synergy of LCCP and SymSeg. This can, however, be addressed with more data that cover multiple over-segmentation patterns, or future improvements on the matching algorithm to consider node split and merge operations. Another limitation is that the proposed method cannot handle deformable objects in general, since it requires matched objects to be identical in both texture and geometry. An aspect that the current effort has not explored is the sensitivity of this approach to vastly different lighting conditions, which may be of concern given that the learned object features are RGB-based. The video data were collected in the same environment under slightly differing lighting conditions. A potential avenue for addressing this issue is data augmentation (e.g., randomizing brightness, color jittering, etc.) during training.

REFERENCES

[1] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*. IEEE, 2011, pp. 127–136.

[2] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter, "Object partitioning using local convexity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 304–311.

[3] A. Ecins, C. Fermüller, and Y. Aloimonos, "Seeing behind the scene: Using symmetry to reason about objects in cluttered environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7193–7200.

[4] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, and L. Van Gool, "Unsupervised semantic segmentation by contrasting object mask proposals," in *International Conference on Computer Vision*, 2021.

[5] J. Lerouge, M. Hammami, P. Héroux, and S. Adam, "Minimum cost subgraph matching using a binary linear program," *Pattern Recognition Letters*, vol. 71, pp. 45–51, 2016.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in Computer Vision*, M. A. Fischler and O. Firschein, Eds. San Francisco (CA): Morgan Kaufmann, 1987, pp. 726–740.

[7] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[8] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway, "Clustering on the Unit Hypersphere using von Mises-Fisher Distributions," *Journal of Machine Learning Research*, vol. 6, no. 9, 2005.

[9] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[10] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[11] X. Chen and K. He, "Exploring simple siamese representation learning," *arXiv preprint arXiv:2011.10566*, 2020.

[12] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, *et al.*, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint arXiv:2006.07733*, 2020.

[13] S. Pirk, M. Khansari, Y. Bai, C. Lynch, and P. Sermanet, "Online learning of object representations by appearance space feature alignment," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 473–10 479.

[14] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2794–2802.

[15] X. Wang, K. He, and A. Gupta, "Transitive invariance for self-supervised visual representation learning," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1329–1338.

[16] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2701–2710.

[17] R. Liu, Z. Wu, S. Yu, and S. Lin, "The emergence of objectness: Learning zero-shot segmentation from videos," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] S. Purushwalkam and A. Gupta, "Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases," *arXiv preprint arXiv:2007.13916*, 2020.

[19] M. Schoeler, J. Papon, and F. Worgotter, "Constrained planar cuts-object partitioning for point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5207–5215.

[20] Y. Xiang, C. Xie, A. Mousavian, and D. Fox, "Learning RGB-D feature embeddings for unseen object instance segmentation," *arXiv preprint arXiv:2007.15157*, 2020.

[21] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," *IEEE Transactions on Robotics*, 2021.

[22] C. Xie, A. Mousavian, Y. Xiang, and D. Fox, "Rice: Refining instance masks in cluttered environments with graph neural networks," in *Conference on Robot Learning (CoRL)*, 2021.

[23] M. Danielczuk, M. Matl, S. Gupta, A. Li, A. Lee, J. Mahler, and K. Goldberg, "Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[24] M. Cho, S. Kwak, C. Schmid, and J. Ponce, "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1201–1210.

[25] V. H. Vo, E. Sizikova, C. Schmid, P. Pérez, and J. Ponce, "Large-scale unsupervised object discovery," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 764–16 778, 2021.

[26] X. Qin, Z. Zhang, C. Huang, C. Gao, M. Dehghan, and M. Jagersand, "Basnet: Boundary-aware salient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 7479–7489.

[27] T. Nguyen, M. Dax, C. K. Mummadi, N. Ngo, T. H. P. Nguyen, Z. Lou, and T. Brox, "Deepusps: Deep robust unsupervised saliency prediction via self-supervision," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "BlenderProc," *arXiv preprint arXiv:1911.01911*, 2019.

[29] R. Kaskman, S. Zakharov, I. Shugurov, and S. Ilic, "HomebrewedDB: RGB-D Dataset for 6D Pose Estimation of 3D Objects," *International Conference on Computer Vision (ICCV) Workshops*, 2019.

[30] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-CMU-Berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[31] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[32] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[33] J. H. Cho, U. Mall, K. Bala, and B. Hariharan, "PiCIE: Unsupervised Semantic Segmentation Using Invariance and Equivariance in Clustering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 16 794–16 804.

[34] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep Clustering for Unsupervised Learning of Visual Features," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149.

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

[36] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com