2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Robot-Supervised Learning for Object Segmentation

Victoria Florence[1], Jason J. Corso[1,2], and Brent Griffin[1,2]

***Abstract*— To be effective in unstructured and changing environments, robots must learn to recognize new objects. Deep learning has enabled rapid progress for object detection and segmentation in computer vision; however, this progress comes at the price of human annotators labeling many training examples. This paper addresses the problem of extending learning-based segmentation methods to robotics applications where annotated training data is not available. Our method enables pixelwise segmentation of grasped objects. We factor the problem of segmenting the object from the background into two sub-problems: (1) segmenting the robot manipulator and object from the background and (2) segmenting the object from the manipulator. We propose a kinematics-based foreground segmentation technique to solve (1). To solve (2), we train a self-recognition network that segments the robot manipulator. We train this network without human supervision, leveraging our foreground segmentation technique from (1) to label a training set of images containing the robot manipulator without a grasped object. We demonstrate experimentally that our method outperforms state-of-the-art adaptable in-hand object segmentation. We also show that a training set composed of automatically labelled images of grasped objects improves segmentation performance on a test set of images of the same objects in the environment.**

I. Introduction

Although robots are highly productive in controlled environments, developing robotics algorithms that continue to learn new tasks in changing environments is an open problem. A robust object detector will be indispensable for automation of these tasks, since many industrial and home service tasks require interaction with numerous, ever-changing objects. Object detection has seen a significant gain in performance in the past decade due to deep learning. Learning-based methods outperform handcrafted visual features by taking a data-driven approach to generating features that are more robust for object detection [1], [2]. However, most deep learning-based methods assume that large quantities of annotated training data are available for each type of object [3], [4], which is impractical when robots encounter new objects and tasks. Thus, failing to detect new objects is a limitation of fixed-dataset, learning-based detection and a more general obstacle for robot autonomy.

For robot perception, simply applying dataset-driven detection methods is wasting a useful asset: robots can take action to improve sensing and understanding of their environments [5]. Various approaches have been created to take advantage of robot embodiment to learn object appearances. We follow the paradigm of past in-hand object segmentation works in which robots grasp unknown objects in order to learn their visual appearance [6]–[11]. Most of these methods predate deep learning and require a human to design a visual model or other visual heuristics for recognizing the robot manipulator. Notably, humans have to redesign these models if there are physical changes to the robot or a new robot is deployed. Work by Browatzki et al. [6], which we compare against, and this paper are the only methods we are aware of without this requirement.

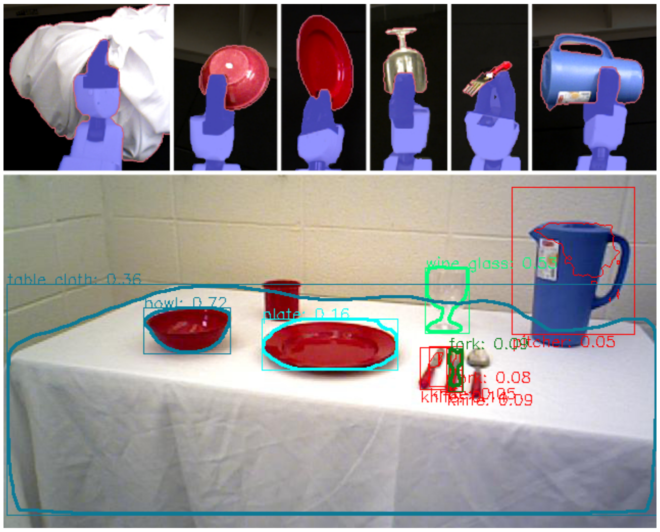


Fig. 1: Our method produces pixelwise annotations of grasped objects (top). These annotations can be used to improve performance of object instance segmentation methods (bottom). The method adapts to new environments, objects, and robotic platforms without human supervision.

This paper introduces a method called robot supervision that automatically generates object segmentation training data through robot interaction with grasped objects. In this way, we enable robots to continue improving their own vision systems over time. Using only the robot's kinematic link coordinate frames and an RGB-D camera, we segment a grasped object and the manipulator from the background using our kinematics-based foreground segmentation. We then separate the robot manipulator from the object using a deep Convolutional Neural Network (CNN) called a Self-Recognition Network (SRN), leaving only the in-hand object (see example in Figure 2). Notably, the robot annotates its own training data for the SRN using our kinematics-based foreground segmentation; thus, the SRN can be retrained autonomously. The end result is a method for generating object labels (like those shown in Figure 1) that is generalizable to many existing robot platforms without human supervision.

[1] Robotics Institute, University of Michigan
[2] Electrical Engineering and Computer Science, University of Michigan
{vflorenc,jjcorso,griffb}@umich.edu

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

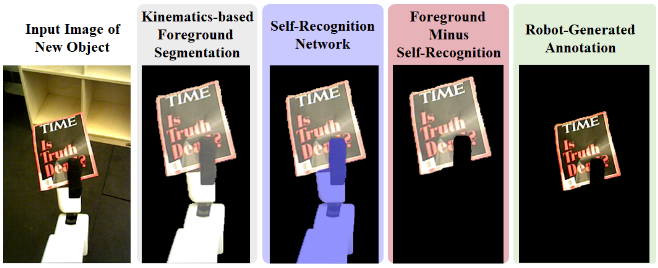

Input Image of New Object
Kinematics-based Foreground Segmentation
Self-Recognition Network
Foreground Minus Self-Recognition
Robot-Generated Annotation

Fig. 2: An overview of our method. After collecting images of an object (left), the robot uses encoder readings and active depth sensing to segment the foreground (manipulator or manipulator + grasped object) of each image (middle-left, grey). Using its self-recognition network (middle, blue), the robot isolates the object from the rest of the foreground (middle-right, red). Thus, the robot generates densely-labeled annotations of newly encountered objects (right).

To test our method, we collect and annotate a new dataset that contains RGB-D images of our robot manipulator with 30 in-hand objects (20 images each, 600 total). We evaluate our method and that of Browatzki et al. [6] on this dataset and show that our method achieves a 27 point (or 75%) mIoU improvement over the baseline method. Finally, we fine-tune an object instance segmentation framework [2] on data produced by our method. The fine-tuning improves object segmentation from 38.1 AP to 49.8 AP on a test set of images. An example of our results and a test image are shown in Figure 1. We provide our source code and data at https://github.com/vflorence/RSLOS.

II. Related Work

A. Interactive Object Segmentation

In interactive object segmentation methods, robot actions or environment continuity are used in order to segment the object from the rest of the image. Some methods learn objects that are placed in uncluttered or known environments [13]–[15]. Other works rely on scene change over time, calling anything that violates the static scene assumption an object [16]–[20]. They rely on the non-guaranteed movement of objects. Other methods in this category push objects to create movement in the scene and group pixels that move together into object labels [21]–[23]. These methods have more control over object movement, but they require a work surface and objects that permit pushing. The benefit of these methods is that they can segment many objects at the same time; however, they do not allow for full pose or environmental control.

B. In-hand Object Segmentation

In-hand object segmentation methods use robot encoder feedback to locate a grasped object and various methods to reason about robot-object occlusion. Omrčen et al. [9] incorporate many data sources such as color distribution, a disparity map, and a pretrained Gaussian Mixture Model (GMM) of the hand to segment unknown objects, but their method does not extract pixelwise object labels. Welk et al. [10] use Eigen-backgrounds, disparity mapping, and tracking on a model of the robot manipulator to isolate objects. Krainin et al. [8] take a self-recognition based approach to object learning by matching a robot manipulator to its 3D mesh model in order to isolate and model an in-hand object. However, their method requires a 3D geometric model of the robot manipulator and focuses on modeling non-deformable objects from multiple viewpoints. Browatzki et al. [6], [7] use a GMM trained on pixel values around a bounding box to isolate a grasped object. This method focuses on viewpoint selection and data association, but their object isolation technique is not robust to pixel-level similarity between the object, background, and robot. While these systems are able to isolate unseen objects for visual learning and oftentimes model the occlusion of the robot manipulator by these objects, they are limited by the need for custom manipulator models, environment-specific heuristics, and parameter tuning.

C. Self-supervised Manipulator Recognition

To the best of our knowledge, da Costa Rocha et al. [24] is the first method to "make use of the kinematic model of a robot in order to generate training labels." This work learns pixelwise self-recognition of the da Vinci surgical robot with unsupervised training labels much like our SRN. da Costa Rocha's method uses a projection of a full geometric model of the manipulator into an RGB image to generate the labels, while our method requires depth sensor readings and only uses link coordinate frames (e.g., in Figure 3). The benefit of our approach is that we learn self-recognition without a full model of the robot and adapt to changes in robot hardware automatically. Additionally, the end goal of our method is in-hand object isolation for object learning, while da Costa Rocha's work focuses on robotic self-recognition.

III. Kinematics-based Foreground Segmentation

A. Kinematics-based Depth Segmentation

We begin segmenting the robot's manipulator from background by over-segmenting the head camera's depth image. We use $D \in \mathbb{R}^{H \times W}$ to represent the depth image, where H and W are the numbers of rows and columns in D. $D(i,j) \in \mathbb{R}$ is the measured depth in millimeters for the pixel at row i, column j. Using the graph-based image segmentation of Felzenswalb [25], we define the depth-image segmentation

$$S(D) := \{s_1, s_2, \dots, s_n\}, \tag{1}$$

where S is exhaustive of the 2D coordinates in D with mutually exclusive subsets.

Next, we use the robot's encoder readings and kinematic model to get approximate 3D coordinates of link positions and project them into the depth image. We take the 3D link position of each ith link relative to the camera's coordinate frame, $\mathrm{P}_i := [\mathrm{x}_i, \mathrm{y}_i, \mathrm{z}_i]^\top$, and find the projected depth-image coordinates using the transform

$$[p_i, 1]^\top := \lfloor \frac{\mathbf{K}\mathrm{P}_i}{\mathrm{z}_i} + 0.5 \rfloor = [h_i, w_i, 1]^\top, \tag{2}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is the head camera's intrinsic camera matrix.

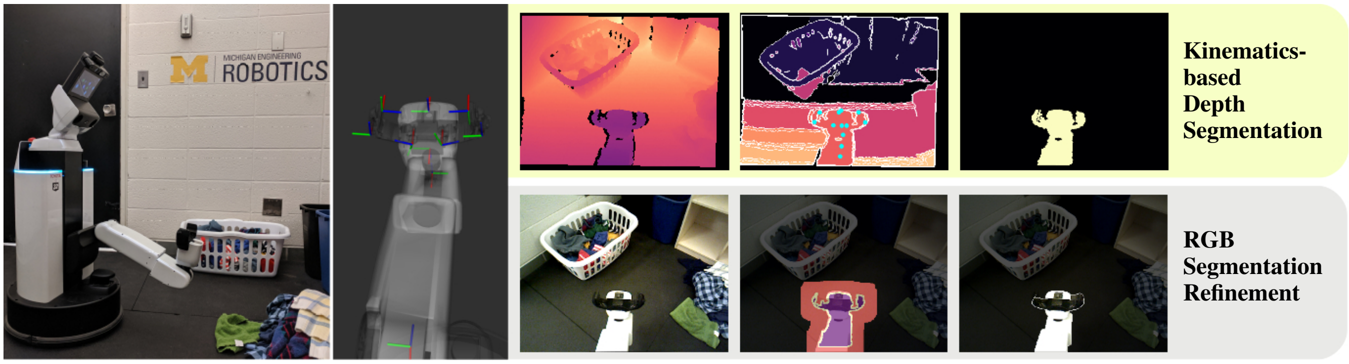

Kinematics-based Depth Segmentation
RGB Segmentation Refinement

Fig. 3: In our kinematics-based foreground segmentation method, we use link coordinate frames from the kinematic model of the robot to localize the robot manipulator (left). Using the RGB-D camera's depth channel (top, left), we over-segment the image. We then project manipulator coordinates into the image (top, middle) and label the segments containing filtered projected coordinates as foreground (top, right). We refine the segmentation with the RGB channels (bottom, left), initializing GrabCut [12] with a depth segmentation-based mask (bottom, middle) to give us our final output (bottom, right).

Using the depth segmentation (1) and projected kinematic points (2), we define our initial foreground segmentation

$$s_{\text{fg}} := \bigcup \{s_j | \exists p_i \in s_j, D(h_i, w_i) \leq \text{z}_i + \lambda\}, \tag{3}$$

where $D(h_i, w_i)$ is the depth measurement corresponding to p_i and λ is a noise threshold for z_i. Put simply, if the depth sensor reading ($D(h_i, w_i)$) of a projected link location (p_i) is within λ (distance) past its expected kinematic location (z_i), the segment (s_j) containing that reading is added to the initial foreground segmentation (s_{fg}).

B. RGB Segmentation Refinement

We refine the initial foreground segmentation using matrix operations. We represent s_{fg} (3) as matrix $\mathbf{M}_0 \in \mathbb{R}^{H \times W}$ s.t.

$$\mathbf{M}_0(i,j) := \begin{cases} 1 & D(i,j) \in s_{\text{fg}} \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where $\mathbf{M}_0(i,j) = 1$ is a foreground element. We post-process $\mathbf{M}_0$ by filling in all holes then performing a morphological binary opening operation with the kernel $\mathbf{J}_8$ where $\mathbf{J}_N \in \mathbb{R}^{N \times N}$ and every element is equal to one. These operations reduce the effect of depth sensor noise, which often manifests as holes and noisy object edges.

Using the processed $\mathbf{M}_0$, we generate two more matrices

$$\mathbf{M}_{\text{p}} := \text{erosion}(\mathbf{M}_0, \mathbf{J}_{10}), \tag{5}$$

$$\mathbf{M}_{\text{r}} := \text{dilation}(\mathbf{M}_0, \mathbf{J}_{75}), \tag{6}$$

where $\mathbf{M}_{\text{p}}$ is precision oriented, and $\mathbf{M}_{\text{r}}$ is recall oriented. Note that $\mathbf{M}_{\text{p}} = 1 \implies \mathbf{M}_0 = 1$, and $\mathbf{M}_0 = 1 \implies \mathbf{M}_{\text{r}} = 1$. The erosion and dilation operations soften the boundary created by the depth segmentation to account for depth sensor noise and any misalignment between the RGB and depth images.

We generate our final foreground segmentation using a GrabCut segmentation [12] on the RGB image $I \in \mathbb{R}^{H \times W}$ corresponding to D. Using $\mathbf{M}_{\text{p}}$, $\mathbf{M}_{\text{r}}$, and the processed $\mathbf{M}_0$, we initialize the GrabCut segmentation using

$$\mathbf{M}_{\text{gc}} := \mathbf{M}_{\text{r}} + \mathbf{M}_0 + \mathbf{M}_{\text{p}}, \tag{7}$$

where $\mathbf{M}_{\text{gc}}(i,j) \in \{0,1,2,3\}$, 0 is background, 1 is probably background, 2 is probably foreground, and 3 is foreground. We refine $\mathbf{M}_{\text{gc}}$ using GrabCut for 8 iterations and convert the refined $\mathbf{M}_{\text{gc}}$ to the binary mask

$$\mathbf{M}_{\text{fg}}(i,j) := \begin{cases} 0 & \mathbf{M}_{\text{gc}}(i,j) \in \{0,1\} \\ 1 & \mathbf{M}_{\text{gc}}(i,j) \in \{2,3\} \end{cases}. \tag{8}$$

$\mathbf{M}_{\text{fg}}$ is the final kinematics-based foreground segmentation mask, where $\mathbf{M}_{\text{fg}}(i,j) = 1$ indicates that $I(i,j)$ corresponds to the robot's manipulator in the foreground (see example in Figure 3).

IV. Robot-supervised Self-recognition Network

Using the kinematics-based foreground segmentation described in Section III, we enable the robot to collect and label its own data to train an SRN that labels instances of the robot manipulator in an image. Specifically, the robot performs foreground segmentation on images where its manipulator is the only object in the foreground (i.e., no objects are grasped) and creates a manipulator annotation from the foreground mask.

To diversify training data for the SRN, we collect images of the robot manipulator in various poses, permuting across all combinations of individual joint positions. Poses are uniformly distributed across each joint's range, and the number of positions per joint can be set as a parameter to match the time available for learning. For each pose we 1) position the camera such that the robot gripper is approximately centered in the image and 2) command the non-varied manipulator joints to a position that puts the gripper beyond the depth camera's minimum range. The data collection joint configuration can be adjusted to fit any robot platform, as long as these two requirements are met.

We use $\mathbf{M}_{\text{fg}}$ (8) to label each manipulator image as a training example, where the background class ID is 0 and the manipulator class ID is 1.

Inspired by work on domain randomization for sim-to-real transfer learning [26], [27], we perform dataset augmentation to close the gap between the automatically labeled

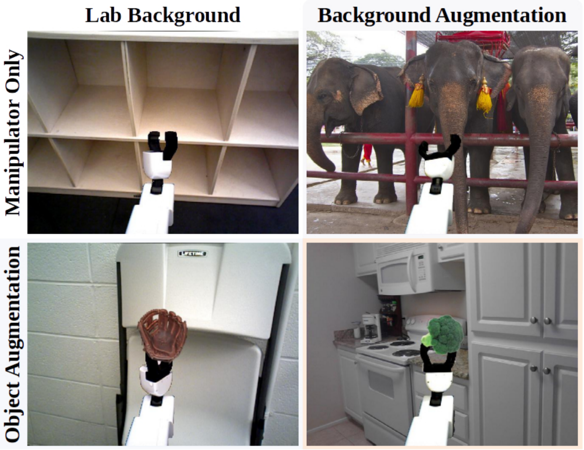


Fig. 4: In the self-recognition network training set, we augment manipulator images with background images taken by the robot (left) and from the COCO dataset (right) as well as foreground objects (bottom). These augmentations increase the diversity of the background and manipulator classes, respectively, and increase the number of example images in the self-recognition network training set.

manipulator-only images and future test-time images with objects in-hand. We did this with background substitution and foreground object augmentation. Examples of these augmentations can be seen in Figure 4.

We do background substitution with background images taken from the popular Common Objects in Context (COCO) dataset [3] as well as pictures taken by the robot without the manipulator in view.

We perform foreground superposition with object classes in the COCO dataset [3]. For each foreground augmentation, we randomly scale and rotate an alpha-matted object image and overlay it at the center of the image.

The background and foreground augmentations can increase the number and type of objects sampled in the training background class, while the foreground augmentations randomly alter manipulator appearance to imitate robot-object occlusion. Increasing dataset diversity and size reduces overfitting and improves the robustness of the learned manipulator segmentation.

We use an object instance segmentation framework, Mask R-CNN, pretrained on the COCO dataset with R-50-FPN backbone as the SRN model. We use the standard multi-task loss from the original paper [2].

V. Robot-supervised Object Annotation

In order to annotate in-hand objects, we repeat the data collection procedure from Section IV with objects grasped by the robot manipulator. We apply our kinematics-based foreground isolation method to create $\mathbf{M}_{\text{fg}}$ (8) for each image (see example in Figure 5). The SRN is applied to predict the manipulator location. In cases where the SRN returns multiple predictions, we take the prediction with the highest score. We call the mask representing the SRN prediction $\mathbf{M}_{\text{SRN}}$. To create the final object labels, $\mathbf{M}_{\text{fg}}$ and $\mathbf{M}_{\text{SRN}}$ are combined for each image as

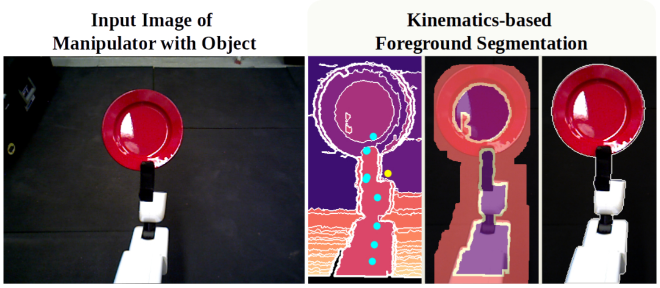


Fig. 5: We use kinematics-based foreground segmentation to segment the gripper and object from the background. The output (right) is combined with the SRN output to create robot-supervised object annotations.

$$\mathbf{M}_{\text{object}} = \mathbf{M}_{\text{fg}} \bigcap \neg \mathbf{M}_{\text{SRN}}. \tag{9}$$

We then perform a final opening on $\mathbf{M}_{\text{object}}$ with $\mathbf{J}_3$, a matrix of ones, for noise removal. Examples of this process can be seen in Figure 6, where $\mathbf{M}_{\text{SRN}}$ is shaded blue and $\mathbf{M}_{\text{object}}$ 9 is outlined in red.

VI. Experiments

A. Experimental Setup

All experiments use the Toyota Human Support Robot (HSR) [28]. Images are gathered with its RGB-D head camera. In the object learning experiments, the robot begins with objects grasped; recent works such as [29] and [30] demonstrate viable methods for unknown object grasping. We use a Pytorch implementation of Mask R-CNN for the SRN and object re-recognition networks [31], [32].

B. Metrics

On the grasped object annotation task, we use pixelwise mIoU as our metric. For each image, we create a ground-truth object annotation that is compared to our method's output as well as the baseline. We calculate a standard, per-image, Intersection over Union (IoU) metric for the object masks and average the results for each class to get a class mean IoU (mIoU). The overall mIoU is an average of all class mIoUs.

On the object re-recognition in context task, we use the COCO API Average Precision (AP) detection metrics for evaluating performance on this task [3]. AP is calculated as the area under the precision recall curve. We provide results for different IoU and pixels-per-object thresholds. For more details about these metrics, we recommend looking at the COCO detection task evaluation metric [3].

The difference in metric is motivated by the fact that the output for the first task is a single mask with labels for each pixel, while the output for the second task can include multiple detections per image region.

C. Grasped Object Annotation Performance

To quantify the performance of our robot-supervised object annotation method, we gather a test set of 600 human-annotated images of the manipulator with an object grasped.

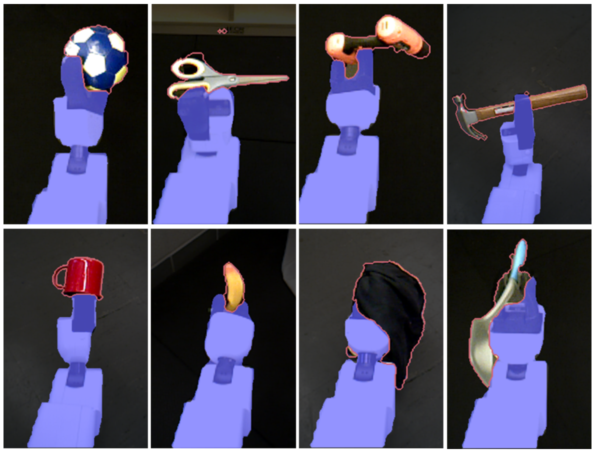


Fig. 6: Some of our method's best qualitative results. The areas shaded blue are labeled as manipulator by the SRN, while the red-outlined areas are the object labels. These pixelwise segmentations of in-hand objects can be used to train data-driven object detectors.

For each of 30 objects from the Yale-CMU-Berkeley (YCB) standardized object dataset [33] we collect 20 images of the grasped object. The object viewing poses are uniformly sampled from the joint spaces of two revolute joints in what could be called the "wrist" of the robot; this configuration allows for coverage of the entire viewing sphere of the grasped object without consideration of robot-object occlusion. Due to robot-object occlusion, it is not possible to view the entire object from a single grasp, and the object is not guaranteed to be visible in every image.

For our method, we use Felzenswalb segmentation parameters of σ = 0.5 and k = 800, which are approximately scaled by image size from the parameter settings in the original paper [25]. Additionally, we set the depth segmentation noise threshold to $\lambda = 200$ mm. The SRN used in our experiments is trained and validated on 208 and 89 images of HSR's manipulator, respectively. We annotate all images automatically with the kinematics-based foreground segmentation method (Section III) and augment them with the methods described in Section IV. The number of background and foreground images that we use is shown in Table I. For all SRN training, we follow the Detectron solver scheduling rules [34].

1) Ablation of Data Augmentation: To show the contribution of each type of augmentation in our SRN dataset creation method, we train multiple SRNs, each using a different combination of data augmentations. We repeat each type of data augmentation three times per the most plentiful augmentation resource involved, repeating resources with fewer elements as necessary. For example, we have more backgrounds than gripper images, so to create the "BG" dataset, we use each background 3 times and repeat the gripper images as necessary. The number of images in each augmented data split is shown in Table II.

Results from the ablative SRNs show that foreground (FG) and background (BG) augmentations add 2.9 and 4.7 point improvements, respectively, to our SRN performance as shown in Table III. The combined foreground, background augmentation add a 6.5 point improvement. Results show that the object recall scores consistently benefit from augmentations. Improved recall scores indicate that there are fewer false negatives on the object labeling task (i.e. fewer false positives by the SRN).

TABLE I: Data Collection

Data	Total	Train	Val
Gripper Images	297	208	89
Background Images (COCO/Lab)	1800 (1746 / 54)	1260 (1220 / 40)	540 (526 / 14)
Foreground Objects	80	56	24

TABLE II: Augmented Datasets

Dataset	Total
Orig - original images	297
FG - foreground augmentation	891
BG - background augmentation	5400
FGBG - foreground, background augmentation	5400
Ours - all data	11988

By adding data examples through our data augmentation methods, we are able to increase the precision of the SRN detector and achieve higher recall on object segmentation.

2) Robot-Supervised Object Annotation Performance: We compare our method to the prior work of Browatzki et al. [6]. Their work focuses on efficient viewpoint selection for object learning and included an in-hand object segmentation method. Similarly to our method, it does not require human-designed vision heuristics for reasoning about robot-object occlusion. The method trains a Gaussian Mixture Model on the pixels within a frame around the expected object location (i.e. between inner and outer bounding boxes). We re-implement their segmentation method and choose parameters based on a parameter sweep over the test set. (bounding box sizes = (270,300), number of Gaussians = 1, threshold = 1E-15).

The overall mIoU results in Table III indicate that our method outperforms the baseline on the task of object segmentation. Additionally, our method outperforms [6] on the majority of objects as shown in Figure IV. Examples of segmentation results for our method are shown in Figure 6. In order to gain insight on the performance of each method, we look at the performance along the axes of pixels per object and average saturation. Data regarding the relationship between object size and method performance can be viewed in Figure 7a, while method performance versus object saturation is shown in Figure 7b. Note that our robot platform, HSR, is black and white, so saturation is a pixel-based metric representing robot-object visual similarity.

Overall, this experiment indicates increased robustness of our method over the prior work on in-hand object segmentation without human-designed, robot-specific heuristics for reasoning about robot-object occlusion. Our method enables us to apply advances in deep learning based object-segmentation without human annotation and demonstrates improved accuracy over the pixel-based method in Browatzki et al. [6].

TABLE III: Method Comparison and Ablation Study

Method	mIoU	Precision	Recall
Orig	0.431	0.788	0.477
FG	0.460	0.780	0.516
BG	0.478	0.784	0.531
FGBG	0.506	0.795	0.563
Ours	**0.639**	**0.823**	**0.716**
Browatzki et al. [6]	0.362	0.685	0.400

TABLE IV: Object Segmentation Performance (mIoU)

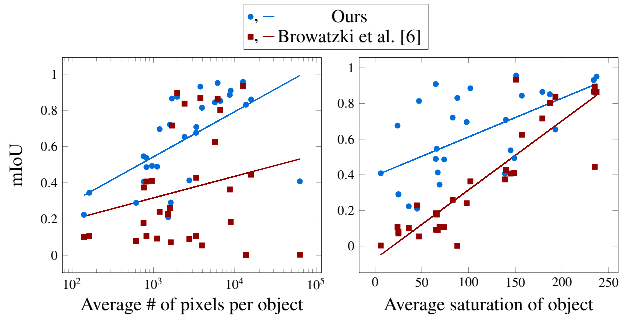
Object	Browatzki et al. [6]	Ours
Pitcher	0.934	**0.957**
Bowl	0.864	**0.951**
Mug	0.867	**0.931**
Wood	0.184	**0.909**
Magazine	0.363	**0.885**
Apple	**0.895**	0.875
Brick	0.716	**0.865**
Plate	0.445	**0.860**
Soccer ball	0.802	**0.852**
Power drill	0.625	**0.844**
Tshirt	0.002	**0.831**
Wine glass	0.054	**0.814**
Scissors	0.260	**0.721**
Hammer	0.428	**0.708**
Screwdriver	0.240	**0.696**
Rope	0.106	**0.676**
Banana	**0.837**	0.654
Padlock	0.177	**0.546**
Fork	0.407	**0.537**
Spoon	0.411	**0.493**
Wrench	0.092	**0.489**
Expo marker	0.107	**0.486**
Spatula	0.090	**0.413**
Tablecloth	0.003	**0.408**
Knife	0.374	**0.406**
Keys	0.106	**0.345**
Baseball	0.071	**0.291**
Golf ball	0.079	**0.289**
Dice	0.101	**0.223**
Sponge	**0.228**	0.210
Averaged	0.362	**0.639**

D. *Object Re-recognition in Context*

In order to analyze the usefulness of the labels produced by our method, we compare a state-of-the-art object instance segmentation framework trained on dataset images to one that is fine-tuned on outputs from our method. The same model, initialization, and training schedules are used as in the SRN training procedure for both networks. As a baseline, we train Mask R-CNN on a subset of the Large Vocabulary Instance Segmentation dataset v0.5 (LVIS v0.5) by Gupta et al. [4] that has 25 classes in common with our selected YCB objects (excluding brick, dice, golfball, rope, and wood). We reduce the LVIS v0.5 dataset to the images containing any of the 25 YCB objects and remove all other classes from the annotations, maintaining the training and validation splits. For our method, we collect and automatically label 20 additional images per object for the 25 overlapping classes. We then fine-tune the LVIS-trained network on our method's output labels, leaving out approximately 8% of the object images as a validation set. Results in Table V show that our method outperforms the dataset baseline on all metrics. This result indicates that despite the noise we observe in the in-hand object segmentation results in Figure IV, the limited training examples of in-hand objects are useful for training a deep CNN to recognize the same objects in different contexts.

TABLE V: Object Instance Segmentation Performance

Dataset	AP	AP$^{0.50}$	AP$^{0.75}$	APs	APm	APl
Bounding Box						
LVIS	39.2	51.0	45.6	35.5	44.3	62.3
LVIS + fine-tuning on our data	49.9	70.8	57.6	50.6	57.3	63.3
Segmentation						
LVIS	38.1	50.9	45.1	30.8	42.4	61.3
LVIS + fine-tuning on our data	49.8	69.7	54.7	36.6	57.0	65.4



(a) (b)

Fig. 7: We examine our method's performance along two pixel-based metrics of image content. On the axis of pixels per object mask, we observe that the baseline method fails on very large objects, and that the performances of both methods decrease on small objects (left). Additionally, we observe that our method's performance was slightly less correlated with pixel-level similarity to the manipulator (right).

VII. Conclusions

We have presented and experimentally validated robot supervision that enables robots to generate new annotations using in-hand object segmentation. Our method of kinematics-based foreground segmentation followed by a robot-supervised SRN achieves significant improvement over the baseline on the task of in-hand object segmentation. Our method performs well on a wide variety of objects and is not specific to a single robot platform. Additionally, experiments indicate that fine-tuning an object instance segmentation framework on labels created by our method improves performance on object segmentation in context. Using our method, robots can generate their own training data and learn to better segment new objects and environments without human supervision.

Acknowledgment

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[2] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[3] 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 *The European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.

[4] A. Gupta, P. Dollar, and R. Girshick, “Lvis: A dataset for large vocabulary instance segmentation,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[5] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, “Interactive perception: Leveraging action in perception and perception in action,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, Dec. 2017.

[6] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven, “Active object recognition on a humanoid robot,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 2021–2028.

[7] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven, “Active in-hand object recognition on a humanoid robot,” *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1260–1269, Oct. 2014.

[8] M. Krainin, P. Henry, X. Ren, and D. Fox, “Manipulator and object tracking for in-hand 3d object modeling,” *The International Journal of Robotics Research (IJRR)*, vol. 30, no. 11, pp. 1311–1327, 2011.

[9] D. Omrcen, A. Ude, K. Welke, T. Asfour, and R. Dillmann, “Sensorimotor processes for learning object representations,” in *IEEE-RAS International Conference on Humanoid Robots*, Nov. 2007, pp. 143–150.

[10] K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann, “Autonomous acquisition of visual multi-view object representations for object recognition on a humanoid robot,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 2012–2019.

[11] A. Venkataraman, B. Griffin, and J. J. Corso, “Kinematically-informed interactive perception: Robot-generated 3d models for classification,” 2019, arXiv:1901.05580.

[12] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004.

[13] A. Zeng, K. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao, “Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1386–1383.

[14] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox, “A joint model of language and perception for grounded attribute learning,” in *International Conference on Machine Learning (ICML)*, 2012.

[15] J. Vasquez-Gomez, L. Sucar, and R. Murrieta-Cid, “View planning for 3d object reconstruction with a mobile manipulator robot,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2014.

[16] E. Herbst, P. Henry, X. Ren, and D. Fox, “Toward object discovery and modeling via 3-d scene comparison,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 2623–2629.

[17] E. Herbst, P. Henry, and D. Fox, “Toward online 3-d object segmentation and mapping,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 3193–3200.

[18] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, “Toward lifelong object segmentation from change detection in dense rgb-d maps,” in *European Conference on Mobile Robots (ECMV)*, Sept. 2013, pp. 178–185.

[19] T. Fäulhammer, R. Ambruş, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt, and M. Vincze, “Autonomous learning of object models on a mobile robot,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 2, no. 1, pp. 26–33, Jan. 2017.

[20] J. Modayil and B. Kuipers, “Bootstrap learning for object discovery,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, Sept. 2004, pp. 742–747 vol.1.

[21] H. van Hoof, O. Kroemer, and J. Peters, “Probabilistic segmentation and targeted exploration of objects in cluttered environments,” *IEEE Transactions on Robotics*, vol. 30, pp. 1198–1209, Oct. 2014.

[22] D. Schiebener, J. Morimoto, T. Asfour, and A. Ude, “Integrating visual perception and manipulation for autonomous learning of object representations,” *Adaptive Behavior*, vol. 21, no. 5, pp. 328–345, 2013.

[23] D. Schiebener, A. Ude, and T. Asfour, “Physical interaction for segmentation of unknown textured and non-textured rigid objects,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 4959–4966.

[24] C. da Costa Rocha, N. Padoy, and B. Rosa, “Self-supervised surgical tool segmentation using kinematic information,” IEEE International Conference on Robotics and Automation (ICRA), 2019, to be published.

[25] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International Journal of Computer Vision (IJCV)*, vol. 59, no. 2, pp. 167–181, Sept. 2004.

[26] J. Tobin, R. H. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[27] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. T. Birchfield, “Deep object pose estimation for semantic robotic grasping of household objects,” in *Conference on Robot Learning (CoRL)*, 2018.

[28] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, “Development of human support robot as the research platform of a domestic mobile manipulator,” *ROBOMECH Journal*, vol. 6, no. 1, p. 4, Apr 2019.

[29] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3406–3413.

[30] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, “Learning ambidextrous robot grasping policies,” *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[31] F. Massa and R. Girshick, “maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch,” https://github.com/facebookresearch/maskrcnn-benchmark, 2018.

[32] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.

[33] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, “Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set,” *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 36–52, Sept. 2015.

[34] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, “Detectron,” https://github.com/facebookresearch/detectron, 2018.