Push-to-See: Learning Non-Prehensile Manipulation to Enhance Instance Segmentation via Deep Q-Learning

Baris Serhan[1*], Harit Pandya[2], Ayse Kucukyilmaz[1], and Gerhard Neumann[3]

Abstract— **Efficient robotic manipulation of objects for sorting and searching often rely upon how well the objects are perceived and the available grasp poses. The challenge arises when the objects are irregular, have similar visual features (e.g., textureless objects) and the scene is densely cluttered. In such cases, non-prehensile manipulation (e.g., pushing) can facilitate grasping or searching by improving object perception and singulating the objects from the clutter via physical interaction. The current robotics literature in interactive segmentation focuses solely on isolated cases, where the central aim is on searching or singulating a single target object, or segmenting sparsely cluttered scenes, mainly through matching visual futures in successive scenes before and after the robotic interaction. On the other hand, in this paper, we introduce the first interactive segmentation model in the literature that can autonomously enhance the instance segmentation of such challenging scenes as a whole via optimising a Q-value function that predicts appropriate pushing actions for singulation. We achieved this by training a deep reinforcement learning model with reward signals generated by a Mask-RCNN trained solely on depth images. We evaluated our model in experiments by comparing its success on segmentation quality with a heuristic baseline, as well as the state-of-the-art Visual Pushing and Grasping (VPG) model [1]. Our model significantly outperformed both baselines in all benchmark scenarios. Furthermore, decreasing the segmentation error inherently enabled the autonomous singulation of the scene as a whole. Our evaluation experiments also serve as a benchmark for interactive segmentation research.**

I. INTRODUCTION

Robotic object manipulation tasks such as sorting, searching or singulation often require a visual understanding of the objects in the task space. Although the state-of-the-art computer vision (CV) algorithms like Mask R-CNN are highly capable in object segmentation and classification [2], their performance decreases when the scene is complex and cluttered due to the presence of occlusions. Similarly, even the human eye might not detect an object when it is hidden under a cluttered heap. On the other hand, unlike CV algorithms, human intelligence can manipulate the clutter based on its prior knowledge of objects to see the invisible. This ability is a result of embodied cognition where perception and action modalities are tightly coupled for learning. Coupling robotic manipulation with visual perception would also enable robots to learn how to tackle complex scenes and facilitate object detection in densely cluttered heaps.

Although the idea of using robotic manipulation and interaction to segment objects can be seen in the early

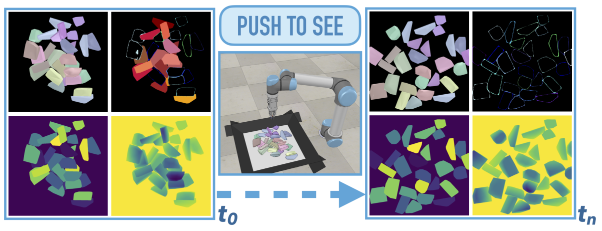


Fig. 1: **An example of scenes before and after the interactions.** The initial heap at t_0 and the singulated heap after conducting n pushes with the Push-to-See model at t_n. Four images at t_0 and t_n in row-order from top left to bottom right: RGB, Segmentation Error, Ground Truth, Depth. The objects which cannot be detected (i.e. false negatives) are coloured in red tones in segmentation error images. In this example, all false negatives at t_0 become true positives at t_n after cluttered objects are singulated with the learned pushing policy.

robotics literature [3], thorough studies on improving visual perception via interactive strategies start from the beginning of the century [4], [5]. These approaches rely upon classical CV techniques such as template matching and optic flow to track the same objects between successive interactions. The figure-ground separation through object or motion features is still the most common approach in the current interactive segmentation literature [6], [7], [8]. However, these approaches face challenges for textureless objects and dense clutters commonly occurring in industrial scenario, since the features becomes difficult to distinguish.

Learning how to push is an essential problem for robotics when the goal is to autonomously search or grasp objects within a dense heap. Non-prehensile manipulation allows objects to be visible or graspable by singulating them from the clutter. Whilst majority of the current robotics literature focuses solely on segmenting/singulating a single target object or works in sparsely populated scenes [7], [9], [10], in this paper, we provide a solution to a more complex problem: the segmentation of all the objects in a dense clutter via learning a non-prehensile manipulation policy. In this work, we refer to a clutter when there are more than 28 objects that are closely piled together in multiple planes and cause partial or full occlusions.

To solve this problem, we suggest a two-fold model that consists of a Deep Q-Network [11] to predict non-prehensile interactions through trial and error (Push-DQN), and a reward generator (Mask-RG), which involves a Mask-RCNN [2] trained solely with depth images, to reinforce the best pushing actions during training. An example of how our model, Push-to-See, changes the scenes before and after manipulation is shown in Figure 1. Our quantitative results

*Corresponding author: baris.serhan@nottingham.ac.uk
[1]School of Computer Science, University of Nottingham, UK
[2]Toshiba Research, Cambridge, UK
[3]Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Germany

indicate a substantial improvement in achieving complete scene singulation with 75% success against baselines with $\sim 7\%$. (see Table I).

The contributions of our work could be summarised as follows:

- We present a novel self-supervised approach for improving instance segmentation through interactions. To the best of our knowledge, we are the first to showcase successful segmentation of dense clutter of over 28 objects.
- Our method inherently enables the autonomous singulation of the scene as a whole, thus almost all objects in the clutter become easily graspable at the end of the task.
- We provide a testing algorithm together with simulation scenes, evaluation metrics, baselines and trained models, which can be used together as a benchmark for interactive segmentation research (`https://github.com/ALRhub/push-to-see.git`).

II. RELATED WORK

In this section, we review existing body of work in the field of instance segmentation and its application to other robotics domains such as grasping, searching and pose estimation.

Instance segmentation For robotics applications such as manipulation or obstacle avoidance, it is crucial to identify all the objects in a given scene. The problem of identifying all the objects in a given scene is referred to as instance segmentation. Recent works in computer vision literature take a top-down approach towards instance segmentation: They employ deep neural networks to generate several proposals for objects then classify them as object instances in a supervised [2] or weakly supervised fashion [12]. While texture information is crucial for segmenting different objects, several robotic tasks require the handling of textureless objects, such as industrial parts and rubble [13]. A few recent approaches learn instance segmentation only from depth information of irregular shaped objects from a single depth image [14] or point clouds [15] for cluttered bin picking scenarios. These approaches learn to extract segmentation boundaries from depth cues, however for arbitrarily shaped unseen objects under severe occlusions additional information is required to infer the correct object boundaries. In this paper, we interact with the environment through pushing actions for obtaining additional information about object boundaries thus improving the instance segmentation.

Interactive segmentation: Segmenting all objects for a complex scene using a single image could be challenging. Thus, for improving the segmentation performance, a few approaches consider segmentation on a video [16] or actively selecting a viewpoint [17]. The advantage of using robots for scene segmentation is the added ability to manipulate the scene through actions such as pushing or picking, a.k.a. interactive segmentation. Interactive segmentation has been widely studied in robotics literature [3], [4], [5], [18], [19], [8], [7], [20], where the dominant approach is to over-segment the image into super-pixels, which are tracked over consecutive frames, clustered together and assigned to the objects they belong to. These approaches employ classical optical flow estimation techniques for matching segmentation-clusters before and after the pushing action, which is challenging for textureless objects due to lack of features. In [18] and [19], authors use grasping action instead of pushing. They remove an object from the scene by grasping and lifting it. The segmentation of that object is then computed by frame difference (before and after object removal) or using a deep neural network. Thus, their efficacy depends largely upon the success of grasping which itself is difficult for dense clutter without accurate segmentation. On the other hand, we aim to learn a policy that leverages pushing actions to improve the segmentation itself.

Applications to other robotic tasks Several robotics tasks rely on precise instance segmentation such as object singulation and searching [21], [9], [22], where the task is to de-clutter the environment. Our approach could be seen as a total singulation approach since we segment every object instance in the scene. In [22] authors assume objects are segmented and present rule based policies for singulating them. Whereas, [21] trains a CNN in supervised manner to predict push location and directions from an RGBD images. Both [21] and [22] greedily select the best push and do not perform long term planning which is important for dense clutters. As a result, their policies suffer from severe performance drops in dense clutters; for example the performance in [22] drops from 88% to 49% when the number of objects in the clutter are increased from 3 to 15. In contrast, we show successful singulation for over 28 objects. A few very recent approaches [9], [23], [24], similar to ours, propose a DQN based policy for object singulation or searching; however they only singulate a single object with a given colour within a clutter of differently coloured objects. On the contrary, we perform singulation of all the objects in the scene within a dense clutter, and do not assume texture information on the objects.

Recent deep network based grasping approaches [25], [26] showcase superior grasping performance over classical approaches for top grasps, however they assume the object is segmented and free from clutter. To circumvent this issue a few approaches [1], [27], [10] perform pushing interactions for improving grasping. In [10], authors combine Dexnet [25] with heuristic based pushing interactions which improves the grasping performance of Dexnet by 15%. In [10], authors attempt to learn a value function, that improves the grasping performance. A recent work [1] proposed two DQNs to learn the synergy between pushing and grasping, i.e., aim to grasp if possible, otherwise push. Authors gave higher award to grasping, so that pushing actions assist the grasping. In contrast to these approaches, our objective is to improve the segmentation performance itself. Moreover, our approach can readily be combined with grasp proposal networks to improve grasping performance as well.

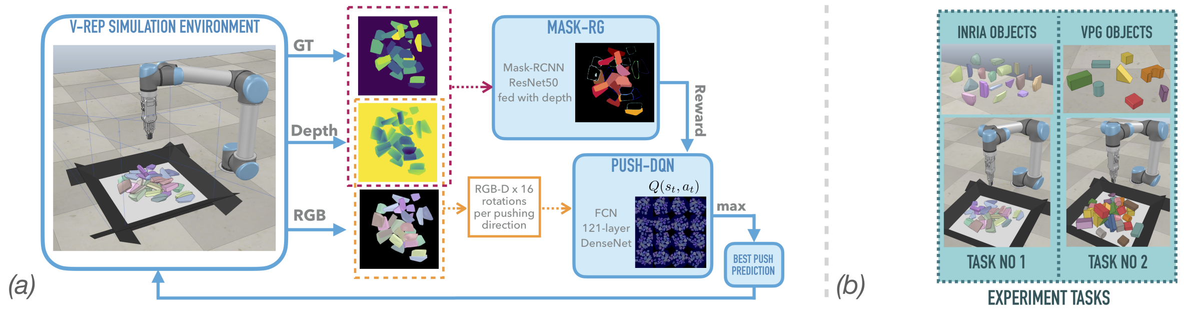


Fig. 2: (a) **Overview of Push-to-See**. Our model is two-fold: a Mask R-CNN based reward generator (Mask-RG) and a DQN based pushing predictor (Push-DQN). RGB, depth and ground truth (GT) segmentation masks are captured from the simulation environment. The depth and GT images are fed into Mask-RG in order to generate a segmentation score based on the difference in category-agnostic instance segmentation quality of the previous scene and the current scene. Meanwhile, 16 RGB-D heightmaps are generated from the depth and RGB to represent the current state-action value for 224×224×16 possible pushing actions similarly as in [1]. Push-DQN is then trained through trial and error by the reward signal to learn the best action-value pair (i.e. pushing action) which would enhance the instance segmentation of the consecutive scene. (b) **Experimental setup** consisting of 2 sets of objects from INRIA and VPG. Dense clutters include 28 objects in Task no 1 and 30 objects in Task no 2. Note the small depth variations in INRIA objects which makes their segmentation challenging. We employ only depth features for segmentation.

III. APPROACH

This paper presents Push-to-See, a methodology that enables robots to autonomously learn how to push a cluttered pile of objects to improve instance segmentation by singulating the clutter through self-supervision. To tackle this problem, we designed a two-fold model, which consists of a Mask R-CNN based Reward Generator (i.e., Mask-RG) to produce segmentation rewards via depth images, and a Deep Q-Network (DQN) based pushing predictor (i.e., Push-DQN), which is trained with this reward signal to find the best actions that decrease the segmentation error in the consecutive scene. The overall autonomous learning pipeline and a summary can be seen in Figure 2(a). The details of Mask-RG and Push-DQN are given in the following two subsections below.

A. *Mask-RG: A Mask R-CNN based Reward Generator*

1) Model: Mask-RCNN [2] has recently been adapted for robot vision by training it via depth images in a Sim2Real study [14]. We used a similar approach in order to predict category-agnostic instance segmentation of the objects in a heap. We triplicated the depth channel and fed those into a Mask-RCNN model with a ResNet-50 FPN backbone. Mask-RG had first been trained offline in simulation separately from Push-DQN. Then, it was used to determine a segmentation quality score for each scene by inference during Push-DQN training.

2) Database Generation: The physics engines and simulation platforms provide an opportunity to collect thousands of samples to train and test the robotics models in a cost-effective way. V-REP simulation environment [28] has been used to generate a database for Mask-RG training in our study. Figure 2(b) shows the 18 3D object models (provided by Larsen Team at INRIA Nancy, France) that form the randomly generated heaps. Our database generation script, first, randomly selected 24 to 32 objects from this set by limiting each object instance count to 4, and then, dropped them one by one onto the workspace by randomising their initial pose in each iteration. Once the objects were settled, the depth data and the ground truth segmentation masks of the scene were collected from the top view. Thus, a database of 12.5k samples of heaps were generated.

3) Training and Test: The collected scenes were randomly reorganised to have a training, validation and a test dataset as, $80\%, 10\%$ and 10% of the database respectively.

10k 1024x1024 pixel depth images were fed into the model. It was then trained for 40 epochs by stochastic gradient decent using ResNet-50 as the backbone, where the learning rate was set to 1.0e-4, batch size to 5, momentum to 0.9, and the weight decay to 5.0e-5.

As the model is category agnostic and the test scenes were composed of the same objects, the coco evaluation on the test set returned high scores:

- Average Precision (AP) @[IoU=0.50:0.95 | area=all | maxDets=100] = 0.822
- Average Recall (AR) @[IoU=0.50:0.95 | area=all | maxDets=100]=0.843

Here, the maximum detection limit (maxDet) was set to 100 and the AP and AR were calculated over multiple Intersection over Unions (IoU) with step size .05.

4) Reward Generation: Mask-RG returns a segmentation score to generate rewards for the training of Push-DQN with the following equation:

$$M_{s_t} = (\sum_{i=1}^{n_t} J_i)/n - 0.02n_f, \tag{1}$$

where M_{s_t} is the Mask-RG segmentation score of the scene at time t, J_i is the Jaccard index of i^{th} true detection, n is the number of objects dropped in the scene, whilst n_t and n_f are the number of true detections and false negatives (non-detected) objects respectively.

B. *Push-DQN: A Deep Q-Network for Non-Prehensile Interaction*

In order to learn best pushing actions that improve the instance segmentation of the consecutive scenes, we used deep reinforcement learning, in particular, the Deep Q-learning framework [11]. Our architecture is inspired by Zeng et al.'s study [1].

1) State and action representations: Each state (s_t) was represented as a 224x224 RGB-D heightmap of the robot's workspace from the top view by merging and transforming the RGB and Depth data captured from a fixed perspective view. Each pixel of the state representation maps 2×10^{-3} m^2 of actual space.

The non-prehensile action (a_t) to select is a straight pushing action of 0.1 m, whose direction vector is collinear to the normal of z-axis at a fixed height. In each state s_t, the agent can perform a pushing action, a_t, in 16 different directions starting from a selected point in s_t . Thus the output of the network represents the Q-values, $Q(s_t, a_t)$, for a discrete 224x224x16 state-action space.

2) Rewards: Our reward function is based on an evaluation of how well the selected pushing action at s_t led to enhance the instance segmentation of the scene at s_{t+1}. First, we generate a segmentation score (M_{s_t}) via Mask-RG. Then, we determine a reward/punishment for the current action by weighting the difference of consecutive segmentation scores as follows:

$$M_\delta = M_{s_{t+1}} - M_{s_t},$$
$$R(s_t, s_{t+1}) = \begin{cases} \text{sgn}(M_\delta)(10M_\delta)^2 & \text{, if } \sum(s_{t+1} - s_t) > c \\ -0.5 & \text{, otherwise ,} \end{cases} \tag{2}$$

where c is the change detection threshold. The change is measured by counting the number of changed pixels between two consecutive heightmaps. As can be seen in Equation (2), when a pushing action does not cause a significant change in the scene, the agent is given a punishment of -0.5.

3) Training: The training of the fully convolutional network of Push-DQN has been done by stochastic gradient decent using the Huber loss function as in [1]. During training, 28 to 34 INRIA objects were randomly dropped in the robot's workspace in V-REP simulation (see the next section for simulation details). The agent performed a pushing action in each iteration either based on the highest Q-value or randomly regarding the exploration rate ϵ. The initial exploration rate of 0.5 has been decayed to 0.1 during training via an ϵ-greedy exploration policy. After each pushing, the instance segmentation quality of the current scene was predicted by Mask-RG and an immediate reward signal $R(s_t, s_{t+1})$ was calculated considering the segmentation of the previous scene according to Equation (2). The expected reward (y_t) was then determined by summing this immediate reward with a discounted ($\gamma = 0.5$) future expectation obtained via a forward pass in Push-DQN as:

$$y_t = R(s_t, s_{t+1}) + \gamma Q(s_{t+1}, \text{argmax}_{a'} Q(s_{t+1}, a')) \tag{3}$$

where a' is the set of all possible pushing actions.

Two models have been trained with different thresholds (Figure 3). During the training of Model 1 (M1), the detections over a confidence score of 0.80 were considered as valid predictions. These were then matched with the ground truth segmentations using 0.80 IoU threshold to determine TPs (true positives). An episode was successful when Mask-RG score M_{s_t} reached over the success threshold of 0.85. If this cannot be achieved after 30 consecutive pushes, that episode was considered as a failure and the scene was reinitiated during the training. For Model 2 (M2), the confidence and mask (IoU) thresholds were set to 0.85, whilst the success threshold to 0.90. The threshold levels of M1 and M2 correspond, respectively, to Condition 1 and 3 in the evaluation experiments (Table I). Both models have been trained around 40k iterations each of which took around a week on a NVIDIA GTX 1080 Ti GPU. Figure 3 shows how the failure rate decreases during training of the models. Whilst initial 75% failure rate decreases to around 12% by M1 in Condition 1, the training of M2 decreases the episode failure rate from around 95% to 57% in Condition 3. This outcome is a result of the increase in the difficulty of the task during the second training due to higher thresholds. M2, on the other hand, performs better in the evaluation experiments that are presented in the next section.

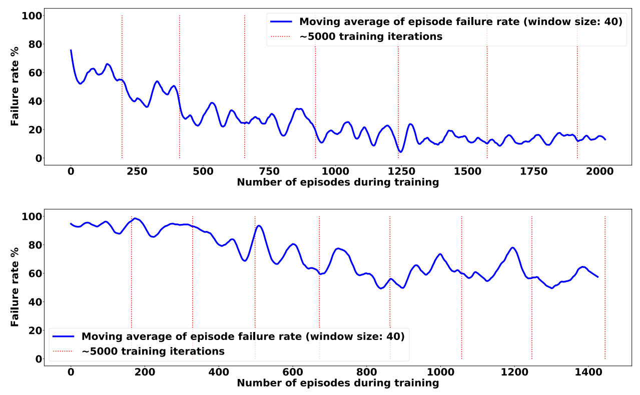


Fig. 3: Moving average of the episode failure rates during two model trainings (window size: 40). (*The training curves have been smoothed out for printing by a Savitzky–Golay filter.*). Upper panel shows the failure rate of Push-to-See (M1) during training in Condition 1, whilst lower panel is the training of Push-to-See (M2) in Condition 3. Red vertical lines indicate each 5000 ± 30 training iterations. Since the episodes require more pushing actions at the beginning of the training, the distance between these lines are smaller.

IV. EXPERIMENTS

To test how well the agent performs, we designed two experiment tasks, as well as a baseline policy to compare the learned policy of the model.

A. Environment

The experiments were conducted on V-REP simulation environment using a UR5 robot. The manipulation trajectories were planned using the inverse kinematics of the simulator and the physical interactions were calculated via Bullet 2.83 with 100 ms simulation time steps.

We generated two cluttered scenes with initially fixed object locations (Figure 2(b)) to benchmark two baselines and two Push-to-See models trained with different thresholds (Figure 3). Task no 1 includes 28 objects from 15 categories from the INRIA 3D models mentioned previously, whilst Task no 2. has 30 objects from 8 VPG categories. We have 3 experiment conditions, where the difficulty of the task was raised from Condition 1 to Condition 3 by increasing the thresholds (see Table I).

B. Experiment Design

The goal of this study is to singulate objects from each other sufficiently, so that the instance segmentation quality of the scene reaches a certain quality level ($M_{s_t} >$ *success threshold*). This level of quality is usually achieved when all 28 objects can be detected by Mask-RG. It should be noted that this is not a strict rule since the segmentation score M_{s_t} can possibly be higher than the success threshold even if there is an undetected object, if all the other objects have very high IoU scores (see Equation (1)).

The agent is considered as successful if it can improve the instance segmentation score M_{s_t} above the success threshold in maximum 30 consecutive non-prehensile actions within a testing episode, and as unsuccessful otherwise.

During the experiments, the exploration rate ϵ was set to zero and no training has been performed. Algorithm 1 shows the computational steps followed to conduct evaluation experiments.

Algorithm 1: Testing task pipeline

```
initialise the scene;
initialise the model with pre-trained weights;
executing pushing action ← False;
(child thread) while executing pushing action do
    predicted action ← $Q_\pi(s_t, a_t)$;
    calculate the pushing trajectory;
    execute action;
    executing pushing action ← False;
end
(main thread) while True do
    camera data ← V-REP simulation;
    if this is the first iteration then
        Mask-RG ← current camera data;
        Mask-RG → initial seg. score $M_{s_{t_0}}$;
    end
    executing pushing action ← True;
    if the very first pushing already executed then
        if $M_{s_t} >$ quality threshold then
            episode success ← True;
        end
        detect any changes in the scene;
        if not episode success then
            calculate rewards;
        end
    end
    wait for executing pushing action;
    if episode success or failure is True then
        reinitialise the scene for the next episode;
    end
    save camera data for the next iteration;
    if not the first iteration then
        camera data ← V-REP simulation;
        Mask-RG ← camera data;
        Mask-RG → segmentation score $M_{s_{t_0}}$;
    end
end
```

C. Baseline Policies

Although there exist rule-based non-prehensile manipulation policies for singulation of objects in sparsely populated scenes [10], [9], to the best of our knowledge, there is not any pushing-only policy in literature which is suitable to use as a baseline for our task that involves densely cluttered environments. Thus, we implemented a heuristic pseudo-random policy to use as one of our baselines. The policy implements the following:

- detect the highest point of the clutter
- select one of the 16 different pushing directions randomly
- move the gripper to the radius of 10 points away from the highest point in the selected direction
- generate a push towards the top of the heap

This pseudo-random baseline can enhance the instance segmentation quality to a certain degree, however, when the complexity of the task increases (e.g. adding more objects or setting high thresholds) it fails to decrease the segmentation error.

Furthermore, the Visual Pushing and Grasping (VPG) model, which is the state-of-the-art model on integrating non-prehensile manipulation with grasping [1], was included as the second baseline. We trained VPG in its authors' setup for 3000 iterations (2400 iterations in [1]) and evaluated in our two benchmark tasks. It should be emphasised that VPG has two action primitives (pushing and grasping) and we did not restrict the VPG's use of grasping actions for not giving it a disadvantage. We assumed that being able to grasp and remove the objects from the scene would benefit in decreasing the segmentation error since the task gets difficult with more objects in the clutter.

D. Results

The two trained models and two baseline policies were tested using Algorithm 1 to evaluate our approach. In each experiment, models were run for a minimum of 40 episodes using 3 different threshold conditions in Task no 1 and one condition in Task no 2. Each episode had maximum of 30 pushing iterations to solve the task, otherwise they were considered as failures. Table I shows the average episode success rate and average number of pushes during the experiments for each model in each condition. As we expected, increasing the mask (IoU) and success thresholds decreased the success rate whilst increasing the number of pushes required to singulate objects. In all cases, Push-to-See performed better than the baseline models.

It is important to note that both Mask-RG and Push-DQN have solely been trained on INRIA objects and the same model weights were used in the second task with the VPG objects without any fine-tuning. As can be seen in the table, the models performed even better with these unseen objects in Task no 2. This is probably the outcome of that the VPG objects have much simpler geometry than the INRIA objects.

Figure 5 shows an example of how the mean average recall (mAR) at .85 IoU threshold increases after each pushing

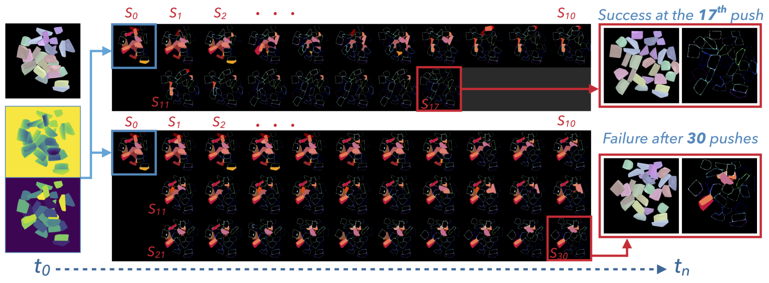
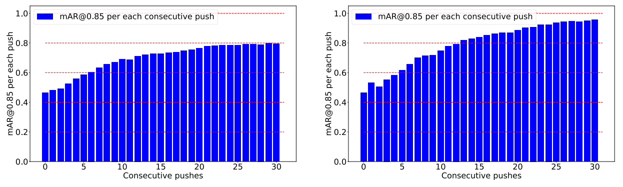


Fig. 4: **A qualitative example of mask prediction during two testing episodes in Task no 1 in Condition 2 (Push-To-See M2 in upper panel, Heuristic baseline in lower panel).** The figure shows how the consecutive pushes decreases the segmentation error of the scene after each interaction. At the initial state s_0, both Mask-RG models receive the same depth and GT images. Red coloured objects are the false negatives (FN). Whilst Push-To-See successfully completes the episode by detecting all of the initial 15 FNs at s_0 after 17 conducting pushes, the heuristic baseline fails to succeed in this episode. After 30 pushing trials, the baseline decreases the number of FNs from 15 to 5 at s_{30}.

TABLE I: Experiment results

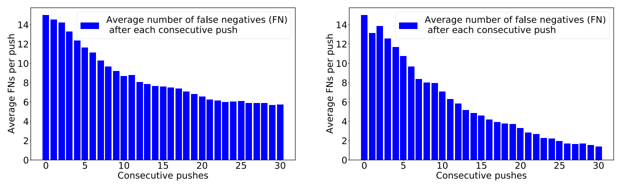
Method	**Avg. #Actions per Episode**	**Success(%)**
Task no 1. (INRIA Objects)		
Condition 1: Mask threshold= **0.80** - Success threshold= **0.85**		
Heuristic Baseline	27.83	30.0%
VPG [1]	26.55	31.0%
Push-to-See M1	15.48	91.3%
Push-to-See M2	**13.37**	**95.0%**
Condition 2: Mask threshold= **0.85** - Success threshold= **0.85**		
Heuristic Baseline	29.48	6.5%
VPG [1]	29.85	4.9%
Push-to-See M1	25.57	52.4%
Push-to-See M2	**23.5**	**75.0%**
Condition 3: Mask threshold= **0.85** - Success threshold= **0.90**		
Heuristic Baseline	30.0	0.0%
VPG [1]	30.0	0.0%
Push-to-See M1	28.88	16.7%
Push-to-See M2	**27.77**	**37.2%**
Task no 2. (VPG Objects)		
Condition 2: Mask threshold= **0.85** - Success threshold= **0.85**		
Heuristic Baseline	15.83	85.7%
VPG [1]	15.6	85.0%
Push-to-See M1	11.02	96.2%
Push-to-See M2	**9.33**	**96.8%**



a Heuristic baseline b Push-to-See M2

Fig. 5: Mean Average Recall @.85 IoU in Task No 1 in Condition 2. Y axes indicate mAR@.85 scores for each particular push conducted one after another. When a task is solved after a push, the final mAR score were carried for the remaining iterations in that episode for calculating the average.

interaction via the heuristic baseline and Push-to-See (M2) during testing in Task no 1 in Condition 2. Although the baseline model has a low success rate in this case, it still improves the segmentation quality significantly. On the other hand, the trained model enhances the segmentation faster and higher than the baseline and completes 75% of the episodes successfully. The comparison on how well the models can find undetected objects (i.e., false negatives) in the same test condition can be seen in Figure 6 (see also Figure 4 for a the qualitative example). 15 of 28 objects of the initial scene can not be detected in this task condition. Whilst the baseline decrease the number of unpredicted objects in the heap to 5.72 after 30 interactions, Push-to-See predicts almost all the objects during the test (The average FN after the 30th push was 1.37.)



a Heuristic baseline b Push-to-See M2

Fig. 6: Average number of false negatives (FN) per consecutive push in Task no.1 and Condition 2. When a task is solved after a push, the final FN score were carried for the remaining iterations in that episode for calculating the average.

V. CONCLUSIONS

In this paper, we presented a novel interactive segmentation approach, to improve instance segmentation quality of densely cluttered scenes by learning how to manipulate objects in a self-supervised manner. The inherent consequence of enhancing the segmentation quality of the scene was the autonomous singulation of all the objects from each other. Our approach achieves 75% success rate for a strict 85% IoU as compared to $\sim$ 7% success achieved by the baselines, which is a significant improvement over the existing state-of-the-art. Moreover, our approach can tackle dense heaps of over 28 objects that lack identifiable features which is crucial for practical applications. Our rigorous experimental protocol also provides a new evaluation methodology, which can be used as a benchmark for interactive manipulation research in future studies.

ACKNOWLEDGEMENT

This research has been supported by HEAP (Human-Guided Learning and Benchmarking of Robotic Heap Sorting, CHIST-ERA | EPSRC, grant no: EP/S033718/2) and NCNR (National Centre for Nuclear Robotics, EPSRC, grant no: EP/R02572X/1) projects. Authors thank to Dr Mihai Andries and the Larsen Team at INRIA, Nancy, for providing the 3D object models.

REFERENCES

[1] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4238–4245, IEEE, 2018.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[3] C. J. Tsikos and R. K. Bajcsy, "Segmentation via manipulation," *Technical Reports (CIS)*, p. 694, 1988.

[4] G. Metta and P. Fitzpatrick, "Better vision through manipulation," *Adaptive Behavior*, vol. 11, no. 2, pp. 109–128, 2003.

[5] P. Fitzpatrick, "First contact: an active vision approach to segmentation," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3, pp. 2161–2166, IEEE, 2003.

[6] T. Patten, M. Zillich, and M. Vincze, "Action selection for interactive object segmentation in clutter," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6297–6304, IEEE, 2018.

[7] J. Kenney, T. Buckley, and O. Brock, "Interactive segmentation for manipulation in unstructured environments," in *2009 IEEE International Conference on Robotics and Automation*, pp. 1377–1382, IEEE, 2009.

[8] H. Van Hoof, O. Kroemer, and J. Peters, "Probabilistic segmentation and targeted exploration of objects in cluttered environments," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1198–1209, 2014.

[9] I. Sarantopoulos, M. Kiatos, Z. Doulgeri, and S. Malassiotis, "Total singulation with modular reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4117–4124, 2021.

[10] M. Danielczuk, J. Mahler, C. Correa, and K. Goldberg, "Linear push policies to increase grasp access for robot bin picking," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1249–1256, IEEE, 2018.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] P. Tang, X. Wang, A. Wang, Y. Yan, W. Liu, J. Huang, and A. Yuille, "Weakly supervised region proposal network and object detection," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 352–368, 2018.

[13] R. Grimm, M. Grotz, S. Ottenhaus, and T. Asfour, "Vision-based robotic pushing and grasping for stone sample collection under computing resource constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 0–0, 2021.

[14] 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 *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7283–7290, IEEE, 2019.

[15] Z. Dong, S. Liu, T. Zhou, H. Cheng, L. Zeng, X. Yu, and H. Liu, "Ppr-net: point-wise pose regression network for instance segmentation and 6d pose estimation in bin-picking scenarios," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1773–1780, IEEE, 2019.

[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*, pp. 2701–2710, 2017.

[17] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," *arXiv preprint arXiv:2007.08073*, 2020.

[18] D. Pathak, Y. Shentu, D. Chen, P. Agrawal, T. Darrell, S. Levine, and J. Malik, "Learning instance segmentation by interaction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2042–2045, 2018.

[19] W. Boerdijk, M. Sundermeyer, M. Durner, and R. Triebel, "Self-supervised object-in-gripper segmentation from robotic motions," *arXiv preprint arXiv:2002.04487*, 2020.

[20] J. Pajarinen and V. Kyrki, "Decision making under uncertain segmentations," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1303–1309, 2015.

[21] A. Eitel, N. Hauff, and W. Burgard, "Learning to singulate objects using a push proposal network," in *Robotics research*, pp. 405–419, Springer, 2020.

[22] Z. Dong, S. Krishnan, S. Dolasia, A. Balakrishna, M. Danielczuk, and K. Goldberg, "Automating planar object singulation by linear pushing with single-point and multi-point contacts," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, pp. 1429–1436, IEEE, 2019.

[23] M. Danielczuk, A. Angelova, V. Vanhoucke, and K. Goldberg, "X-ray: Mechanical search for an occluded object by minimizing support of learned occupancy distributions," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9577–9584, IEEE, 2020.

[24] Y. Yang, H. Liang, and C. Choi, "A deep learning approach to grasping the invisible," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2232–2239, 2020.

[25] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[26] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," in *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[27] A. Boularias, J. A. Bagnell, and A. Stentz, "Learning to manipulate unknown objects in clutter by reinforcement," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[28] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.