Improved Reinforcement Learning Pushing Policies via Heuristic Rules

Marios Kiatos\textsuperscript{1}, Iason Sarantopoulos\textsuperscript{1}, Sotiris Malassiotis\textsuperscript{2} and Zoe Doulgeri\textsuperscript{1}

\textbf{Abstract—} Non-prehensile pushing actions have the potential to singulate a target object from its surrounding clutter in order to facilitate the robotic grasping of the target. To address this problem we utilize a heuristic rule that moves the target object towards the workspace’s empty space and demonstrate that this simple heuristic rule achieves singulation. We incorporate this effective heuristic rule to the reward in order to train more efficiently reinforcement learning (RL) agents for singulation. Simulation experiments demonstrate that this insight increases performance. Finally, our results show that the RL-based policy implicitly learns something similar to one of the used heuristics in terms of decision making. Qualitative results, code, pretrained models and simulation environments are available at https://github.com/robot-clutter/improved_rl.

\section{Introduction}

Despite the research effort during the last decades, extracting target objects from unstructured cluttered environments via robotic grasping is yet to be achieved. Until to this day, robotic solutions are mostly limited to highly structured and predictable setups within industrial environments. In these environments the robots perform successfully because the tasks require characteristics such as precision, repeatability and high payload. However, tasks such as tight packing \cite{1}, throwing objects on target packages \cite{2} or object grasping and manipulation in cluttered environments \cite{3} require a much more different skill set and are still performed mostly by humans. Even though humans lack precision and strength, they are equipped with dexterity and manipulation skills that are missing from today’s robots, endowing humans with the ability to deal effectively with extremely diverse environments. Such a fundamental manipulation skill is robust grasping of target objects \cite{4}–\cite{6}. However, when dealing with clutter, grasping affordances are not always available unless the robot intentionally rearrange the objects in the scene. As a result, non-prehensile manipulation primitives are used for object rearrangement \cite{7} and a body of research emerged for total or partial singulation of objects using pushing actions, in order to create space around the target object required by the robotic fingers for grasping it. Total singulation \cite{8} refers to the creation of free space around the object, while partial singulation refers to the creation of free space along some object sides.

Both hand-crafted and data-driven singulation methods have been developed for object singulation. Hand-crafted methods include analytic or heuristic methods, where analytic use exact models for making predictions assuming accurate knowledge of the environment and heuristic methods follow simple rules that can lead to task completion without assuming accurate knowledge of the environment. Analytic or heuristic-based methods use insights for the task, lacking however the generalization ability of the data-driven methods. The success of deep learning \cite{9} in fields like computer vision or language processing motivated the robotics community to solve robotic problems using deep learning \cite{10}, \cite{11} or reinforcement learning \cite{12}–\cite{15}. However, deep learning approaches come with the cost of large data requirements, a problem that is exacerbated in robotics due to huge state/actions spaces as well as time consuming, costly and risky training when dealing with real robots. This led to training agents in simulated environments and transfer them in the real robotic system either by robust feature selection \cite{8} or by sim-to-real methods \cite{16}.

In this paper, our objective is to study the learning-based policy proposed in our previous work \cite{8} with respect to pushing policies that follow simple but effective heuristic rules, answering two questions: Can the heuristics that achieve total singulation be used during training to increase the performance of the RL-based policy? Does the RL-based policy implicitly learn something similar to the heuristics? In this work, we aim to answer the above questions by:

- demonstrating that heuristic policies that move the target object towards the empty-from-obstacles space can be effective in a singulation task,
- demonstrating that guiding the training with these heuristic rules in the reward, increase the performance of the RL-based policy and
- performing experiments to study the similarity of the initial RL-based policy with the heuristic-based policies, in terms of decision making.

In the following section the related work is presented. Section III describes the environment and the problem formulation. Section IV presents the heuristic and the learning based policies used in this work as well as training details for the learning-based policies. Section V provides the results from our experiments and finally Section VI draws the conclusions.

\section{Related Work}

\subsection{Analytic and heuristic-based approaches}

Lynch \textit{et al.} \cite{17} pioneered research on analytic models of push mechanics. Following works focused on planning...
methods to reduce grasp uncertainty in cluttered scenes by utilizing push-grasping actions [18], [19]. Cosgun et al. [20] proposed a method for selecting pushing actions in order to move a target object in a planned position as well as clearing the obstructed space from other objects, assuming however knowledge of exact object poses. In contrast to these analytic methods where exact knowledge of the objects and the environment is assumed, other works employ heuristics for manipulating table-top objects. Hermans et al. [21] proposed a method that separates table-top objects, exploiting the visible boundaries that the edges of the objects create on the RGB image. Similarly, Katz et al. [22] segmented the scene for poking the objects in the scene before grasping them. Chang et al. [23] proposed a heuristic singulation method which pushes clusters of objects away from other clusters. Their method checked if a push separated the target cluster into multiple units and iteratively kept pushing until the target cluster remained a single unit after the push. The downside is that the clusters should be tracked in order to produce future decisions. Finally, Danielczuk et al. [24] proposed two heuristic push policies for moving objects towards free space or for diffusing clusters of objects, using the euclidean clustering method for segmenting the scene into potential objects. We adapt the free space heuristic to our problem and demonstrate that achieves total singulation.

B. Learning-based approaches

Recently, a lot of researchers employed learning algorithms for object manipulation. Boularias et al. [25] explored the use of reinforcement learning for training policies to choose among grasp and push primitives. They learnt to singulate two specific objects given a depth image of the scene but the agent required retraining for a new set of objects. Zeng et al. [26] used Q-learning to train end-to-end two fully convolutional networks for learning synergies between pushing and grasping, leading to higher grasp success rates. Yang et al. [27] adapted [26] to grasp a target object from clutter by partially singulating it via pushing actions. Kurenkov et al. [28] and Novkovic et al. [29] used deep reinforcement learning to learn push actions in order to unveil a fully occluded target object. In contrast to singulation, achieving full visibility of the target does not always create free space between the target and surrounding obstacles, which is necessary for prehensile grasping. On the contrary to model-free reinforcement learning approaches, Huang et al. [30] proposed a model-based solution for target object retrieval. Specifically, they employ visual foresight trees to intelligently rearrange the surrounding clutter by pushing actions so that the target object can be grasped easily. All the above approaches focus on the partial singulation of objects in order to grasp them.

On the other hand, many works develop methods that totally singulate the objects. Eitel et al. [31] trained a convolution neural network on segmented images in order to singulate every object in scene. However, they evaluated a large number of potential pushes to find the optimal action. Kiatos et al. [32] trained a deep Q-network to select push actions in order to singulate a target object from its surrounding clutter with the minimum number of pushes using depth features to approximate the topography of the scene. Sarantopoulos and Kiatos [33] extended the work of [32] by modelling the Q-function with two different networks, one for each primitive action, leading to higher success rates and faster network convergence. Finally, Sarantopoulos et al. [8] proposed a modular reinforcement learning approach which uses continuous actions to totally singulate the target object. They used a high level policy, which selects between pushing primitives that are learnt separately and incorporated prior knowledge, through action primitives and feature selection, to increase sample efficiency. Their experiments demonstrated increased success rates over previous works. We build upon this work to demonstrate that incorporating insights from heuristics to reinforcement learning algorithms via reward shaping, can increase the performance further and study the similarity of the decisions that this learning-based policy makes with respect to the heuristic-based policies.

III. PROBLEM FORMULATION

The objective is to properly rearrange the clutter in order to totally singulate the target object. Total singulation means that the distance between the target object and its closest obstacle is larger than a predefined constant distance $d_{\text{sing}}$. Given an RGB-D image of the scene, the agent selects the pushing action that maximizes the probability of singulating the target object.

A. Environment

The environment consists of a robotic finger, a single overhead RGB-D camera and rigid objects in clutter resting on a rectangular workspace with predefined dimensions $s \times s$ and adequate surrounding free space. The pose, dimensions and the number of obstacles are random. Finally, the world frame $\{T_w\}$ is placed to the center of the workspace with its z-axis being opposite to the gravity direction as shown in Fig. 1.
A. Heuristic-based pushing policies

The heuristic policy used in this work is a policy which selects pushing actions for moving the target object towards the empty space of the scene, which is inspired by [24]. The assumption here is that moving the target object towards the empty space will create empty space around it, and thus accomplishing total singulation.

In order to select the best action, we need to calculate the empty space map (ESM) shown in Fig. 3(e). To produce this map, we first create a mask of the obstacles of the scene (Fig. 3(a)) by removing the target object from the visual state representation $S$, since we only interested in calculating the distances from the obstacles. Then, we calculate the points $c \in C$ that belong to the contour $C$ of this mask (Fig. 3(b)). Subsequently, we create the obstacle distance transform (ODT) showing the distances of each workspace point from the obstacles (Fig. 3(c)). Hence, the value of each pixel $(i, j)$ of ODT is its minimum distance from every point of the contour $C$, i.e.:

$$\text{ODT}(i, j) = \min_{c \in C} ||(i, j) - c||$$

To avoid pushing the target object out of the workspace limits, we create a limits distance map (LDM) by calculating for each pixel its minimum distance from the limits of the workspace, as shown in Fig. 3(d). Then, the final map is given by keeping the minimum value between the obstacles distance map and the limit distance map:

$$\text{ESM}(i, j) = \min\{\text{ODT}(i, j), \text{LDM}(i, j)\}$$

as shown in Fig. 3(e). As a final step we normalize the values of the map to $[0, 1]$, given its minimum and maximum value.

Given the empty space map, we can select the direction $\theta$ and the pushing distance $d$ for pushing the target object towards the empty space. In order to determine the direction, we find all the pixels $p$ of ESM map whose value is above 0.9, i.e. the top 10%. From this set, we select as the optimal pixel $p^*$ the pixel that has the minimum distance from the centroid $o$ of the target object, calculated using the target’s mask $M$. Note that taking multiple best values, instead of just the maximum, can produce better decisions. For example, if one pixel far from the target object has the value 0.97 and another pixel much closer to the target has the value 0.95, it would not be efficient to just greedily select the maximum value, because the agent could oscillate between pushes.
that unnecessarily attempt to reach remote areas. Then, the direction $\theta$ is computed by the line segment between $o$ and $p^*$ and the pushing distance $d$ is the length of this line segment. Finally, we enforce an upper limit to $d$ for avoiding long pushes that can result to unpredictable behaviour of the target object. Fig. 4(a) shows the selected push of this policy, called Empty Space Policy (ES) for this scene.

Even though ES policy selects the optimal pixel between multiple candidates, it can still produce pushes based on regions that are far from the target. For this reason, we also propose a variant called Local Empty Space policy (LES) that selects the best action taking into account only a local neighborhood of the target object. To this end, we crop the ESM map around the target object, producing a local empty space map (LESM) shown in Fig. 4(b) along with the selected action by LES policy. In particular, the crop size corresponds to $1/4$ of the workspace area. Notice that for the same scene, the two policies, ES and LES, produce very different actions.

### B. Learning-based pushing policies

To train the reinforcement learning policies we formulate the problem as a Markov Decision Process (MDP) with time horizon $T$. An MDP is a tuple $(S, A, R, T, \gamma)$ of continuous states $S$, continuous actions $A$, rewards $R$, an unknown transition function $T$ and a discount factor $\gamma$.

**States:** We use both the visual state representation presented in Section III-B and the full state representation taken from simulation. The full state representation contains the exact poses of the objects as well as their bounding boxes. To reduce the dimensionality of the visual state representation, we train an autoencoder on a large dataset of observations $V$ and use the latent vector $V_{enc}$ as the input to the agents (Fig. 2). The architecture of the autoencoder is similar to the one proposed in [8], but trained with cross-entropy loss instead of regression. Note that in this work we encode the whole view of the scene instead of a downscaled crop centered around the target.

**Actions:** All the agents use the pushing actions defined in Section III-C.

**Rewards:** First we define a sparse reward that will motivate the RL agent to discover a policy purely for the singulation task. Then, to investigate how insights from the heuristics, described in Section IV-A, improve the performance of reinforcement learning policies, we present two additional reward schemes. To achieve that, we use the ESM and LESM maps for reward shaping. Specifically, we measure the error $e$ between the predicted angle $\theta_p$ that the agent produced and the angle $\theta_h$ that the heuristic policy would have taken if used:

$$e(\theta_p, \theta_h) = \frac{1}{2} (1 - \cos(|\theta_h - \theta_p|)) \in [0, 1],$$

This error will be zero if the directions are the same ($|\theta_h - \theta_p| = 0$) and it will be one if the directions are exactly opposite ($|\theta_h - \theta_p| = \pi$). To this end, we propose three different versions of reward:

1) For training the vanilla RL agent, we assign $r = -1$ if the target falls out of the workspace and $r = +1$ if the target is totally singulated. In any other case, we assign a small negative reward $r = -1/T_{max}$ to each push for motivating the agent to minimize the number of actions, where $T_{max}$ is the maximum number of timesteps.

2) To guide the RL policy towards the global empty space, we use the ESM map to compute the error $e$. Specifically, we assign $r = (-1-e)/(2T_{max})$ for each push in order to penalize the pushes that move the target away from the optimal $p^*$ computed from the ESM map and simultaneously minimize the number of actions. Moreover, we assign $r = -1$ if the target falls out of the workspace and $r = +1$ if the target is totally singulated.

3) Finally, to guide the RL policy towards the local empty space, we use the same reward as above but utilize the
LSM map to compute the error $e$.

**Terminal states:** The episode is terminated if the target is totally singulated, falls out of the workspace or the maximum number of timesteps has been reached.

We exploit the fact that we train in simulation by employing an asymmetric actor-critic algorithm, proposed in [34], in which the critic is trained with the full state representation while the actor gets the visual state representation as inputs. In particular, we use the Deep Deterministic Policy Gradient (DDPG) [35] as the actor-critic algorithm. In contrast to [8] we train the agents in an off-policy way by filling a replay buffer with 100K transitions of random pushing actions. During training we do not update the replay buffer with new transitions and thus we can pass the entire dataset multiple times (epochs) to update the networks.

The actor and the critic are both 3-layered fully-connected with 512 units each layer. The target networks are updated with polyak averaging of 0.999. We train the networks for 50 epochs with batch size 32 and discount factor $\gamma = 0.9$. Both networks are trained using Adam optimizer with learning rate $10^{-3}$.

V. Results

The objective of the experiments are two-fold: (a) to evaluate if the use of heuristics during training can improve the performance of the RL-based policy in the singulation task and (b) to examine if there is any similarity in the decision making between the policy produced by reinforcement learning and the heuristics. The policies that we evaluate are the following:

- **ES:** a heuristic policy that chooses pushing actions according to the empty space map of Section IV-A.
- **LES:** a heuristic policy which selects the optimal action based on the local empty space map of Section IV-A.
- **RL:** policy trained with the reward (1) of Section IV-B.
- **RL-ES:** policy trained with the reward (2) of Section IV-B, that guides the agent using the empty space map.
- **RL-LES:** policy trained using the reward (3) of Section IV-B that guides the agent using the local empty space map.

We use the Bullet physics engine [36] to advance simulation. The objects are approximated as rectangulars of random dimension. The number of obstacles is between 8 and 13. Furthermore, we use a rectangular robotic finger, $d_{sing} = 3$ cm, a workspace $0.5 \times 0.5$ m and keep all the dynamic parameters of the simulation fixed. All polices are evaluated in the same 200 scenes in simulation.

### A. Performance of the policies

The proposed heuristics where developed to solve the singulation task in an environment with adequate empty space. As we expected ES and LES achieve high success rates and in particular 97.9% and 98.8% respectively. These results show the effectiveness of these heuristics and consequently they can be used as insights to the RL-based policies for improving their performance.

| Policy | Success Rate % | Mean #actions | Std #actions |
|--------|----------------|---------------|--------------|
| RL     | 89.1           | 2.57          | 1.31         |
| RL-ES  | 94.8           | 4.08          | 2.73         |
| RL-LES | 92.3           | 3.67          | 2.39         |

$ES = $ Empty Space, $LES = $ Local Empty Space

### B. Similarity of decision making between learning and heuristics

As we see in Fig. 5(a), RL-ES and RL-LES policies outperform RL policy. The improved success rates demonstrate the importance of incorporating insights from heuristics to guide reinforcement learning agents through reward shaping. This leads to faster convergence and improved success rates. Finally, the RL-ES policy outperforms the RL-LES due to the fact that it is optimized with ES map which takes into consideration the whole scene and thus producing actions that lead to increased success rate.

**Fig. 5. Qualitative results for the performance of the different policies. (a) Success rates; (b) Mean number of actions**

For evaluating the similarity of the decision making between heuristic and learning based policies, we compare the pushing directions decided by the policies RL, RL-ES, RL-LES with the pushing directions that would have been decided by the heuristic policies ES, LES if used in the same scenes. We only consider the differences in the direction and not in the pushing distance as the latter is dependant in the direction. To this end, we run 200 episodes for each policy RL, RL-ES, RL-LES. For each scene we compute the angle of the pushing direction $\theta_p$ produced by each policy and executed in the environment and the angle of the pushing direction $\theta_h$ that the heuristic ES or LES would have produced, if used. Then, we calculate the error between these angles $e(\theta_p, \theta_h)$ from Eq. (3).

Fig. 7(a) shows the distribution of the errors for the 200 episodes between RL and ES policies (blue bars) along with the distribution of errors between RL-ES and ES policies (orange bars). Notice that the distribution of RL-ES is shifted...
Fig. 6. A sample of actions taken by the policies in the same scene. (a) The initial state of the scene. (b) The actions taken by each policy. The start and the end of the arrow is placed on the initial and the final position of the push. (c) The next states for each action.

Fig. 7. The distribution of the errors between the directions chosen by the policies. (a) The errors between the RL, RL-ES policies with the heuristic ES. With blue color the $e(\theta_{RL}, \theta_{ES})$ and with orange color the error $e(\theta_{RL-ES}, \theta_{ES})$; (b) The errors between the RL, RL-LES policies with the heuristic LES. With blue color the $e(\theta_{RL}, \theta_{LES})$ and with orange color the error $e(\theta_{RL-LES}, \theta_{LES})$.

towards smaller errors compared to the RL distribution (see the difference in the range of 0-0.2 of the error), indicating that RL and RL-ES decide on different actions for the same scene and thus RL learns something different. On the other hand, Fig. 7(b) shows the distribution of the errors between RL and LES policies (blue bars) along with the distribution of errors between RL-LES and LES policies (orange bars). In this figure we observe that the distribution of errors between RL and RL-LES are matching, indicating similar decision making. From the above, we can conclude that the RL policy implicitly learns something more similar to the local heuristic than to the global heuristic. Fig. 6 demonstrates a sample of actions taken by the different policies. Notice that in this sample the RL policy decide on a similar action with the local policies LES and RL-LES than the global ES and RL-ES.

This conclusion can also explain the results of Fig. 5(a) related to the performance of the policies. If the RL agent learns implicitly something similar to the LES policy, then we expect that guiding with directions from LES should have a smaller impact in the performance than guiding with ES. As we see in Fig. 5(a), guiding the RL agent to learn a policy closer to ES, the success rate was increased from 89.1% (RL) to 94.8% (RL-ES), in contrast to guiding the agent to learn a policy closer to LES, where the corresponding improvement of the success rate was smaller (92.3% for RL-LES).

Finally, the conclusion that RL learns something closer to the local heuristic LES is to be expected from another point of view. The discount factor $\gamma$ limits the time horizon that the agent can optimize the Q-values into the future and given the limited pushing distance in this task, the agent learns to act better by taking into account observations from the local neighborhood of the target object, rather than observations corresponding to a distance far from the target.

VI. CONCLUSIONS

In this paper, we employed simple heuristic rules to achieve total singulation by pushing the target object towards the free space of the workspace. Experiments demonstrated that if we incorporate these heuristics to the reward of RL agents, singulation success is increased. Finally, we showed that the pure RL-based policy implicitly learns something more similar to the local heuristic.

REFERENCES

[1] F. Wang and K. Hauser, “Dense robotic packing of irregular and novel 3-d objects,” *IEEE Transactions on Robotics*, 2021.
[2] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, “Tossing-bot: Learning to throw arbitrary objects with residual physics,” *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.
