

Robotic depalletizing via reinforcement learning of a pushing policy

Argyris Dimou^{1,2}[0000–0003–2207–277X], Marios Kiatos³[0000–0002–2522–2529],
and Sotiris Malassiotis¹[0000–0002–3911–7527]

¹ Information Technologies Institute (ITI) Center of Research and Technology Hellas
(CERTH) 57001 Thessaloniki, Greece

`{ardimou,malasiot}@iti.gr`

² Department of Physics, University of Thessaloniki, 54124 Thessaloniki, Greece

³ Progressive Robotics, 54645 Thessaloniki, Greece

`mk@progressiverobotics.ai`

Abstract. Depalletizing, the process of unloading objects from pallets, is a critical task in warehouse automation. Robotic systems face challenges due to the diverse shapes and sizes of objects, often leading to failed grasps in cluttered environments. Pushing actions offer a promising solution to facilitate object singulation. In this work, we employ reinforcement learning (RL) for obtaining optimal pushing policies in depalletizing. We formulate the singulation task as goal reaching, where the target object is moved towards a goal area free from obstacles. Our approach, Goal-FCN (Goal-Driven Fully Convolutional Network), combines feature extraction with a fully convolutional neural network to predict effective pushing actions based on RGB-D images and scene segmentation. Experiments demonstrate the efficacy of our RL-based approach in achieving high success rates in challenging depalletization scenarios.

Keywords: Robotics · Machine learning · Reinforcement learning

1 Introduction

In recent years, depalletizing has emerged as a critical challenge in robotic manipulation, spanning multiple research domains. Past research endeavors have primarily focused on detecting objects within the scene, segmenting them [9], and identifying and planning the most efficient actions to aid in the unloading process [11]. Suction grasping is a widely method used by various robotic systems that enhance speed and accuracy of the process. Researchers have introduced innovative depalletizing solutions, including compact robots by Nakamoto et al. [16] leveraging camera and depth images for speed, and Tanaka et al.’s [22] suction-type end effector for simultaneous extraction and improved load resistance. Krug et al. [14] proposed a novel grasp representation scheme, while Fontanelli et al. [7] utilized a flexible gripper for versatile handling of varied box shapes.

While suction grasping is a common approach in controlled settings, it is often unsuitable for depalletizing tasks in cluttered scenarios. Suction grasping

relies on creating a vacuum seal between the gripper and the object’s surface, which works well for flat and smooth objects. However, in cluttered environments such as supermarket pallets, objects vary greatly in shape, size, and surface texture, making it difficult for suction-based systems to reliably grasp them. Additionally, suction grippers struggle with porous or irregular surfaces, common characteristics of many items found in depalletizing tasks. Consequently, suction grasping is not a viable option for these scenarios, further emphasizing the need for alternative manipulation strategies. To overcome these difficulties we utilize prehensile grasping. However, in order to be efficient, empty space is needed. To create space, we employ non-prehensile actions [21] like pushing. These actions, offer avenues for creating essential empty spaces and repositioning objects without directly grasping them. By incorporating non-prehensile actions alongside prehensile ones [4, 5, 13], robotic systems can adeptly negotiate cluttered scenarios, thereby enhancing their overall manipulative capabilities. 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 [19] refers to the creation of free space around the object, while partial singulation refers to the creation of free space along some object sides.

In this paper, we propose a reinforcement learning approach to tackle the difficulties in depalletizing a pallet including diverse objects. In summary, we:

- We employ a non-prehensile approach to enhance the efficiency of depalletizing, generating space within cluttered environments, similar to a mixed supermarket’s pallet.
- Propose a reinforcement learning (RL) framework, specifically the Goal-FCN (Goal-Driven Fully Convolutional Network), for selecting optimal pushing actions.
- Showcase the model’s adaptability and generalization capabilities through successful evaluations in two distinct cluttered environments, demonstrating high success rates.

2 Related work

Recently, a lot of researchers employed learning algorithms for object manipulation. Boularias *et al.* [2] explored the use of reinforcement learning to obtain policies, choosing 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.* [25] used Q -learning to train end-to-end two fully convolutional networks in order to learn synergies between pushing and grasping, leading to higher grasp success rates. Yang *et al.* [23] adapted [25] to grasp a target object from clutter by partially singulating it via pushing actions. Kurenkov *et al.* [15] and Novkovic *et al.* [17] 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.* [10] proposed a model-based solution for target object retrieval. Specifically, they employed 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.* [6] trained a convolutional neural network on segmented images in order to singulate every object in the scene. However, they evaluated a large number of potential pushes to find the optimal action. Kiatos *et al.* [12] 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 [18] extended the work of [12] 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.* [19] 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 different pushing primitives that are learnt separately. Although, their experiments demonstrated increased success rates over previous works, they considered scenes with low clutter and thus this approach is only beneficial in scenes with enough free space. Our work extends the existing literature by addressing the task of singulating all objects within a pallet, sequentially. While previous methodologies have primarily concentrated on isolating specific objects from cluttered scenes, our approach encompasses the singulation of multiple objects, increasing the level of difficulty.

3 Problem formulation

The objective is for an agent to perform the optimal pushing actions to effectively singulate all the objects of a pallet of diverse objects, making them available for easy grasps, simplifying the depalletizing process. Given an RGB-D image, the segmentation of the scene and the goal region where the pallet objects should be driven, the agent selects the pushing action that maximizes the probability of singulating the target object.

3.1 Environment

The environment consists of a robotic finger, rigid objects in clutter resting on a rectangular workspace with predefined dimensions $s \times s$ and a single overhead RGB-D camera. The scene comprises a variety of objects such as boxes, cylinders, and bottles, each characterized by dimensions randomly sampled from a predefined range. This diversity contributes to the complexity of the environment, presenting a challenging depalletizing task. Additionally, the pose, geometry, and quantity of obstacles are randomized, further enhancing the variability

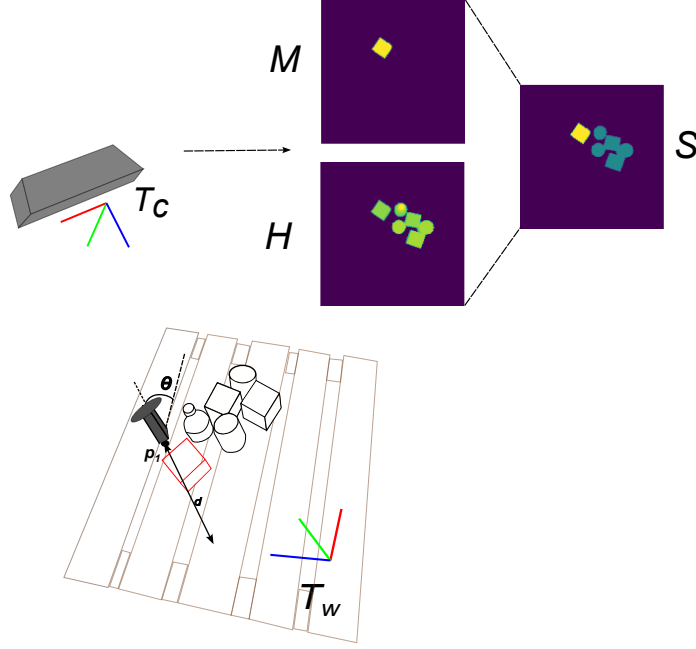


Fig. 1. Illustration of the environment including the support surface, the objects, the robotic finger and the camera. The inertia frame T_w is placed on the center of the table. The visual input $I = \{H, M, S\}$ is a tuple of maps representing orthographic top-down views based on the camera pose T_c . A push action is defined by the initial position p_1 that the push starts, the direction angle θ and the pushing distance d . The goal is to singulate the target object (red contour) from the surrounding obstacles.

and realism of the scenario. The world frame T_w is placed to the center of the workspace with its z-axis being opposite to the gravity direction. An overview of the environment is shown in Fig. 1, where the target object is highlighted in red, and the rest of the objects in black. Note that we choose as initial target the object with the smallest distance from the center of the workspace. Everytime an object is unloaded, we choose as target the object which is closer to the goal region, the same way as a person would operate.

3.2 Visual Representations

We represent the visual input I as a tuple of 2D map images $I = \{M, H, S\}$. To compute these maps, we capture an RGB-D image from a camera as shown in Fig. 1. Given the RGB image, we generate the mask of the target object using a color detector or an advanced object detector. Note that we assume that the target is at least partially visible and can be visually distinguished from

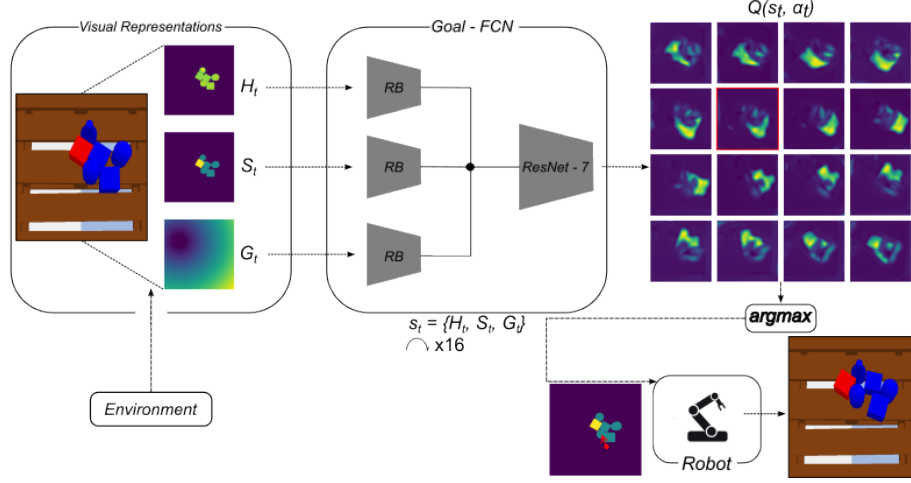


Fig. 2. The heightmap H_t , the segmentation of the scene S_t and the goal G_t are the input of our model Goal-FCN. Initially, we extract features for each image through a residual block $RB(c)$. Then the output features are concatenated and fed to a seven-layer fully convolutional residual network, ResNet-7. The outputs are pixel-wise Q maps with the same size and resolution, which represent the future expected rewards. Robot chooses the pixel with the highest Q -value as the optimal push

the obstacles. The mask and depth images are orthographically projected in the gravity direction using known extrinsic and intrinsic parameters of the camera to construct a mask map M and a depth heightmap H respectively. Finally, the heightmap S is an image generated by the fusion of H and M that provides a segmentation of the scene between the target, the obstacles and the support surface. The pixels that belong to the target have the value 0.5, those belonging to an obstacle the value 1 and the rest of them belonging to the support surface the value 0. The edges of the maps are defined with respect to the workspace limits. In our experiments, this area covers 0.5×0.5 m surface. The maps have image resolution 100×100 and thus each pixel $i \in I$ represents a 5×5 mm vertical column of 3-D space in the robot's workspace.

3.3 Actions

The robot executes push actions by moving the robot finger along a line segment $[\mathbf{p}_1, \mathbf{p}_2] \in \mathbb{R}^2$ parallel to the support surface. The final point is determined by the angle θ with respect to x axis and the pushing distance d , i.e. $\mathbf{p}_2 = \mathbf{p}_1 + [d \cos(\theta), d \sin(\theta)]^T$. Notice that the height h from the table remains the same during the action execution.

4 Pushing policy

Our policy is optimized using deep Q -learning. Thus, we formulate the total singulation as a Markov Decision Process (MDP) with time horizon T : at any given state s_t the robot executes a pushing action a_t according to a policy $\pi(s_t)$, then transitions to a new state s_{t+1} and receives an immediate reward $r(s_t, a_t, s_{t+1})$. The goal is to find an optimal policy π^* that maximizes the total expected reward, given by $r_t = \sum_{i=t}^T \gamma^{i-t} r(s_i, a_i, s_{i+1})$, where γ is a discount factor.

Inspired by [25], we model the Q -function as a fully-convolutional neural network (FCN). A goal observation G is used as input to the FCN (Goal-FCN), which defines the regions that the target and the obstacles should be depalletized (Fig. 2). Thus, the pushing actions move either the obstacles or the target. To generate the goal image $G \in \mathbb{R}^{H \times W}$, we utilize the empty space around the objects. To simulate a variety of depalletizing scenarios, including both challenging and simpler contexts, we opt for a randomized selection process to determine the center \mathbf{p}^* of the goal region within the initial empty space map. This approach introduces variability, allowing for scenarios where the goal area may be either more or less accessible for the entire pallet, mirroring real-world conditions. Taking into account that the segmentation of the scene, which is used to calculate the mask of the target M , could be slightly inaccurate especially in a cluttered scene, we create a goal region slightly larger than the target. Then, we assign the value 0 to the pixels inside a circle with radius $0.65 \times r$ and center \mathbf{p}^* . This circle is defined in a way that entirely encloses the target’s bounding box, i.e. the radius r is computed as slightly larger than the half length of the diagonal that belongs to the target’s bounding box. This 0-pixel region defines the goal area that the target should be moved. To the rest of the pixels $g_i \in G$, we assign the value that corresponds to the distance between the position of the pixel i and the circle defined by the goal area. Essentially, we provide to the network the position of each pixel with respect to the area that the target should be moved, which may allow the network to make decisions based on the objects location. Note that the pixels $g_i \in G$ whose distance from the circle is smaller than d_{sing} , compose the singulation area i.e. the area that should be freed from obstacles. To this end, we represent each state $s_t = (H_t, S_t, G_t)$.

4.1 Goal-FCN

The Goal-FCN takes as input the heightmap H_t , the segmentation of the scene S_t and the goal G_t and outputs pixel-wise Q maps with the same size and resolution as the inputs as shown in Fig. 2. Each image is fed to a 2-layer residual block [8] for feature extraction and subsequently the output features are concatenated and fed to a seven-layer fully convolutional residual network similar to the one of [24]. Note that, the ResNet-7 has the following layers: C(3,64)-MP-RB(128)-MP-RB(256)-RB(512)-RB(256)-RB(128)-UP-RB(64)-UP-C(1,1), where C(k, c) is a convolutional layer with $k \times k$ filters and c channels, RB(c) is a residual block with two convolutional layers using 3×3 filters and c channels, MP is a 3×3 max

pooling layer with stride = 2, and UP is a bilinear $2\times$ upsampling layer. Each value of pixel $q_i \in Q$ represents the future expected reward of executing a pushing action that starts from a position defined by the pixel i . To simplify learning the angle θ , we account for different pushing directions by rotating the input images into $k = 16$ discrete orientations (different multiplies of 22.5°) and then consider only horizontal pushes to the right. Rotations are done with nearest-neighbor sampling to avoid blurring artifacts, as well as 0-padding to maintain fixed image sizes. Thus, during training each image is rotated according to the angle θ before feeding to the Goal-FCN. On the contrary, during inference we rotate each image $k = 16$ times before feeding to the network. The pixel with the highest Q value across all 16 output maps determines the optimal pushing action. Note that the pushing distance is not a learnable parameter and is kept fixed to 0.1 m. Our pixel-wise parameterization of both state and action spaces enables the use of FCNs as Q-function approximators and provides sample efficiency, since each pixel-wise predictions shares convolutional features for all pushing locations and orientations (i.e. translation and rotation equivariance).

4.2 Rewards

We define a sparse reward scheme. If the agent reduces the target’s distance from the optimal pixel p^* or pushes an obstacle out of the singulation area, we assign the value 0.5. If the push results to goal reaching i.e. the target is moved to goal area and there is no obstacle in the singulation area, we assign the value 1. Otherwise, we assign to each push the value 0. To check if the target has reached the goal area, we compute the intersection-over-union (IoU) between the target mask and the 0-pixel area in image G . In this particular problem, we consider more important to demonstrate the ability of the model to singulate all the objects of the pallet in the goal region, rather than the precision of the singulation of each object. Thus, we set the value of the IoU threshold to consider the object singulated, equal to 0.5

4.3 Terminal States

The episode is terminated when all the target objects reach the goal area, if any object of the scene has fallen out of the support surface, or the predefined maximum number of timesteps T_{max} has been reached. The episode is considered successful only on the first case of goal reaching. In any other case the episode is considered as failed.

4.4 Training

Our Goal-FCN is trained to minimize the temporal difference error δ_t of $Q(s_t, a_t)$ to a fixed target value y_t :

$$\delta_t = |Q(s_t, a_t | \theta_t) - y_t| \quad (1)$$

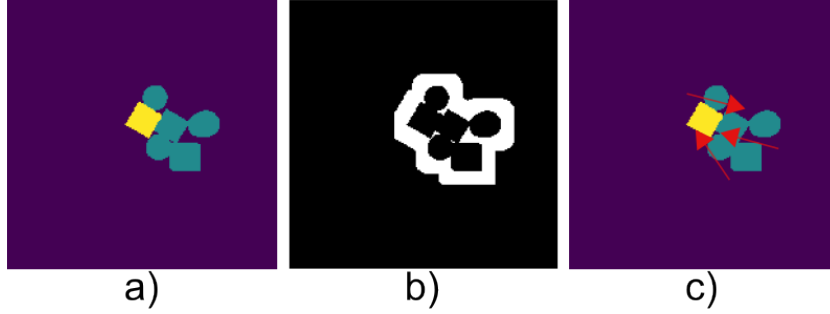


Fig. 3. Illustration of the guided exploration. (a) The state representation S . (b) The set of table pixels around the target from which \mathbf{p}_1 are sampled. (c) Examples of pushes around the target, with the initial pushing position \mathbf{p}_1 and the final pushing position \mathbf{p}_2 depicted by the start and the end of the arrows

$$y_t = r(s_t, a_t, s_{t+1}) + \gamma \max_{a'} Q(s_{t+1}, a' | \theta_t^-) \quad (2)$$

via Huber loss:

$$L = \begin{cases} \frac{1}{2} \delta_t, & \text{if } |\delta_t| \leq 0.1 \\ |\delta_t| - \frac{1}{2}, & \text{otherwise} \end{cases} \quad (3)$$

where θ_t, θ_t^- are the parameters of the Goal-FCN and the corresponding target network at time t respectively. The target network is updated with a polyak averaging of 0.999. Similar to [25], we pass gradients only through a single pixel i and thus all other pixels backpropagate zero loss. We train the Goal-FCN using the Adam optimizer with learning rate 10^{-4} and with a discount factor of $\gamma = 0.5$. The model is trained in PyTorch with an Nvidia GeForce RTX 3060. Furthermore, we use prioritized experience replay [20] to balance the data that update the network.

Our exploration strategy is ϵ -greedy with the ϵ initialized at 0.9 then annealed to 0.05. To make the exploration more efficient, we devise prior constraints to reduce the complexity of the action space by performing random exploration and guided exploration with equal probability. During the guided exploration we aim at sample actions that either move the target object or the obstacles from the singulation area. We enforce the initial position \mathbf{p}_1 to be around the objects of the scene. Specifically, we randomly sample this position as pixels that are around either the target or the obstacles of the scene as shown in figure Fig. 3. The angle θ is computed by the line that starts from \mathbf{p}_1 and ends at \mathbf{p}_2 .

During training we generate random goals in order to learn a general goal-reaching policy. This allows to perform data augmentation inspired by hindsight experience replay [1] and thus make the learning more efficient and overcome the difficulties with the sparse rewards. In particular, after experiencing the

episode s_0, s_1, \dots, s_T we store to the replay buffer the non-empty transitions i.e. the transitions that led to a change in the scene, not only with the original goal but also with a goal generated by the position of the target object in state s_T if the target is singulated. Since the goal being pursued does not influence the environment dynamics, we can replay each trajectory using arbitrary goals, assuming we use an off-policy RL algorithm.

5 Results

The objective of the experiments is to evaluate the performance of the policy in the depalletizing task, i.e. the success rate and the mean number of actions before singulating all the objects.

5.1 Evaluation Metrics

To evaluate the policies we run N episodes. An episode is considered successful if all the objects of the workspace are totally singulated within T_{max} timesteps. In any other case (end of maximum timesteps or fall out of the surface) the episode is considered as failed. For Env-1, $T_{max} = 30$, while $T_{max} = 35$ for Env-2, due to its increased difficulty. We evaluate the performance with 2 metrics 1) the average success rate % over N episodes, which measures the ability of the policy to complete the task, i.e. to singulate all the objects and 2) the average mean number of actions over N episodes, until the task completion.

5.2 Simulation Experiments

We use the Bullet physics engine [3] to advance simulation. The simulation environment consists of a UR5 robot arm with a rectangular robotic finger, a 0.5×0.5 workspace and a simulated 640×480 RGB-D camera in a top view. The objects used in these simulations include 3 different object types, boxes, bottles, cylinders the shapes and dimensions of which are randomly chosen during experiments. We use $d_{singG} = 3$ cm and keep all the dynamic parameters of the simulation fixed.

Environments We extensively test the performance of the above policy in two environments. Both environments depict a grid like, initially cluttered scene, where the geometry of the objects is randomly created. The first environment (Env-1) is a 2×2 grid. To test the robustness of the policy in a more cluttered environment, we increase the number of obstacles to 6, a 2×3 grid (Env-2). While we train the policy in Env-1 we evaluate it in both the environments to test its generalization ability.

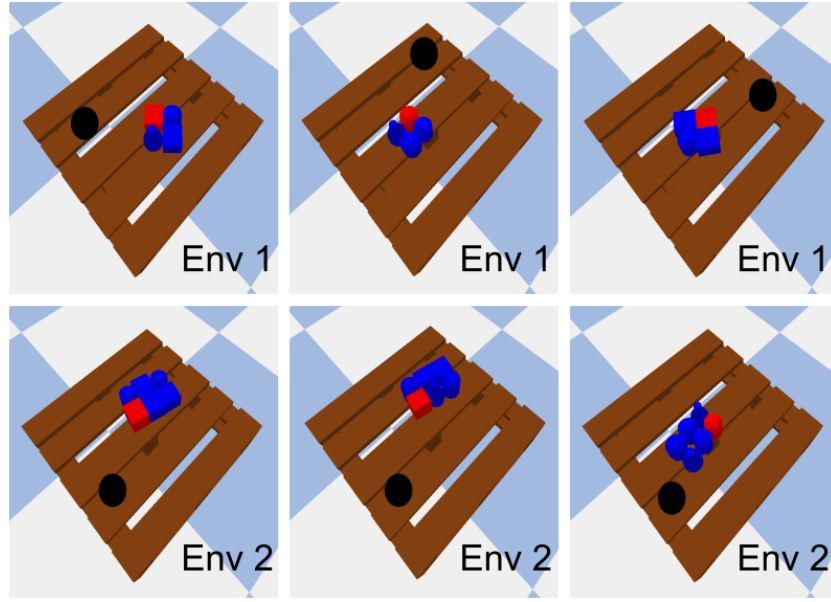


Fig. 4. Samples of scenes from the two different environments, Env 1 and Env 2 that the policies were evaluated. The target object is highlighted in red, the rest of the objects are depicted in blue, while the black areas are the goal regions of the scenes.

Table 1. Performance evaluation on different environments (Success rate %/Mean actions)

Environment	success rate	mean actions
Env-1	93.6	14.5
Env-2	83.3	22.9

Performance Evaluation After training the model for 10000 episodes in environment 1, the policy is evaluated for 1000 episodes with different scenes in both environments. Table 1 shows qualitative results for each environment. As we see the results show the efficacy of our RL, Goal driven approach, achieving high success rates in Env-1, without a significant drop of success rate in Env-2. Thus, our model has the ability to generalize in a more challenging environment. Fig. 5 shows the relationship between the number of actions and the success rate for each environment. The trends of both curves align, indicating a similar progression. However, it's notable that the success rate in the more challenging Env-2 demonstrates a slower increase, requiring a larger number of steps to achieve comparable results

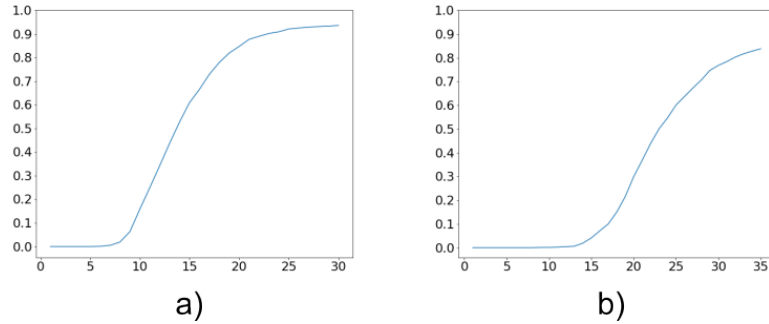


Fig. 5. Number of actions before singulation in the two different environments: (a) Env-1 (b) Env-2.

6 Conclusions

In conclusion, our proposed methodology, based on RL through the Goal-FCN network, offers a solution to the problem of depalletization. By performing non-prehensile pushing actions and incorporating goal regions, our approach demonstrates precision in handling diverse 3D objects within cluttered environments. In our next steps we aim to develop a policy which identifies the optimal goal region, resulting to a more effective unloading process.

Acknowledgments. This work was supported by the European Commission’s Horizon Europe Program through the SESTOSENSE project (<http://sestosenso.eu/>, HORIZON-CL4-DigitalEmerging Grant 101070310)

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., Zaremba, W.: Hindsight experience replay. *Advances in neural information processing systems* **30** (2017)
2. Boularias, A., Bagnell, J.A., Stentz, A.: Learning to manipulate unknown objects in clutter by reinforcement. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015)
3. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning (2016)
4. Dogar, M., Srinivasa, S.: A framework for push-grasping in clutter. *Robotics: Science and systems VII* **1** (2011)
5. Dogar, M.R., Srinivasa, S.S.: A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots* **33**(3), 217–236 (2012)

6. Eitel, A., Hauff, N., Burgard, W.: Learning to singulate objects using a push proposal network. In: *Robotics research*, pp. 405–419. Springer (2020)
7. Fontanelli, G.A., Paduano, G., Caccavale, R., Arpentì, P., Lippiello, V., Villani, L., Siciliano, B.: A reconfigurable gripper for robotic autonomous depalletizing in supermarket logistics. *IEEE Robotics and Automation Letters* **5**(3), 4612–4617 (2020). <https://doi.org/10.1109/LRA.2020.3003283>
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 770–778 (2016)
9. Holz, D., Topalidou-Kyniazopoulou, A., Stückler, J., Behnke, S.: Real-time object detection, localization and verification for fast robotic depalletizing. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 1459–1466 (2015). <https://doi.org/10.1109/IROS.2015.7353560>
10. Huang, B., Han, S.D., Yu, J., Boularias, A.: Visual foresight tree for object retrieval from clutter with nonprehensile rearrangement. *IEEE Robotics and Automation Letters* pp. 1–1 (2021). <https://doi.org/10.1109/LRA.2021.3123373>
11. Khairuddin, U., Razi, N.A.Z.M., Abidin, M.S.Z., Yusof, R.: Smart packing simulator for 3d packing problem using genetic algorithm. *Journal of Physics: Conference Series* **1447**(1), 012041 (jan 2020). <https://doi.org/10.1088/1742-6596/1447/1/012041>, <https://dx.doi.org/10.1088/1742-6596/1447/1/012041>
12. Kiatos, M., Malassiotis, S.: Robust object grasping in clutter via singulation. In: *2019 International Conference on Robotics and Automation (ICRA)*. pp. 1596–1600. IEEE (May 2019). <https://doi.org/10.1109/ICRA.2019.8793972>
13. Kiatos, M., Sarantopoulos, I., Koutras, L., Malassiotis, S., Doulgeri, Z.: Learning push-grasping in dense clutter. *IEEE Robotics and Automation Letters* **7**(4), 8783–8790 (2022). <https://doi.org/10.1109/LRA.2022.3188437>
14. Krug, R., Stoyanov, T., Tincani, V., Andreasson, H., Mosberger, R., Fantoni, G., Lilienthal, A.J.: The next step in robot commissioning: Autonomous picking and palletizing. *IEEE Robotics and Automation Letters* **1**(1), 546–553 (2016). <https://doi.org/10.1109/LRA.2016.2519944>
15. Kurenkov, A., Taglic, J., Kulkarni, R., Dominguez-Kuhne, M., Garg, A., Martín-Martín, R., Savarese, S.: Visuomotor mechanical search: Learning to retrieve target objects in clutter. In: *2020 IEEE International Conference on Intelligent Robots and Systems (IROS)* (2020)
16. Nakamoto, H., Eto, H., Sonoura, T., Tanaka, J., Ogawa, A.: High-speed and compact depalletizing robot capable of handling packages stacked complicatedly. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 344–349. IEEE (2016)
17. Novkovic, T., Pautrat, R., Furrer, F., Breyer, M., Siegwart, R., Nieto, J.: Object finding in cluttered scenes using interactive perception. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 8338–8344. IEEE (2020)
18. Sarantopoulos, I., Kiatos, M., Doulgeri, Z., Malassiotis, S.: Split deep q-learning for robust object singulation. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. pp. 6225–6231 (2020)
19. Sarantopoulos, I., Kiatos, M., Doulgeri, Z., Malassiotis, S.: Total singulation with modular reinforcement learning. *IEEE Robotics and Automation Letters* **6**(2), 4117–4124 (2021)
20. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015)

21. Stüber, J., Zito, C., Stolkin, R.: Let's push things forward: A survey on robot pushing. *Frontiers in Robotics and AI* **7**, 8 (2020). <https://doi.org/10.3389/frobt.2020.00008>
22. Tanaka, J., Ogawa, A., Nakamoto, H., Sonoura, T., Eto, H.: Suction pad unit using a bellows pneumatic actuator as a support mechanism for an end effector of depalletizing robots. *ROBOMECH Journal* **7**(1), 1–30 (2020)
23. Yang, Y., Liang, H., Choi, C.: A Deep Learning Approach to Grasping the Invisible. *IEEE Robotics and Automation Letters* **5**(2), 2232–2239 (apr 2020). <https://doi.org/10.1109/LRA.2020.2970622>
24. Zeng, A., Song, S., Lee, J., Rodriguez, A., Funkhouser, T.: Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics* **36**(4), 1307–1319 (2020)
25. Zeng, A., Song, S., Yu, K.T., Donlon, E., Hogan, F.R., Bauza, M., Ma, D., Taylor, O., Liu, M., Romo, E., Others: Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In: *IEEE International Conference on Robotics and Automation (ICRA)*. pp. 1–8. IEEE (2018)