Learning Multi-action Tabletop Rearrangement Policies

Bingjie Tang, Gaurav S. Sukhatme

***Abstract*— The ability to rearrange a physical environment depends to varying degrees on perceptual skill, the ability to navigate unstructured environments, manipulate objects effectively, and long-horizon task planning. Previous studies on rearrangement are restricted by object-centric assumptions and either deal with sparse arrangement of objects or do not adapt to different goal configurations. We propose a feature-based method that jointly learns two action primitives and a rearrangement planning policy in a table-top setting. Two separate fully-connected networks map visual observations to actions and another deep neural network learns rearrangement planning conditioned on the goal specification, perceptual input and selected action primitive. We directly compare our method with a state-of-the-art model in simulation and achieve comparable results on general rearrangement tasks. We show that our system can handle more challenging settings (non-singulated objects and object swaps) which the state-of-art-model cannot.**

I. INTRODUCTION

Rearrangement is a canonical task since it integrates multiple perception and manipulation skills that are necessary to build sophisticated robots that can carry out complex tasks with minimum human supervision in unstructured environments [1]. Generally speaking, the goal of rearrangement tasks is to bring a physical environment to a specified state. In this work, we focus on the tabletop object rearrangement problem, as represented in Fig. 1.

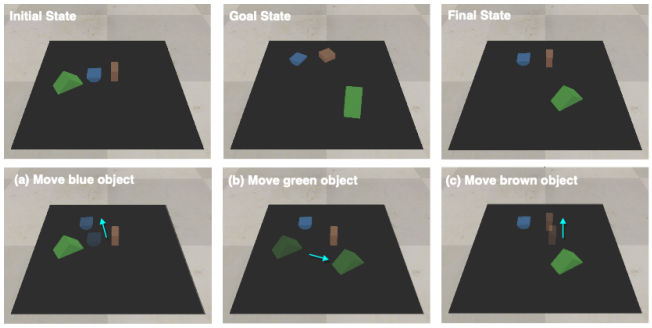


Fig. 1: **Overview.** Given a goal specification image and RGB-D images of the current scene from the camera, our system predicts a sequence of actions that can transition the current object arrangement to the goal configuration.

Previous studies in task and motion planning have explored this problem while relying heavily on object-centric assumptions, e.g. known object models and object segmentation. However, in real life tasks, robots usually encounter diverse object shapes and placement densities. In such environments, the performance of most object-centric methods will be compromised, e.g. model-based methods do not generalize well to novel objects and densely cluttered object arrangements will result in noisy object segmentation. Therefore, we propose an end-to-end feature-based approach to learn two action primitives, and rearrangement planning through trial-and-error, that can generalize to novel objects and complete rearrangement tasks in densely cluttered environments.

We jointly learn two action primitives: PUSH and GRASP , and a rearrangement PLACE policy in a deep Q-learning framework. Through pre-trained vision models, we first extract dense feature maps for the current RGB-D image from the camera, and the goal specification image. Three separate deep neural networks are used to predict the pixel-wise Q-value maps respectively for PUSH actions, GRASP actions and rearrangement PLACE actions. We incorporate correlation convolution as part of both the GRASP network and the rearrangement PLACE network which captures the similarity between the current feature map and the goal feature map by leveraging spatially-consistent visual representations. We directly pass the dense feature map for PUSH through a fully convolutional network (FCN) to get the best pushing pose that maximizes future grasp success. Combining the dense feature map and similarity distribution and passing it through a FCN, the GRASP network picks the best grasp candidates while filtering out grasp poses for objects that are already at their goal positions. The rearrangement PLACE network learns to predict the best placement location by maximizing the similarity between local visual features of the grasped object and the goal specification image feature map, while avoiding placement at positions that are already occupied by objects. The main contributions of our work are:

- A feature-based model that maps directly from pixels to actions to reposition objects. The model prioritizes the movement of objects that are not at their goal positions.
- An end-to-end planning approach with no object-centric assumptions, e.g. object segmentation, that can predict the placement for grasped objects to complete rearrangement tasks with different goal configurations.
- Our method achieves comparable results on general rearrangement tasks with a state-of-the-art model in simulation. Additionally, it handles more challenging settings (non-singulated objects and object swaps) which the state-of-art-model struggles with.

Our system is trained through interacting with simulated object arrangements under self-supervision. We evaluate

Both authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA 90089. bingjiet|gaurav@usc.edu. GS holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

our system on a simulated UR5 robot and demonstrate rearrangement policies (learned offline in simulation) on a Franka Panda robot arm.

II. Background & Related Work

A. Primitive Learning

Classic model-based methods for robotic action primitives such as [2]–[5], have been successful in structured environments such as warehouses, while heavily relying on known object shapes, poses, or manually-engineered motion planning. However, manipulating unknown objects in unstructured environments has been identified as one of the fundamental skills for general-purpose robots that can carry out long-horizon complex tasks in different settings. Previous studies focused on data-driven methods have leveraged deep learning, especially deep reinforcement learning, for robot object-agnostic action learning that can generalize to novel objects for a single action primitive [6], [7], as well as for the synergies between prehensile and non-prehensile skills [8], [9]. While these learning-based methods mainly focus on model's accuracy and efficiency for executing actions (e.g. success rate), the learning objective in our system includes additional prioritization for long-horizon rearrangement tasks.

B. Rearrangement

Rearrangement of unknown objects in an unstructured environment boils down to several sub-tasks for the robot: infer information from perceptual data, navigate itself in the environment, manipulate objects precisely and solve multi-step task planning [1].

In the task and motion planning (TAMP) literature, there has been significant work on rearrangement tasks that tackles the planning problem with predefined transformations instead of perceptual inputs, e.g. push or pick-and-place an object of known shape and pose [10]–[12]. Another thread of research on rearrangement tasks involves perception [13], [14], which means the system takes in raw sensory input and outputs actuation commands. Our method falls into the category of such end-to-end approaches since we take raw visual input from camera. Vision for robotic manipulation or task and motion planning initially were applied for pose estimation and object detection [3], [5], [15], [16]. End-to-end models that integrate these vision-based object-centric methods show great performance and sample efficiency while restricted by the object-centric representations when encounters unknown objects or adversarial environments [17], [18]. Other vision-based end-to-end approaches to multi-step planning tasks remove object-centric assumptions by leveraging spatially consistent visual feature correlations [14], [17], [19], [20]. Their work learns policies from expert demonstrations while our model learns from sparse reward functions. We only train our model on general tasks, and show that it generalizes to more challenging settings, e.g. object swap and cluttered environment, without explicit demonstration on such tasks.

Our work is most closely related to NeRP [18] wherein a deep neural network-based approach is proposed that can rearrange unseen objects on a tabletop. NeRP achieves state-of-the-art results for tabletop rearrangement, but requires segmented visual data as input, and struggles when the scene segmentation quality drops and fails to get object-correspondence from perceptual data. Our method can still capture feature-correspondence in such scenarios (e.g. cluttered environments) and complete the task. Also NeRP learns from previously stored expert demonstrations and our approach learns through sparse reward.

III. Learning multi-action rearrangement

A. Problem Formulation

We formulate the tabletop rearrangement problem as a Partially Observable Markov Decision Process (POMDP). A POMDP is a 7-tuple $(S, A, T, R, \Omega, O, \gamma)$ where $s \in S$ denotes a state and the state space, $a \in A$ denotes an action and the action space, T is a set of conditional transition probabilities between states, $R : S \times A \to \mathbb{R}$ is the reward function, Ω denotes a set of observations, O denotes a set of conditional observation probabilities, and $\gamma \in [0, 1]$ is the discount factor. In this task, we define the state s as the positions and orientations of objects in the workspace. The actions $a \in A$ consist of the choice of action primitive ψ, the end-effector position x and orientation θ:

$$a = (\psi, x, \theta), \psi \in \{\text{PUSH}, \text{GRASP}, \text{PLACE}\}, x, \theta \in \mathbb{R}^3.$$

The reward function for PUSH and GRASP is a sparse reward function - 1 for successful grasps and 0.5 for successful pushes. We consider a GRASP successful if the antipodal distance between parallel-jaw gripper fingers after a GRASP attempt is lower than a pre-defined threshold. We consider a PUSH successful if it makes changes to the scene - determined by whether the pixel-wise difference in the depth image after a PUSH is larger than a pre-defined threshold. The reward for PLACE is defined as the variance of average distance to goal positions of all objects after a PLACE :

$$r_t = (\sum_i^N d_i^t - \sum_i^N d_i^{t-1})/N, \quad (1)$$

where d_i^t represents the euclidean distance between object i's current position and its goal position at time t, N is the total number of objects in the workspace. Observation is defined as the RGB-D image captured by an ego-centric view camera. For each observation o_t, we rotate it in 16 different directions and use it as input to predict the pixel-wise Q-value map for executing actions with different orientations.

The goal specification is given by an RGB-D image of the goal object arrangement captured by the camera from the same ego-centric viewpoint. Our model learns to transform the initial object arrangement to match the goal specification image with two transformations: PUSH and pick-n-place which includes GRASP and PLACE .

B. Learning PUSH & GRASP

Given the current observation of the scene o_t, i.e. the RGB-D image captured by the ego-centric camera at time t, we use fully-connected neural networks (FCNs) to model Q-functions

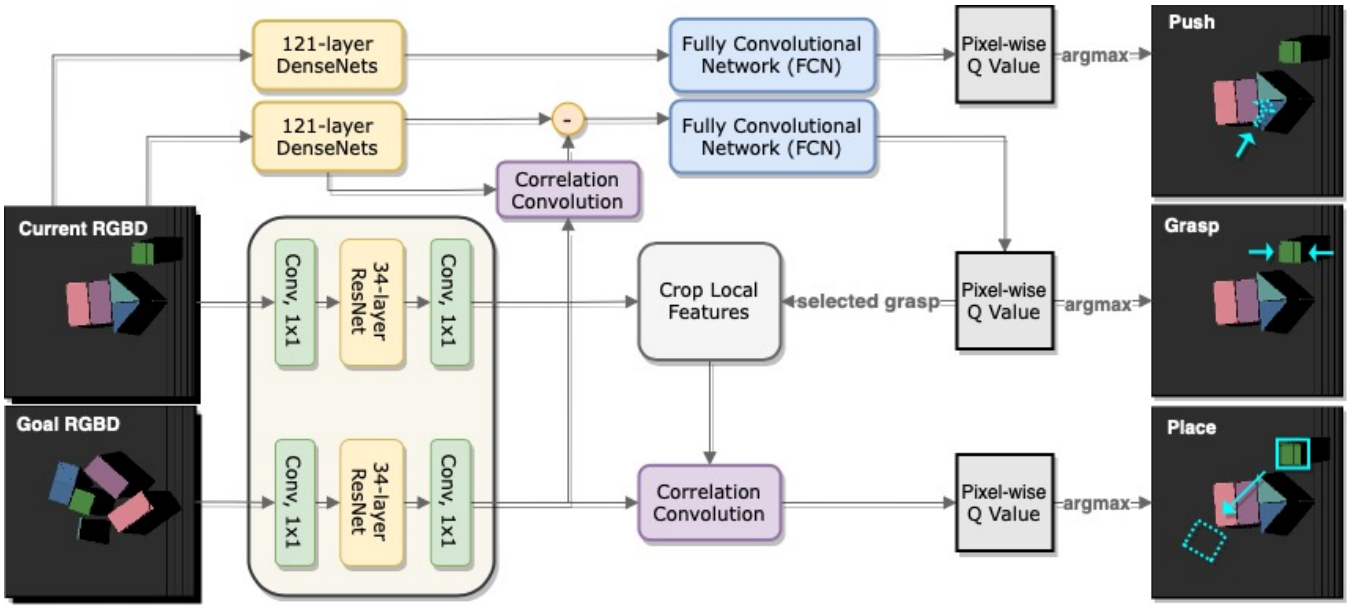


Fig. 2: **System overview.** Our system takes in an RGB-D visual observation of the scene and a goal-specification RGB-D image. It predicts the next action primitive (i.e. PUSH or GRASP), including the position and orientation for executing that action. When it executes a GRASP and is successful, it predicts the best position and orientation to PLACE the grasped object that maximizes the expected rearrangement reward defined in Eq. 1.

that estimate the expected reward for each action candidate. The network structure is shown in Fig. 2. The 121-layer DenseNet [21] module contains two separate DenseNets that are pretrained on ImageNet [22] for RGB and depth feature extraction respectively. In each FCN module, we have two 1×1 convolutional layers; we apply batch normalization and ReLU activation before every convolutional layer. After FCN, we upsample with bilinear mode to have a pixel-wise Q-value estimate of the same size as input images. Each pixel unit in the Q-value map corresponds to the expected reward for executing an action at this pixel location.

At each timestep t, the robot picks the action with the highest Q-value and calculates loss by computing the temporal difference (TD) between the estimated reward and the actual obtained reward after execution. Note that we only compute the loss for the selected pixel/pose (where the robot will take the next action), all other pixels/poses backpropagate with loss 0. We generate the label for PUSH at time t, y_t^{PUSH} , by calculating the depth image difference after the push, if it is higher than a predefined threshold we consider the PUSH successful. For GRASP , we obtain the label at time t, y_t^{GRASP} , via the feedback signal from the gripper. We use Huber Loss for both action primitives. For the executing action at time t, let y_t denotes the label, Q_t denote the estimated reward, the TD is given by $|Q_t - y_t|$, loss is calculated as:

$$L = \begin{cases} \frac{1}{2}(Q_t - y_t)^2, & |Q_t - y_t| < 1, \\ |Q_t - y_t| - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{2}$$

For GRASP , we add a correlation convolution with the goal specification image visual feature map. Based on the intuition of dense feature template matching, we calculate cross-correlation between dense feature maps for the goal image and the current image. Let $\psi(o_t)$ denote the dense feature map of current visual obeservation o_t extracted by 121-layer DenseNet and $\phi(o_g)$ denote the dense feature map of goal image o_g extracted via 34-layer ResNet. The correlation convolution output can be represented as:

$$\varphi_{\text{grasp}} = \psi(o_t) * \phi(o_g).$$

Since each pixel in o_t represents a grasp candidate, φ_{grasp} captures the visual similarity between o_t and o_g over all available grasp candidates. We subtract φ_{grasp} from the feature map in order to lower the Q-value for areas that are highly similar to the goal image. We further pass it through a fully convolutional network (FCN) to get a pixel-wise Q-value map for grasping actions. In this case, without an extra object prioritizing mechanism, the robot can avoid repetitively picking up objects that are already near their goal positions while maximizes the potential grasp success.

C. *Learning* PLACE *for Rearrangement*

Similar to [14], we consider learning PLACE policy as training a deep neural network to predict the Q-value map at time t given the current observation o_t, the goal specification o_g and the successfully executed GRASP at time $t-1$. If the previous executed action is not a GRASP or the previous executed GRASP failed, we do not execute the PLACE action.

The PLACE policy learning objective is to find the best placement for the grasped object. We pass the visual observation at time $t-1$, i.e. the RGB-D image before we execute GRASP , and the goal specification o_g through two separate neural networks to obtain the dense visual feature maps $\phi(o_{t-1})$ and $\phi(o_g)$. Each of these two networks consists

of a convolutional layer, a 34-layer ResNet which is pre-trained on ImageNet [22], and another convolutional layer as shown in Fig. 2. Given the executed GRASP τ_{t-1}, we crop a partial dense feature map on $\phi(o_{t-1})$ with a predefined crop window size centered at τ_{t-1}. Let the cropped dense feature map be $\phi(o_{t-1})[\tau_{t-1}]$. Intuitively, if $\phi(o_{t-1})[\tau_{t-1}]$ captures the visual features of the grasped object, the system needs to use $\phi(o_{t-1})[\tau_{t-1}]$ as a template and find the best matching local features to it. The correlation convolution between $\phi(o_{t-1})[\tau_{t-1}]$ and $\phi(o_g)$ can output a feature similarity distribution showing the resemblance between $\phi(o_{t-1})[\tau_{t-1}]$ and the local features at every placement in $\phi(o_g)$:

$$\varphi_t^{\text{similarity}} = \phi(o_{t-1})[\tau_{t-1}] * \phi(o_g).$$

We also apply correlation convolution between depth images of the goal configuration o_g^{depth} and o_t^{depth}:

$$\varphi_t^{\text{depth}} = \phi(o_t^{\text{depth}}) * \phi(o_g^{\text{depth}}),$$

which outputs a pixel-wise distribution over the workspace indicating whether a pixel location is occupied by objects. The pixel-wise Q-value map is calculated by:

$$\varphi_t^{\text{place}} = \varphi_t^{\text{similarity}} - \varphi_t^{\text{depth}},$$

where we avoid placing the grasped object on top of other object by lowering the Q-value for occupied pixels. The location in $\phi(o_g)$ that has the highest similarity and is not occupied by other objects at the same time is considered as the best PLACE τ_p for the grasped object:

$$\tau_p = \arg\max \varphi_t^{\text{PLACE}}.$$

φ_t^{PLACE} is a pixel-wise Q-value map and each pixel represents a potential placement for the grasped object.

We also use Huber Loss for learning the PLACE policy. The label y_t^{PLACE} is given by Eq. 1, $y_t^{\text{PLACE}} > 0$ when the robot moves the object closer to its goal position. The temporal difference is given by $|\varphi_t^{\text{PLACE}} - y_t^{\text{PLACE}}|$. The loss can be calculated as same as in Eq. 2. Similar to PUSH and GRASP, we only backpropagate the selected pixel with the loss value and any other pixels backpropagate with loss 0.

IV. Evaluation

A. Simulation Configuration

We use a simulated UR5 robot with parallel-jaw gripper in the CoppeliaSim [23] simulator, as shown in Fig.3.

B. General Tasks

In general task scenarios, we measure our system's ability to complete rearrangement tasks with different numbers of objects and directly compare our results with the state-of-the-art model [18]. In each scenario with N objects in the scene, we generate a new goal arrangement and a new initial arrangement with randomly picked object positions, orientations, shapes and colors. We train our system with 5-object scenarios till convergence over 3000 iterations. In each iteration, the robot executes one action (e.g. PUSH, GRASP, PLACE). We added novel shapes that the robot has not seen during training in the testing scenarios, which shows our model's ability to generalize to novel shapes.

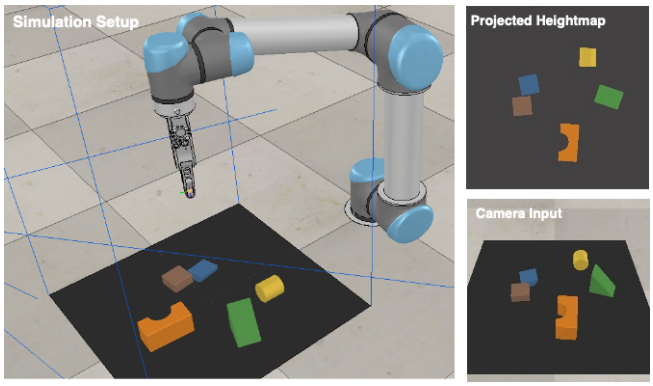


Fig. 3: **Simulation configuration.** A RGB-D camera with resolution 640×480 is placed to provide visual inputs from an ego-centric view of the workspace. We compute the point cloud using camera intrinsics, and assign the color value for every point through pixel-wise matching with the RGB image. We calculate the heightmap by orthographically back-projecting the point cloud upwards in the gravity direction.

We evaluate our system by the general GRASP success rate, the GRASP success rate after PUSH, the overall task completion rate and rearrangement planning step count. We do not directly evaluate PLACE as an action primitive because we assume the robot always successfully place the object at the selected pose. However, the quality of PLACE prediction can be inferred from results shown in Sec.IV-B.2.

1) General GRASP *success rate and* GRASP *success after* PUSH*:* These two performance metrics show the system's ability to accurately execute action primitives. General GRASP success rate is defined as the ratio of successful grasps in all grasp attempts, which measures the accuracy of the GRASP policy. GRASP success rate after PUSH is defined as the ratio of successful grasps executed directly after a PUSH action, which indicates the potential benefits to GRASP success via helpful PUSH actions. General GRASP success rate and GRASP success rate after PUSH over the training process are shown in Fig. 4. In Fig. 4, we observe that primitive learning converges around 1500 iterations and reaches ∼80% of both general GRASP success rate and GRASP after PUSH success rate. Also, GRASP after PUSH success rate converges slightly higher than general GRASP success rate which shows some PUSH actions improve future GRASP success. We also report the average GRASP success and GRASP to PUSH ratio over all testing scenarios in Table I. GRASP to PUSH ratio is defined as the number of executed grasps divided by the number of executed pushes. The accuracy of executing GRASP actions and the number of PUSH actions can both potentially affect the number of planning steps in each task completion.

2) Task completion rate and planning steps: These two performance metrics show the system's ability to finish the rearrangement task through sequential decision making. We consider an episode to be complete when the maximum error between any object's current position and its goal position is less than 5 cm. This means for any object i in the scene

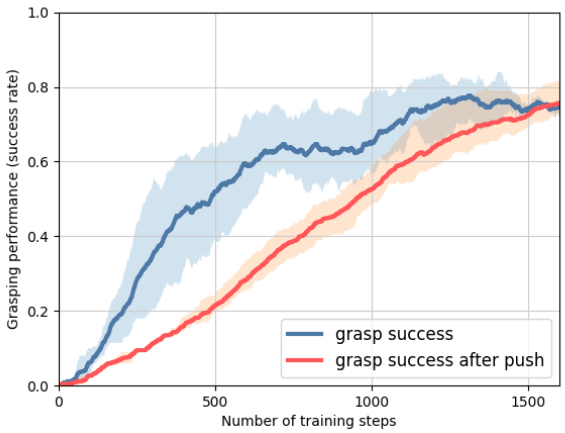


Fig. 4: **Success rates.** GRASP success rate and GRASP success rate after a PUSH both increase during the training process.

GRASP Success (%)	78.40 ± 2.90
GRASP to PUSH Ratio (%)	80.70 ± 1.80

TABLE I: Quantitative evaluation results for two primitive actions. Numbers reported are average over all tests.

if (x_t^i, y_t^i) is object i's current position, (x_g^i, y_g^i) is its goal position, when an episode is completed: $x_t^i \in [x_g^i - 0.05, x_g^i + 0.05], y_t^i \in [y_g^i - 0.05, y_g^i + 0.05]$. We count the average steps in each completion to show the planning efficiency of the rearrangement policy. Table II shows quantitative results over 50 randomly generated test scenarios where we directly compare our system with the state-of-the-art model NeRP [18]. We achieve competitive results with NeRP and our model's completion rate does not drop as much as NeRP when the workspace becomes more cluttered (i.e. more objects in the scene). With an average grasp success rate of 78.4% (Table I) and PUSH , we expect the number of planning steps executed by our model to be higher than NeRP.

Model	NeRP [18]		Ours	
Obj.	Completion	Steps	Completion	Steps
3	98.25 ± 0.57	4.58 ± 0.82	97.46 ± 0.85	5.37 ± 0.09
4	97.60 ± 1.20	5.70 ± 1.38	95.34 ± 1.57	9.02 ± 1.18
6	98.09 ± 0.40	8.69 ± 2.15	92.50 ± 1.11	9.30 ± 0.50
7	90.62 ± 1.03	9.47 ± 2.23	93.75 ± 1.28	12.57 ± 2.18

TABLE II: **Comparison between NeRP [18] and our method.** We achieve comparable completion rate (%) and average planning steps in scenarios that contain different number of objects but do better than NeRP in cluttered scenes. Statistics on NeRP are reported in their original paper. Both models are trained on random rearrangement of 5 objects.

C. Challenging Tasks: Swap

In this task setting the goal positions of certain objects are occupied by other objects in the initial arrangement. This requires the robot to first move the "placeholder" and then put the target object at its goal position. In special cases, it requires the robot to swap two objects in order to achieve the goal configuration.

We test our model with 3-object scenarios where the blue object and the green object always occupy each others' goal positions in the initial arrangement. A third (brown) object is added in the scene for a sanity check. We expect the robot to be able to successfully rearrange the brown object since its goal position is unoccupied. In each scenario, we generate a new goal arrangement for all three objects and obtain a new initial arrangement by switching the positions of the blue object and the green object, and randomly choosing an initial position for the brown object.

We evaluate our model over 50 test scenarios, and report the average success rate and the number of planning steps in Table III. We also show an example completion of our model in Fig. 5. With 64.5% task success rate, our model can complete the 3-object rearrangement task within 5.6 steps. Note that with the swapping action required, for 3-object rearrangement tasks the optimal solution costs 4 steps. As stated in Sec. III, our system can only predict the best placement location and the orientation in the xy-plane. Therefore, when an object's orientation changes in the xz-plane or the yz-plane, such as the brown object shown in Fig. 5, our model cannot adjust it to the exact goal configuration shown in the goal image.

D. Challenging Tasks: Clutter

In this task setting we start with initial configurations of objects that are densely cluttered. This requires the robot to use PUSH actions to singulate objects or to prioritize GRASP actions for already singulated objects first. In object-centric methods, densely cluttered scenes usually result in noisy object segmentation which severely lowers the performance of these algorithms and grasp-only systems struggle when no feasible grasp pose is available in a densely cluttered environment. Our system leverages PUSH actions that singulate objects from clutter for future GRASP success.

Let O be an object arrangement and $\{(x_1, y_1), ..., (x_n, y_n)\}$ denote n objects' position in O, To distinguish if an object arrangement is cluttered, we define the clutter coefficient of an object arrangement O as $c(O)$:

$$c(O) = -\log\{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)\},\ \hat{y}_i = \mathbf{kNN}(x_i)$$

in which $\mathbf{kNN}(x_i)$ estimates y_i through k-nearest neighbors regression on every other object's position in the scene. Clutter coefficient is calculated as the negative logarithm of the mean squared error (MSE) for all predictions $\hat{y}_i$. When objects are closer to each other (i.e. the scene is more cluttered), MSE decreases and $c(O)$ increases. We consider object arrangements with $c(O) \geq 2.0$ (MSE< 0.01) as 'cluttered'. 3 example object arrangements are shown in Fig. 7.

To test our model's ability to rearrange from a cluttered initial configuration, in each test scenario, we randomly pick 5 shapes and colors, generate a random goal arrangement and a random initial arrangement with clutter coefficient value larger than 2.0. We test our system on 50 5-object test

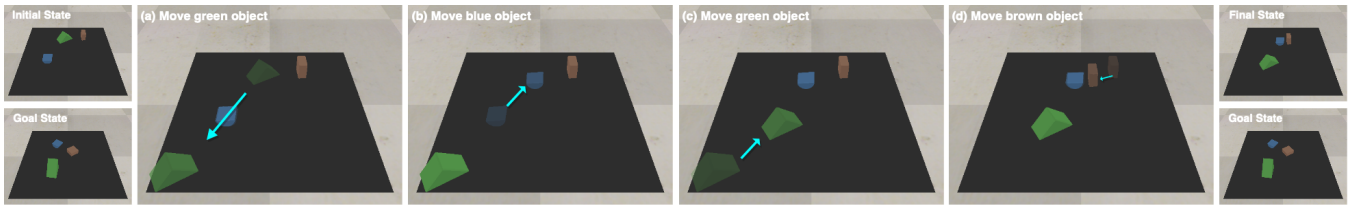


Fig. 5: **Swapping two objects.** In this 3-object scenario, we intentionally set the green object and the blue object to occupy each others' goal positions in the initial arrangement. Therefore the robot needs to swap these two objects.

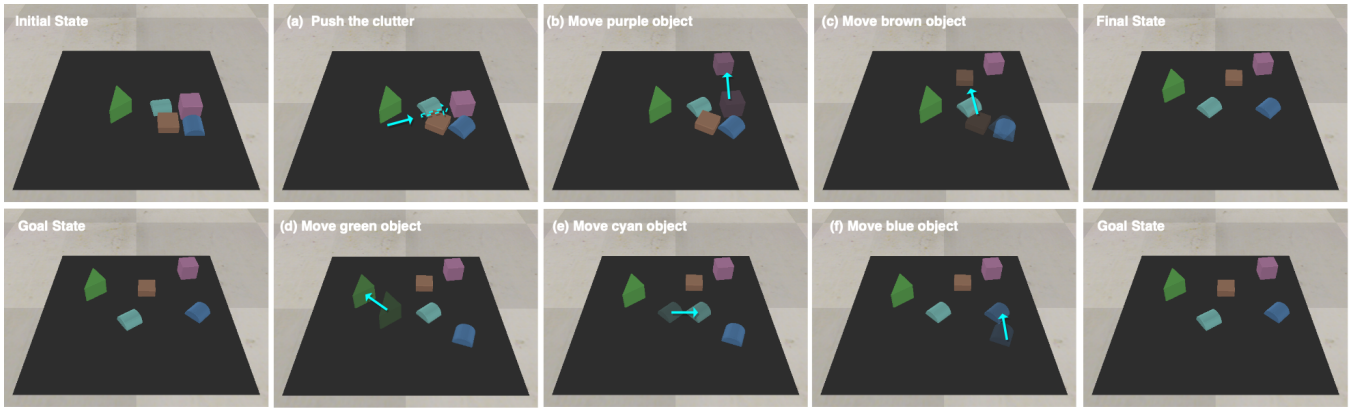
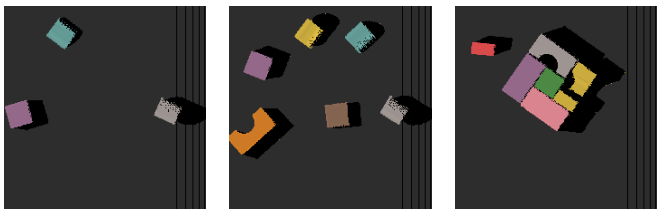


Fig. 6: **Rearranging 5-object clutter.** In this 5-object scenario, we intentionally set 4 objects close together in the initial state (clutter coefficient:2.04) so a pre-grasp PUSH is required.



(a) 1.69, non-clutter (b) 1.82, non-clutter (c) 2.54, clutter

Fig. 7: **Measuring clutter.** 3 example arrangements with clutter coefficient values.

scenarios; the results are reported in Table III. We achieve 86.67% average task success rate within 10.46 planning steps. The number of planning steps increases because there are collaborative PUSH actions involved in the task completion.

Task	Success Rate (%)	Planning Steps
Swap	62.50 ± 1.21	5.6 ± 0.34
Clutter	86.67 ± 1.42	10.46 ± 1.19

TABLE III: **Challenging tasks.** Quantitative evaluation results for challenging rearrangement tasks. Note that we do not re-train our model on these specific settings, we use the same model trained with general 5-object scenarios.

E. Real Robot Demonstration

We illustrate the execution of the system on a Franka Panda robot with a parallel-jaw gripper (Fig.8) wherein the arrangements and action trajectories in the real robot demonstration are collected in simulation (the robot demonstration is thus an example execution trace of a plan learned offline).

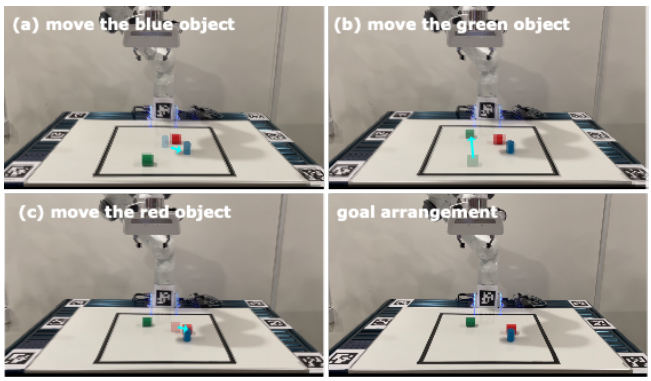


Fig. 8: **Robot demonstration.** 3-object rearrangement.

V. Conclusion and Future Work

We presented a feature-based end-to-end approach to rearrange objects of unknown shape on an open tabletop with two jointly-learned transformations PUSH and GRASP and a learned rearrangement PLACE policy from visual input. Our model trains through a deep Q-learning framework while leveraging the spatial consistency in visual features. It shows competitive results with the state-of-the-art object-centric method and generalizes to more challenging rearrangement tasks including swapping objects, and repositioning dense non-singulated cluttered arrangements. Our model is limited to positions and orientations in the xy-plane which results in misalignment around the z-axis at the goal pose. We plan to address this in future work and are actively exploring extending our vision-only method to incorporate other perceptual modalities to tackle more challenging settings.

REFERENCES

[1] "Rearrangement: A Challenge for Embodied AI," *ArXiv*, vol. abs/2011.01975, 2020.

[2] K. M. Lynch, "Estimating the friction parameters of pushed objects," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1993.

[3] Y. Yoon, G. N. DeSouza, and A. C. Kak, "Real-time tracking and pose estimation for industrial objects using geometric features," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, 2003.

[4] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE transactions on robotics*, 2005.

[5] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[6] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

[7] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational grasp generation for object manipulation," in *2019 IEEE/CVF International Conference on Computer Vision*.

[8] 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)*, 2018.

[9] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning*, 2018.

[10] O. Ben-Shahar and E. Rivlin, "Practical pushing planning for rearrangement tasks," *IEEE Transactions on Robotics and Automation*, 1998.

[11] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push planning for object placement on cluttered table surfaces," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, 2011.

[12] M. Stilman, J.-U. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proceedings 2007 IEEE international conference on robotics and automation*, 2007.

[13] M. Danielczuk, A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg, "Mechanical search: Multi-step retrieval of a target object occluded by clutter," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019.

[14] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," *Conference on Robot Learning (CoRL)*, 2020.

[15] M. Gupta and G. Sukhatme, "Interactive perception in clutter," in *The RSS 2012 Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012.

[16] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[17] D.-A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[18] A. Qureshi, A. Mousavian, C. Paxton, M. Yip, and D. Fox, "Nerp: Neural rearrangement planning for unknown objects," in *Proceedings of Robotics: Science and Systems*, 2021.

[19] C. Paxton, Y. Barnoy, K. Katyal, R. Arora, and G. D. Hager, "Visual robot task planning," in *2019 international conference on robotics and automation (ICRA)*, 2019.

[20] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Learning to smooth and fold real fabric using dense object descriptors trained on synthetic color images," in *International Conference on Robotics and Automation (ICRA)*, 2021.

[21] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, 2012.

[23] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly V-REP): A versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems*.