Feature-based Multi-action Tabletop Rearrangement

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 struggles with.**

I. INTRODUCTION

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

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 generate visual feature maps of the current RGB-D image of the workspace 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 visual feature map for PUSH through a fully convolutional network (FCN) to get the best pushing pose that maximizes future grasp success. The GRASP network picks the best grasp candidates while filtering out grasp poses for objects that are already at their goal positions based on the current visual feature map and its correlation with the goal feature map. 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 visual feature map, while avoiding placement at positions that are already occupied by objects.

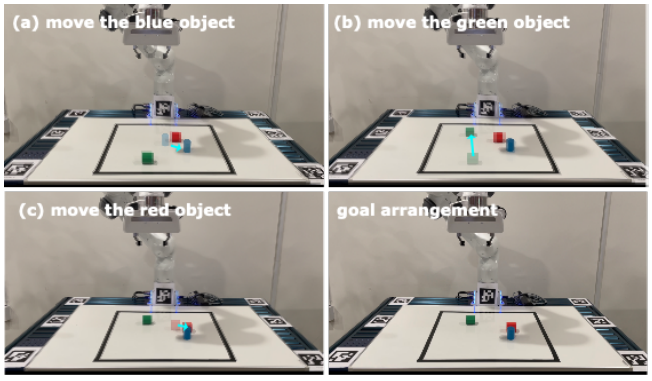


Fig. 1: **Overview.** Given a goal specification image and RGB-D images of the current scene from the camera, our system predicts a sequence of actions that can transition the current object arrangement to the goal configuration.

The main contributions of our work are:

- An end-to-end learning-based planning approach with no object-centric assumptions, e.g. object segmentation, that maps directly from pixels to actions to reposition objects and prioritize the rearrangement of objects that are not at their goal positions.
- 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 our system on a simulated UR5 robot and demonstrate rearrangement policies (learned offline in simulation) on a Franka Panda robot arm.

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.

II. Background & Related Work

A. Primitive Learning

Model-based methods for robotic action primitives [2], [3], [4], have been successful in structured environments, while heavily relying on known object shapes, poses, or manually-engineered motion planning. However, manipulating unknown objects in unstructured environments is a fundamental skill 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 [5], [6], and for synergies between prehensile and non-prehensile skills [7], [8]. While these learning-based methods have mainly focused on model accuracy and efficiency for executing actions (e.g. success rate), as we will discuss, the learning objective in our system includes additional prioritization for long-horizon rearrangement tasks.

B. Rearrangement

Rearranging unknown objects in an unstructured environment reduces to several sub-tasks for the robot: infer information from perception, navigate 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 [9], [10], [11], [12], [13], [14], [15]. So called 'end-to-end' approaches [16], [17] take in raw sensory input and directly produce actuation commands. Our work also falls into this category. Vision for robotic manipulation or task and motion planning initially were applied for pose estimation and object detection [3], [18], [4], [19]. End-to-end models that integrate these vision-based, object-centric methods show great performance and sample efficiency, but perform poorly when they encounter unknown objects or adversarial environments due to the object-centric representations they employ [20], [21], [22]. Other vision-based end-to-end approaches to multi-step planning tasks remove object-centric assumptions by leveraging spatially consistent visual feature correlations [20], [17], [23]. and learning policies from expert demonstrations.

Our work is most closely related to NeRP [21] 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 successfully captures feature-correspondence in such scenarios (e.g. cluttered environments) and completes the task. In contrast to NeRP, which learns from previously stored expert demonstrations, our approach learns from sparse reward. Further, our model is trained only on general rearrangement tasks, and we show that it generalizes to more challenging settings, e.g. object swaps and cluttered environments, without explicit demonstration on such tasks.

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 $\mathbf{x}$ and orientation θ:

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

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 higher than a pre-defined threshold. Since we only use PUSH to singulate cluttered objects in order to enable future GRASP actions, and we do not consider PUSH itself as a rearrangement action, we designed our PUSH reward to encourage changes in the scene. PUSH is successful if the pixel-wise difference in the depth image after it is executed is larger than a pre-defined threshold. The reward for PLACE is defined as the variance of average distance to goal positions of all N objects after a PLACE :

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

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

The goal specification o_g 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 three actions: PUSH , 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 that estimate the expected reward for each action candidate. The network structure is shown in Fig. 2. The 121-layer DenseNet [24] module contains two separate DenseNets that are pretrained on ImageNet [25] for RGB and depth feature extraction respectively. In each FCN module, we have two 1×1 convolutional layers with batch normalization and ReLU

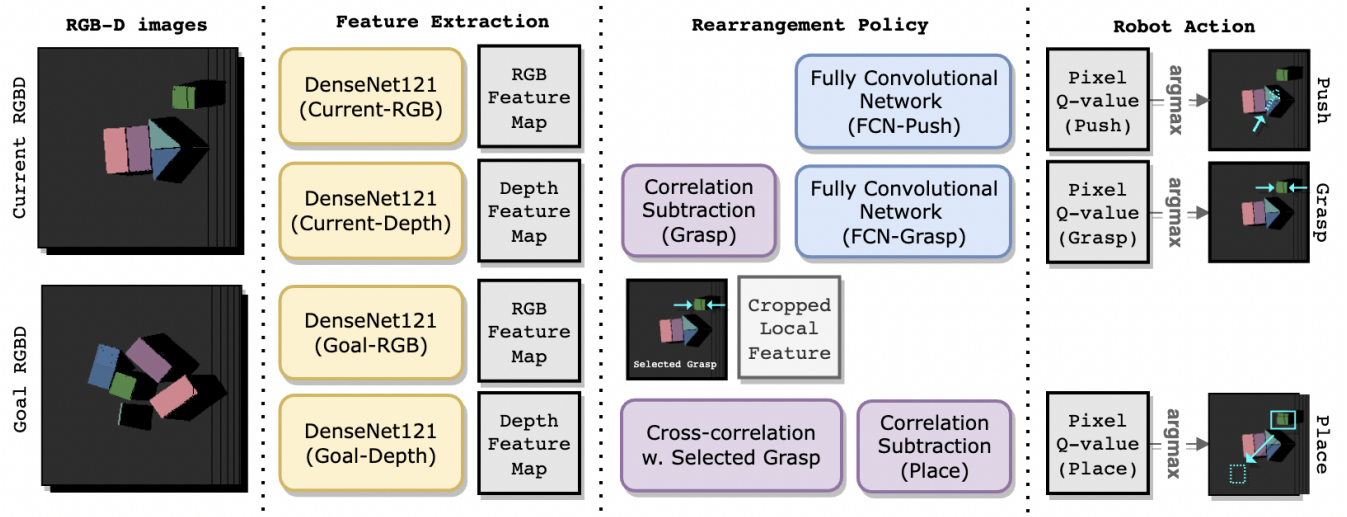


Fig. 2: **System overview.** Our system takes in current and goal RGB-D image of the workspace and predicts the next action (i.e. PUSH or GRASP), along with the position and orientation for executing that action.

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. 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 y_t^{PUSH} for PUSH at time t by calculating the depth image difference after executing the action. If the difference is higher than a predefined threshold we consider the PUSH successful, i.e. $y_t^{\text{PUSH}} = 1$, otherwise $y_t^{\text{PUSH}} = 0$. For GRASP , we obtain the label y_t^{GRASP} at time t via the feedback signal from the gripper, if GRASP is successful, $y_t^{\text{GRASP}} = 1$, otherwise $y_t^{\text{GRASP}} = 0$. 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:

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

Zeng et al. [7] proposed Visual Pushing Grasping (VPG) which also uses deep neural networks to map visual observations to PUSH and GRASP actions. VPG [7] is designed for tabletop decluttering, so its only optimizing objective is the GRASP success rate. However, for rearrangement tasks, objects that are already at their goal positions, despite their GRASP accessibility, should not be repetitively picked up.

Let $\phi(o_t)$ and $\phi(o_g)$ denote the feature maps of current image o_t and goal image o_g extracted by separate 121-layer DenseNets. In our work, to avoid repetitive GRASP actions for correctly rearranged objects, we calculate the cross-correlation between $\psi(o_t)$ and $\phi(o_g)$:

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

Since each pixel represents a grasp candidate, φ_{grasp} captures the similarity between o_t and o_g over all GRASP candidates. We subtract φ_{grasp} from $\phi(o_t)$ resulting in lower Q-values for areas that are highly similar to the goal image:

$$\phi'(o_t) = \phi(o_t) - \varphi_{\text{grasp}}. \tag{3}$$

At locations where objects have been successfully rearranged (i.e. within error range of their goal positions), φ_{grasp} is high and hence by subtracting φ_{grasp} from $\phi(o_t)$, these locations have lower Q-values and the robot can avoid grasping these objects without an extra object sequencing mechanism. We further pass $\phi'(o_t)$ through a FCN to get a pixel-wise Q-value estimate for GRASP and calculate loss as shown in Eq.2. By minimizing TD-error between Q-value estimate and GRASP execution outcome and applying correlation subtraction (Eq.3) at the same time, the robot learns a GRASP policy that maximizes the successful grasping while preventing redundant grasping.

C. *Learning* PLACE *for Rearrangement*

Similar to [17], we consider PLACE policy as predicting the Q-value distribution 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 a successful GRASP , we automatically assume the next action is to PLACE the grasped object. If the previous executed action is not a GRASP or if the previous executed GRASP failed, we do not execute the PLACE action, and pick the next PUSH or GRASP based on Q-value predictions.

The PLACE policy objective is to find the best placement for the grasped object. As shown in Fig.2, 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

separate pretrained 121-layer DenseNets[24] to obtain the visual feature maps $\phi(o_{t-1})$ and $\phi(o_g)$. Given the executed GRASP $\tau_{t-1}^{\text{grasp}}$, we crop a partial feature map $\phi(o_{t-1})[\tau_{t-1}^{\text{grasp}}]$ on $\phi(o_{t-1})$ with a predefined crop window size centered at $\tau_{t-1}^{\text{grasp}}$. Intuitively, if $\phi(o_{t-1})[\tau_{t-1}^{\text{grasp}}]$ captures the visual features of the grasped object, the system can use $\phi(o_{t-1})[\tau_{t-1}^{\text{grasp}}]$ as a template and find the best matching local features to it in the goal image. The cross-correlation between $\phi(o_{t-1})[\tau_{t-1}^{\text{grasp}}]$ and $\phi(o_g)$ can output a feature similarity distribution showing the resemblance between $\phi(o_{t-1})[\tau_{t-1}^{\text{grasp}}]$ and the local features at every placement in $\phi(o_g)$:

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

Different from Zeng et al. [17], we also apply cross-correlation between only depth images of 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 in the current scene or in the goal scene. The pixel-wise Q-value map is calculated by:

$$\varphi_t^{\text{place}} = \varphi_t^{\text{similarity}} - \varphi_t^{\text{depth}}. \quad (4)$$

where we avoid placing the grasped object on top of other objects or at other goal positions corresponding to other objects by lowering the value for occupied pixels. φ_t^{PLACE} is a pixel-wise Q-value map and each pixel represents a potential placement for the grasped object. 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_t^{place} for the grasped object: $\tau_t^{\text{place}} = \arg\max \varphi_t^{\text{PLACE}}$. To model the end-effector rotation of PLACE, we rotate the current image o_t in 16 different directions as input and pick the one with the highest Q-value prediction.

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 $|\max \varphi_t^{\text{PLACE}} - y_t^{\text{PLACE}}|$. 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. Experimental Evaluation

A. Simulation Configuration

We use a simulated UR5 robot with parallel-jaw gripper in the CoppeliaSim [26] 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 NeRP [21]. We generate new training scenarios following the method described in NeRP. In each scenario, 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

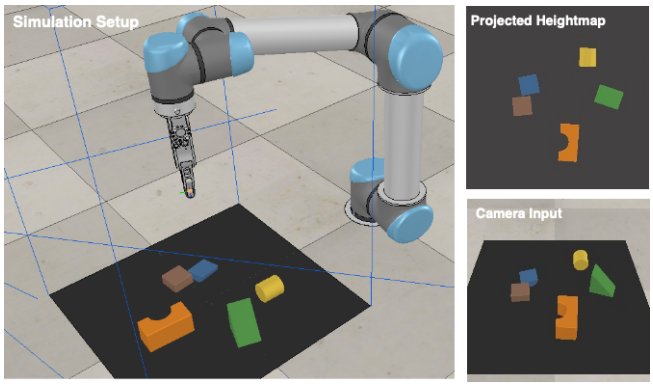


Fig. 3: **Simulation configuration.** A RGB-D camera provides visual inputs from an ego-centric view of the workspace.

5-object scenarios till convergence over 3000 iterations. In each iteration, the robot executes one action (e.g. PUSH , GRASP , PLACE). We test with novel shapes that the robot has not seen during training, which shows our model's ability to generalize.

Task completion rate and *planning steps* are measured to 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 at the end of each completed episode, for any object in the scene, the Euclidean distance between its current position to its goal position is less than 5cm. We count the average steps in each completion to show the planning efficiency of our policy. Table I shows quantitative results over 50 randomly generated test scenarios where we directly compare our system with the state-of-the-art model NeRP. In terms of *task completion rate*, we are competitive with NeRP in scenes with fewer objects and we outperform NeRP when the workspace becomes more cluttered (i.e. more objects in the scene). However, our system plans and executes more steps compared to NeRP due to collaborative PUSH actions and failed GRASP attempts. We further evaluate and discuss PUSH and GRASP as action primitives in Sec.IV-E.0.a.

Model	NeRP [21]		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 I: **Comparison with NeRP.** Both models trained on random rearrangement of 5 objects. NeRP statistics from [21].

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 reposition the object to its goal position.

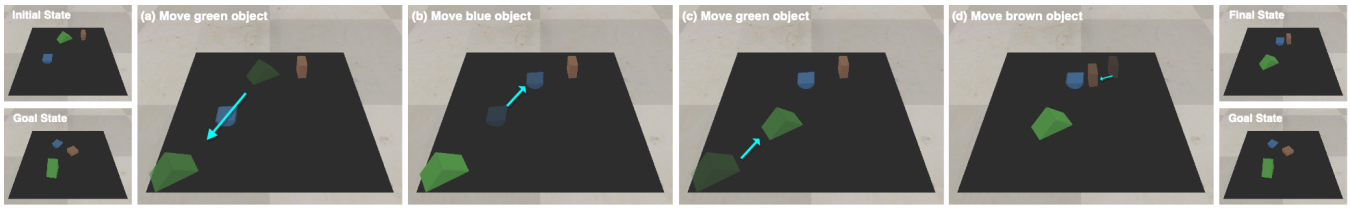


Fig. 4: **Swapping two objects.** In this 3-object scenario, we intentionally set the green object and the blue object to occupy each others' goal positions in the initial arrangement. Therefore the robot needs to swap these two objects.

We test our model with 3-object scenarios where the first object and the second object always occupy each others' goal positions in the initial arrangement. A third object is added to the scene for a sanity check. Since the goal position of the third object is unoccupied, the robot is expected to successfully rearrange it. 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 first two objects, and randomly choosing the third object's initial pose.

We evaluate our model over 50 test scenarios, and report the average *task completion rate* and *planning steps* in Table II. We also show an example completion of our model in Fig. 4. With 62.5% task success rate, our model can complete the 3-object rearrangement task within 5.60 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 only models the rotation of end-effector for PLACE 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. 4, our model cannot adjust it to the exact goal configuration shown in the goal image.

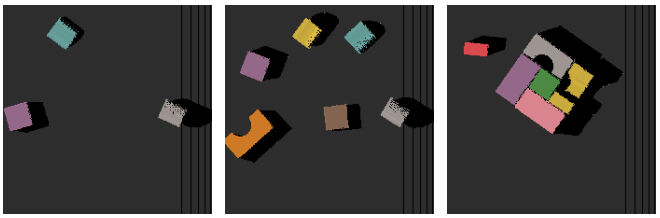
D. Challenging Tasks: Clutter

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 and prioritize rearrangement for already singulated objects. 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 addresses this challenge by leveraging PUSH actions that singulate objects from clutter for future GRASP success.

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

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

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 $c(\mathbf{O})$ increases. We consider object arrangements with $c(\mathbf{O}) \geq 2.0$ (MSE< 0.01) as 'cluttered'. 3 example object arrangements are shown in Fig. 5.



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

Fig. 5: **Measuring clutter.** 3 example arrangements with clutter coefficient values.

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 $\mathbf{O_g}$ and an initial arrangement $\mathbf{O_0}$ with $c(\mathbf{O_0}) \geq 2.0$. We test our system on 50 5-object test scenarios; the results are reported in Table II. We achieve 86.67% average task success rate within 10.46 planning steps. The average number of planning steps increases compared to the general tasks because there are collaborative PUSH actions involved in the task completion. An example clutter rearrangement is shown in Fig.6.

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

TABLE II: **Quantitative evaluation results for challenging rearrangement tasks.** Our model is not re-trained on these specific settings, we use the same model trained with general 5-object scenarios. NeRP [27] does not report swap scenario statistics, making a direct comparison infeasible.

E. Ablation Studies

We conduct ablation studies that explore different vision models for visual feature extraction and substantiate the necessity of correlation subtractions in our system.

a) Visual Feature Extractors: We believe the quality of visual features directly affects the model's accuracy for action execution. With three different vision models, we evaluate their impact on our system by *general* GRASP *success rate* and GRASP *success rate after* PUSH . *General* GRASP *success rate* is defined as the ratio of successful GRASP in all GRASP attempts. This is a measure of the accuracy of the GRASP policy.

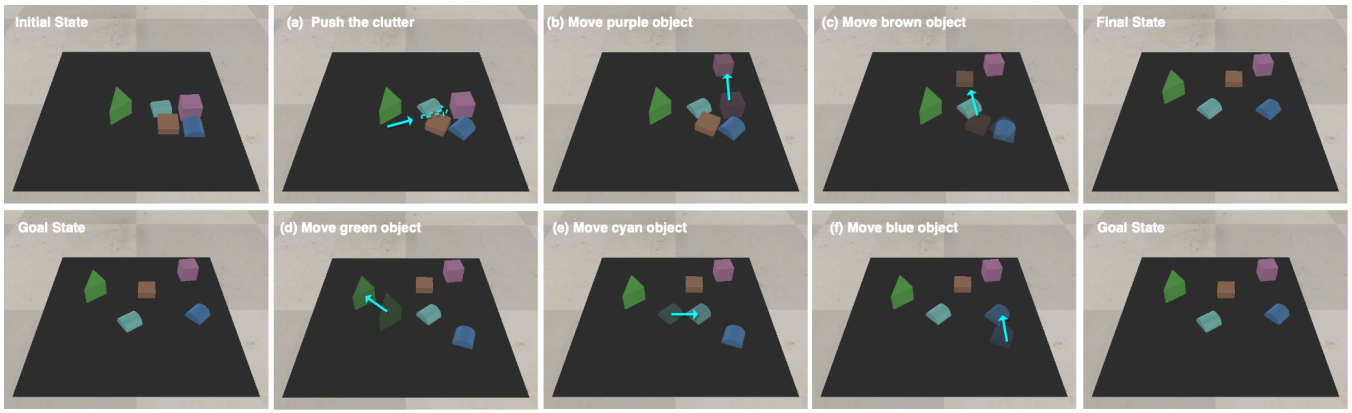
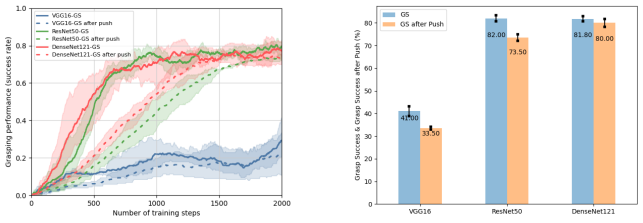


Fig. 6: **Rearranging in clutter.** 4 objects are clustered together (clutter coefficient: 2.04) so a pre-grasp PUSH is required.

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 that precede them. PLACE as an action primitive is not directly evaluated for accuracy because we assume the robot can always successfully release the grasped object at a given location. However, the quality of PLACE prediction can be inferred from results shown in Sec.IV-B. *General* GRASP *success rate* and GRASP *success rate after* PUSH over the training process are shown in Fig. 7a. In Fig. 7a, we observe that VGG16 [28] has significantly lower GRASP *success rate* than ResNet50 [29] and DenseNet121 [24]. Both ResNet50 and DenseNet121 converged after 1500 iterations and reached $\sim 80\%$ GRASP *success rate*. Only DenseNet121 shows improvement in GRASP *success rate after* PUSH , indicating its ability to learn a collaborative PUSH policy. We also report the average testing GRASP *success (GS)* and GRASP *success rate after* PUSH in Fig. 7b. There is no significant difference in average GRASP *success rate* between ResNet50 and DenseNet121; we pick DenseNet121 [24] as our visual feature extractor due to its higher GRASP *success rate after* PUSH so our model can solve cluttered scenarios where PUSH actions are needed to singulate objects. With an average GRASP *success* of 81.8% with DenseNet121 (Fig. 7b) and included collaborative PUSH actions, we expect the number of planning steps executed by our model to be higher than NeRP since 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.



(a) training (b) testing

Fig. 7: **Quantitative evaluation results (%) for PUSH and GRASP with different visual feature extraction models.** All vision models are pre-trained on ImageNet[25].

b) Correlation Subtraction: To demonstrate the necessity of applied correlation subtractions for GRASP (Eq.3) and PLACE (Eq.4), we conduct two separate ablation studies where we remove each correlation subtraction. In Figs.8 and 9, the heatmaps' rotation models the end-effector orientation when executing GRASP or PLACE . The centers of the red circles are the highest Q-value pixel locations in the heatmaps. An example of removing the GRASP correlation subtraction in Eq.3 is shown in Fig.8. Without Eq.3, the robot repeatedly grasps the green object due to its highest GRASP Q-value even though it is within the predefined error range from its goal position and there are other graspable objects that are not rearranged. By adding GRASP correlation subtraction, the brown object, which is not previously rearranged, becomes the highest GRASP Q-value object. We show an example of skipping the PLACE correlation subtraction in Eq.4 where we directly use $\varphi_t^{\text{similarity}}$ as PLACE Q-value estimate in Fig.9. Without Eq.4, the robot causes the brown and blue objects to collide since the goal position of the brown object is blocked by the blue object. However, by applying Eq.4, we avoid collision by placing the brown object at an intermediate position that is unoccupied and near its goal position.

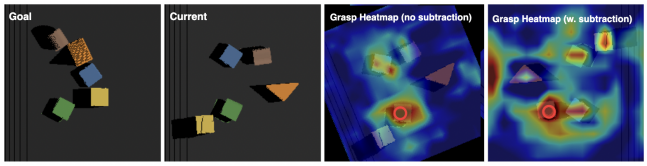


Fig. 8: **Correlation subtraction for GRASP .** The GRASP Q-value estimate for the green object remains the highest even though it is already at its goal position.

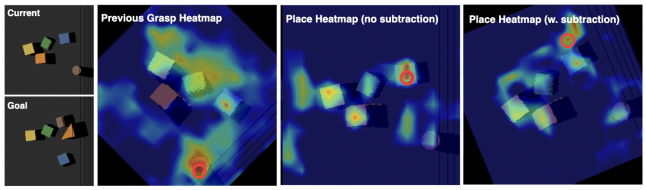


Fig. 9: **Correlation subtraction for PLACE .** The robot picks the brown object for rearrangement. Without Eq.4, the placement (at red circle) will collide with the blue object.

F. Real Robot Demonstration

We illustrate the execution of the system on a Franka Panda robot with a parallel-jaw gripper (Fig.1) 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).

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] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su, "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] 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.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] S. D. Han, N. M. Stiffler, A. Krontiris, K. E. Bekris, and J. Yu, "Complexity results and fast methods for optimal tabletop rearrangement with overhand grasps," *The International Journal of Robotics Research*, 2018.

[13] S. H. Cheong, B. Y. Cho, J. Lee, C. Kim, and C. Nam, "Where to relocate?: Object rearrangement inside cluttered and confined environments for robotic manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[14] R. Wang, K. Gao, D. Nakhimovich, J. Yu, and K. E. Bekris, "Uniform object rearrangement: From complete monotone primitives to efficient non-monotone informed search," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[15] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," in *IEEE International Conference on Robotics and Automation*, 2022.

[16] 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.

[17] 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.

[18] M. Gupta, J. Müller, and G. S. Sukhatme, "Using manipulation primitives for object sorting in cluttered environments," *IEEE Transactions on Automation Science and Engineering*, 2015.

[19] 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.

[20] 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.

[21] 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.

[22] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "Ifor: Iterative flow minimization for robotic object rearrangement," *arXiv preprint arXiv:2202.00732*, 2022.

[23] 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.

[24] 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.

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

[26] 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*, 2019.

[27] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robotics and Automation Letters*, 2020.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.