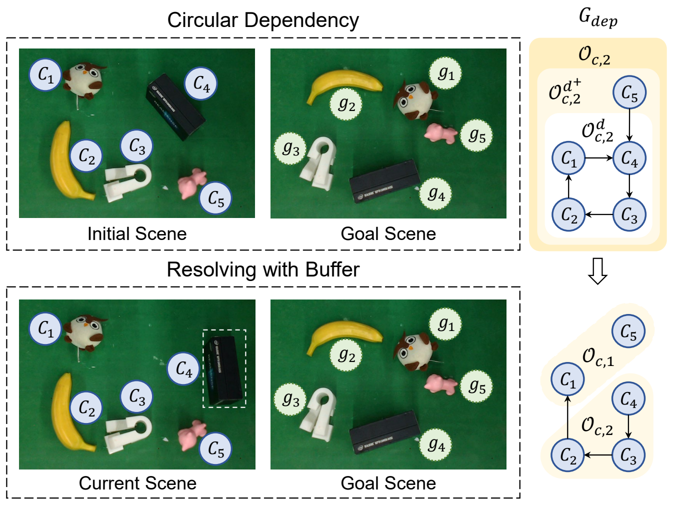

Fig. 1. An example to illustrate circular dependency, the corresponding dependency digraph, and buffer. After moving c_4 to the buffer (marked with the white box), the circular dependency breaks.

Appendix

A. *Proof of Theorem 1*

Circular Dependency. It is possible to have circular dependencies [1], [2]. An example is shown in Fig. 1. The objects cannot be moved to the goals until the circular dependency is broken. Formally, these objects form cycles in the graph G_{dep}, which we denote the set of circular dependent objects as $\mathcal{O}_{c,2}^{d}$:

$$\mathcal{O}_{c,2}^{d}: \{o_c^{i_0} | \exists o_c^{i_1}, o_c^{i_2}, ..., o_c^{i_n} \in \mathcal{O}_c, n \geq 1, \\ (o_c^{i_0}, o_c^{i_1}), (o_c^{i_1}, o_c^{i_2}), ..., (o_c^{i_n}, o_c^{i_0}) \in A_{dep}\}$$

Furthermore, if an object has a path directed to $\mathcal{O}_{c,2}^{d}$, it also cannot be moved to the goals before the circular dependency breaks. The set including this kind of objects, along with objects in $\mathcal{O}_{c,2}^{d}$, is denoted as $\mathcal{O}_{c,2}^{d+}$:

$$\mathcal{O}_{c,2}^{d+}: \mathcal{O}_{c,2}^{d} \cup \left\{o_c^{i_0} | \exists o_c^{i_1} \in \mathcal{O}_{c,2}^{d}, (o_c^{i_0}, o_c^{i_1}) \in A_{dep}\right\}$$

Then we have $\mathcal{O}_{c,2}^{d} \subseteq \mathcal{O}_{c,2}^{d+} \subseteq \mathcal{O}_{c,2}$. Once existing circular dependencies, some objects must be temporarily moved to the intermediate places to break the dependencies. These intermediate places are defined as buffers. A buffer can be a place that is currently not occupied and does not overlap with all goal regions. Fig. 1 shows an illustration of the buffer. We follow the assumption in [3] that the capacity of the buffer is infinite, which is practical since the workspace in daily life is large enough relative to the space occupied by the objects.

Optimal Number of Steps. In [3], the problem of resolving objects within circular dependency mirrors the Feedback Vertex Set (FVS) problem, which is a set of vertices whose removal makes G_{dep} acyclic. After resolution, all objects can be placed to goals without buffers. Then we have the optimal number of steps for object rearrangement proved in [3]:

Lemma 1. *Given the dependency graph G_{dep} of a tabletop object rearrangement problem of M objects, the minimum steps to resolve objects in $\mathcal{O}_{c,2}^{d}$ equals the size of the minimum FVS of G_{dep}. Let this size be $|B|$, the minimum total pick-n-place steps to achieve the goal configuration is $M + |B|$.*

Based on the lemma, we have the proof of Theorem 1:

Proof. For each object in the current scene, it belongs to $\mathcal{O}_{c,1}$ or $\mathcal{O}_{c,2}^{d+}$ or $\mathcal{O}_{c,2} \setminus \mathcal{O}_{c,2}^{d+}$. We denote $0/1$ as the existence of objects in these three sets, *e.g.* $\{111\}$ means the current scene contains objects in all three sets, resulting in 2^3 situations. Notably, there are paths from objects in $\mathcal{O}_{c,2} \setminus \mathcal{O}_{c,2}^{d+}$ to those in $\mathcal{O}_{c,1}$. Thus, the existence of objects in $\mathcal{O}_{c,2} \setminus \mathcal{O}_{c,2}^{d+}$ indicates the existence of objects in $\mathcal{O}_{c,1}$. In addition, the empty scene is not considered. Therefore, there are at most 5 cases for the current scene: $\{010\}, \{100\}, \{101\}, \{110\}, \{111\}$. Furthermore, we merge case 2 and case 3 as $\{10\cdot\}$, *i.e.* the current scene contains objects in $\mathcal{O}_{c,1}$, but does not contain objects in $\mathcal{O}_{c,2}^{d+}$, and merge case 4 and case 5 as $\{11\cdot\}$, *i.e.* the current scene contains both objects in $\mathcal{O}_{c,1}$ and $\mathcal{O}_{c,2}^{d+}$.

Let $\mathcal{F}^{\pi}(\mathcal{O}_c|\mathcal{O}_g)$ be the total steps of a policy π to reach the goal configuration $\mathcal{O}_g$ from $\mathcal{O}_c$. Following **Lemma 1**, an optimal policy π^* has $\mathcal{F}^{\pi^*}(\mathcal{O}_c|\mathcal{O}_g) = M + |B|$. To prove π^0 is optimal, we show that π^0 can achieve the minimum total pick-n-place steps in all cases:

1) Existence of both $\mathcal{O}_{c,1}$ and $\mathcal{O}_{c,2}^{d+}$, $\{11\cdot\}$.

In this case, there exist objects $o_c^i \in \mathcal{O}_{c,1}$, the policy π^0 grasps objects $o_c^i \in \mathcal{O}_{c,1}$ and directly places them to their goals. Note that moving an object $o_c^i \in \mathcal{O}_{c,1}$ may bring new objects whose goals are initially occupied by o_c^i into the set of $\mathcal{O}_{c,1}$. Such objects are then moved to their goals. That is, objects in $\mathcal{O}_{c,1}$ are iteratively moved to their goals until there is no object in $\mathcal{O}_{c,1}$. Let the number of these objects be M^0, then the number of steps to move them to the goals is M^0.

After these objects are all moved to their goals, there must remain only objects in $\mathcal{O}_{c,2}^{d+}$. To prove this, we assume there is an object $o_c^{i_0}$ that does not belong to $\mathcal{O}_{c,2}^{d+}$ after moving all objects in $\mathcal{O}_{c,1}$. That is, the reachable subgraph of $o_c^{i_0}$ is acyclic. So, there must be another object $o_c^{i_1}$ in the reachable subgraph of $o_c^{i_0}$, that does not depend on any other objects. As a result, $o_c^{i_1} \in \mathcal{O}_{c,1}$, which is in conflict with the assumption.

For objects in $\mathcal{O}_{c,2}^{d+}$, the policy π^0 first resolves them as $\pi_{\mathcal{G}}^{fvs}$, which costs $|B|$ times of place in the buffer. After resolving the circular dependency by $\pi_{\mathcal{G}}^{fvs}$, all the remaining $M - M^0$ objects can be moved to their goals without additional places in buffers. Therefore, the number of steps to move these objects to the goals is $M - M^0 + |B|$.

Finally, the total steps of the policy equals $M + |B|$.

2) Existence of $\mathcal{O}_{c,1}$, absence of $\mathcal{O}_{c,2}^{d+}$, $\{10\cdot\}$.

In this case, there is no circular dependency. That is, all the objects can be moved to the goals without buffers *i.e.* $|B| = 0$. Therefore, $\mathcal{F}^{\pi^0}(\mathcal{O}_c|\mathcal{O}_g) = M = M + |B|$.

3) Existence of only $\mathcal{O}_{c,2}^{d+}$, $\{010\}$.

In this case, the policy π^0 first resolves circular dependency as $\pi_{\mathcal{G}}^{fvs}$, which costs $|B|$ times of place in the buffer. Then, all the objects can be moved to their goals without additional places in buffers. Therefore, the number of steps to move all the objects to the goals is $M + |B|$.

Overall, we have $\mathcal{F}^{\pi^0}(\mathcal{O}_c|\mathcal{O}_g) = \mathcal{F}^{\pi^*}(\mathcal{O}_c|\mathcal{O}_g) = M + |B|$ in all possible cases, demonstrating that the policy π^0 is optimal. $\square$

B. Proof of Theorem 2

Proof. It has been stated in Eqn. 7 that with ideal perception, the optimal place policy $\bar{\pi}^0_{\mathcal{P}}$ follows a rule conditioned on the object matching derived from the grasp policy. With perception noise, the derived object matching may be wrong.

Given two current objects $o_c^{i_0}$, $o_c^{i_1}$ and their ground-truth matched objects $o_g^{j_0}$, $o_g^{j_1}$, assume that the object matching derived from $\pi^0_{\mathcal{G}}$ wrongly pairs $o_c^{i_0}$ to $o_g^{j_1}$. Instead, by improving the in-hand object matching independently, $\bar{\pi}^1_{\mathcal{P}}$ can get the correct matching of $o_c^{i_0}$ to $o_g^{j_0}$.

Now let $o_c^{i_0}$ be the in-hand object. If $o_g^{j_0}$ and $o_g^{j_1}$ are both occupied, π^0 and π^1 will both move $o_c^{i_0}$ to the buffer, then the wrong matching does not affect their pick-n-place steps. However, if $o_g^{j_0}$ is not occupied, π^1 will move $o_c^{i_0}$ to the goal. Instead, π^0 will move $o_c^{i_0}$ to the buffer or the goal of $o_c^{i_1}$, which calls for additional steps to move $o_c^{i_0}$ to its goal, or brings additional occupancy of the goal of $o_c^{i_1}$. Thus, for the rearrangement of $o_c^{i_0}$, π^0 costs more steps than π^1.

Notably, each time there is only one in-hand object, and the effect on total pick-n-place steps can be accumulated if there is more than one wrong matching of the in-hand objects. Therefore, $\mathcal{F}^{\pi^0}(\mathcal{O}_c|\mathcal{O}_g) \geq \mathcal{F}^{\pi^1}(\mathcal{O}_c|\mathcal{O}_g)$, indicating that the improvement of perception of the place policy is valuable to reduce the total pick-n-place steps, *i.e.* improve task-level performance.

□

C. More Implementation Details of See Policy

RL Algorithm Details. We train the see policy with SAC [4], [5]. The temperature parameter α is initialized as 0.2 with automatic entropy tuning. All the networks are trained with Adam optimizer using fixed learning rates 3×10^{-4}. The future discount γ is set as a constant at 0.99. Note that during the training stage, the network parameters of the finetuned CLIP are fixed.

Network Architecture. For finetuned CLIP, we use the CLIP-Adapter [6] containing a 3-layer MLP with sizes [512, 128, 512]. For see policy, the policy and critic share the same state encoder with separate MLPs to output action and Q value respectively. To be specific, for policy, we employ ResNet50 [7] to encode the state Δf, followed by MLPs to generate the mean and standard deviation of a normal Gaussian distribution. Subsequently, the action $a_{\mathcal{S}}$ is sampled from the distribution. Each angle of delta orientation is limited by the range of $[-1.57, 1.57]$. The MLPs of policy and critic are with sizes [1024, 1024, 3] and [1024, 1024, 1] respectively.

D. More Implementation Details of Grasp Policy

RL Algorithm Details. The reinforcement learning algorithm used to train our networks is discrete SAC [4], [5]. The temperature parameter α is initialized as 0.2 with automatic entropy tuning. All the networks are trained with Adam optimizer using fixed learning rates 3×10^{-4}. Our future discount γ is set as a constant at 0.99. Note that during the training stage, the network parameters of the finetuned CLIP and graspnet are fixed.

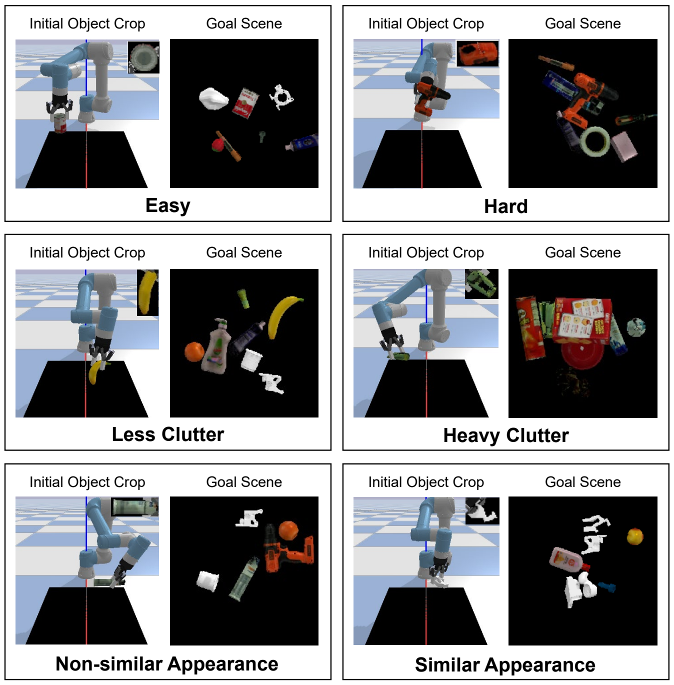

Fig. 2. Example testing cases of different categories in see.

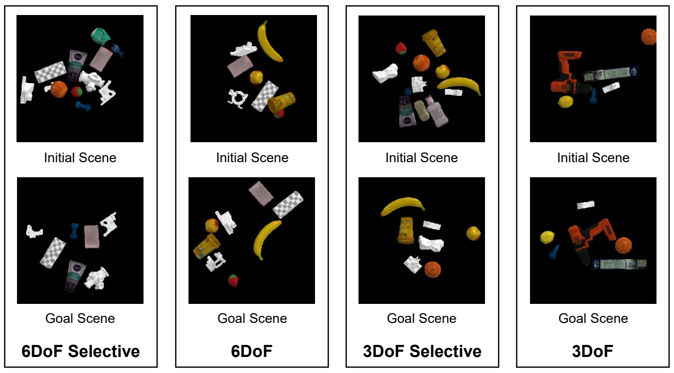

Fig. 3. Example testing cases of different categories in object rearrangement.

Network Architecture. We adopt the transformer architecture of the text encoder in [8], and conduct multi-head attention with different query, key and value. Parameters of each cross transformer include width of 512, head of 8 and layer of 1. The sizes of hidden layers of MLP_1, MLP_2 and MLP_3 are [256, 512, 512], [256, 512, 512] and [512, 256, 1].

E. Case Visualization

Fig. 2 shows example testing cases of see with different categories. Fig. 3 visualizes example testing cases of different categories in the tabletop rearrangement.

F. Generalization on Perception Noise

To demonstrate the generalization of our method in noisy perception, we conduct experiments with unseen camera angle and table background in Table I. The original camera angle is nearly top-down, while the newly tested camera overlooks the workspace from the front at a 36° downward angle. Additionally, we conduct experiments with an unseen wooden table background. To overcome worse object matching, the maximum see steps is extended from 5 to 10. Environment

TABLE I
SIMULATION RESULTS OF OBJECT REARRANGEMENTS ON GENERALIZATION OF PERCEPTION NOISE
(MAXIMUM SEE STEP=10)

Environment	Rotation	Selectivity	Init. #obj.	Goal #obj.	Task Completion		Completion Planning Steps		Overall Planning Steps	
					seen	unseen	seen	unseen	seen	unseen
Front Camera	3-DoF	✗	4-8	4-8	87.5	70.0	19.6±3.5	22.5±3.2	20.9±3.7	24.8±4.7
	3-DoF	✓	9-13	4-8	83.3	66.7	18.6±3.7	18.3±0.5	20.5±3.9	22.2±2.8
	6-DoF	✗	4-8	4-8	80.0	66.7	21.8±3.9	19.2±1.7	23.4±3.8	22.8±3.5
	6-DoF	✓	9-13	4-8	75.0	66.7	19.6±2.5	25.6±1.5	22.2±3.5	27.1±1.8
Wooden Background	3-DoF	✗	4-8	4-8	92.0	66.7	15.3±2.0	12.2±1.1	16.5±4.3	18.1±4.8
	3-DoF	✓	9-13	4-8	91.7	66.7	14.4±0.8	18.3±1.5	15.7±1.5	22.2±3.5
	6-DoF	✗	4-8	4-8	90.0	66.7	12.1±1.8	13.0±1.8	13.9±1.9	18.7±4.7
	6-DoF	✓	9-13	4-8	84.6	66.7	20.0±2.3	20.0±1.3	21.5±3.7	23.3±4.5

* All cases are with challenges of clutter and swap.

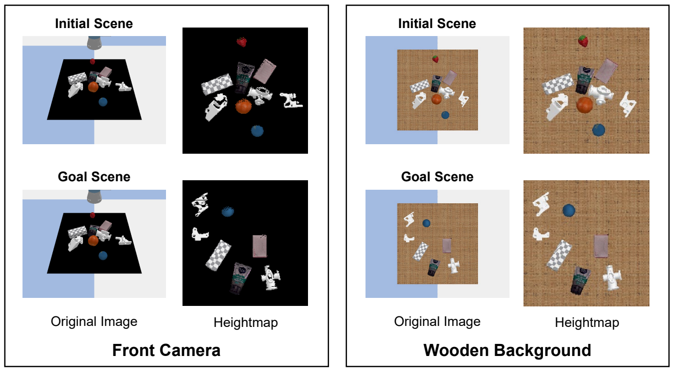

Fig. 4. Example testing cases of generalization on perception noise in object rearrangement.

settings of the unseen camera view and table background are visualized in Fig. 4. Results show that our method can generalize to the unseen camera view and the table background. For the unseen table background, our policy achieves similar completion rates with more planning steps compared to that with the original background. For the unseen camera angle, there are performance drops both in completion rates and planning steps.

For the unseen table background, we take the object-centric image crops as input, thereby the textured background slightly affects the M-to-N object matching. Accordingly, the performance of the grasp policy is hindered to some extent. Therefore, the policy needs to spend more planning steps. Additionally, by extending the maximum see steps, the see policy reduces the background influence on the in-hand object matching, prompting the whole policy to reach similar completion rates as that with the original background.

Our current model can adapt to unseen camera views by projecting the camera images into heightmaps as described in [9] (*i.e.* the top-down workspace image). However, the unseen camera view simultaneously affects the current and goal scenes. Even with the object-centric image crops, the impact is still significant. Due to the affected M-to-N object matching, the performance of the grasp policy is hindered, calling for more planning steps. Due to the affected in-hand object matching, even extending the maximum see steps, the improvement of active perception is inapparent. Therefore, compared to the results of the original camera view, there remains a $5\%\sim10\%$ margin in completion rates.

REFERENCES

[1] K. Gao, S. Feng, and J. Yu, "On minimizing the number of running buffers for tabletop rearrangement," *Robotics: Science and Systems XVII*, 2021.

[2] K. Gao, D. Lau, B. Huang, K. E. Bekris, and J. Yu, "Fast high-quality tabletop rearrangement in bounded workspace," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1961–1967.

[3] 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*, vol. 37, no. 13-14, pp. 1775–1795, 2018.

[4] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[5] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207*, 2019.

[6] P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao, "Clip-adapter: Better vision-language models with feature adapters," *arXiv preprint arXiv:2110.04544*, 2021.

[7] 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, pp. 770–778.

[8] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[9] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.