IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Tidiness Score-Guided Monte Carlo Tree Search for Visual Tabletop Rearrangement

Hogun Kee[1], Wooseok Oh[1], Minjae Kang[1], Hyemin Ahn[2], and Songhwai Oh[1]

Abstract—**In this paper, we present the tidiness score-guided Monte Carlo tree search (TSMCTS), a novel framework designed to address the tabletop tidying up problem using only an RGB-D camera. We address two major problems for tabletop tidying up problem: (1) the lack of public datasets and benchmarks, and (2) the difficulty of specifying the goal configuration of unseen objects. We address the former by presenting the tabletop tidying up (TTU) dataset, a structured dataset collected in simulation. Using this dataset, we train a vision-based discriminator capable of predicting the tidiness score. This discriminator can consistently evaluate the degree of tidiness across unseen configurations, including real-world scenes. Addressing the second problem, we employ Monte Carlo tree search (MCTS) to find tidying trajectories without specifying explicit goals. Instead of providing specific goals, we demonstrate that our MCTS-based planner can find diverse tidied configurations using the tidiness score as a guidance. Consequently, we propose TSMCTS, which integrates a tidiness discriminator with an MCTS-based tidying planner to find optimal tidied arrangements. TSMCTS has successfully demonstrated its capability across various environments, including coffee tables, dining tables, office desks, and bathrooms. The TTU dataset is available at: https://github.com/rllab-snu/TTU-Dataset.**

Index Terms—**Manipulation Planning, Data Sets for Robot Learning, Deep Learning Methods**

I. Introduction

TABLETOP tidying addresses the challenge of enabling an embodied AI agent to autonomously reorganize objects on a table according to their composition. Tidying up involves rearranging objects by determining an appropriate configuration of given objects, without providing an explicit target configuration. Previous research has encountered difficulties in defining the tidying up problem, primarily due to the lack of public datasets and metrics to assess tidiness. To address these issues, we collect a structured dataset for tabletop tidying, and train a tidiness discriminator and tidying planner to transform a messy table into an organized one through a simple and effective framework. We refer to this as the tidiness score-guided Monte Carlo tree search (TSMCTS).

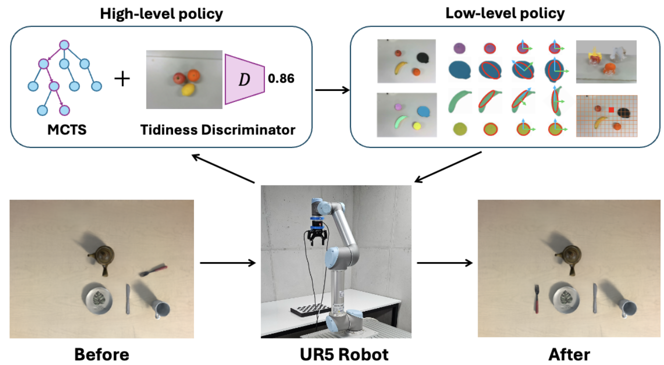

Fig. 1. The hierarchical policy of TSMCTS iteratively finds pick-and-place actions to tidy up objects on a table. The high-level policy finds which object to pick and place according to the current configuration. The low-level policy finds grasp points and trajectories of the end effector. Details of each policy are described in Section V.

In previous research on object rearrangement, goal configurations are provided either as target positions or as images of the desired arrangement [1]–[3]. This setup enables straightforward evaluation by comparing the state to the goal. However, using an image as a goal requires objects to be prearranged for goal generation, limiting flexibility. To accommodate a wider variety of goals, recent studies employ language labels or descriptions to define more abstract goals [4]–[6]. Nonetheless, these language-conditioned rearrangement studies require separate encoding modules to integrate language with image or positional inputs. To connect these domains, CLIP [7] and BLIP [8] models focus on mapping features into a unified latent space. While these approaches show promising performance in extracting semantic features [9], [10], they still struggle with understanding the spatial relationships among objects.

In this paper, we propose a novel tabletop rearrangement algorithm and a novel tabletop tidying up (TTU) dataset. As depicted in Figure 1, our proposed method, TSMCTS, learns a tidiness score function and finds a sequence of pick-and-place actions that generates a tidy configuration based on object combinations without explicit goals. We collect a tabletop tidying dataset from diverse environments (e.g., coffee tables, dining tables, office desks, bathrooms) and train a tidiness discriminator to measure the degree of tidiness reliably, even with unseen objects and real-world images. Then, we use the tidiness discriminator as a utility function of MCTS [11] to

Manuscript received February 20, 2025; Revised June 5, 2025; Accepted July 19, 2025.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II191190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning, 50%) and by IITP grant funded by the Korea government (MSIT) (No. RS-2024-00336738, Development of Complex Task Planning Technologies for Autonomous Agents, 50%). *(Corresponding author: Songhwai Oh.)*

[1]H. Kee, W. Oh, M. Kang and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: {hogun.kee, wooseok.oh, minjae.kang}@rllab.snu.ac.kr, songhwai@snu.ac.kr).

[2]H. Ahn is with Artificial Intelligence Graduate School (AIGS), Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea (e-mail: hyemin.ahn@unist.ac.kr).

Digital Object Identifier (DOI): see top of this page.

find a sequence of actions. The MCTS-based planner employs a tidying policy trained with our tabletop tidying up dataset using an offline reinforcement learning method, specifically Implicit Q-learning (IQL) [12], as its tree policy.

The proposed method, TSMCTS, achieves a tidying success rate of 88.5% in simulation experiments and 85% in real robot experiments. These results experimentally demonstrate TSMCTS's ability to tidy up various combinations of objects using both simulations and real robots. We also perform a human evaluation, which demonstrates that the proposed method can tidy up a table as much as humans can perceive it well-arranged. To this end, we propose a comprehensive framework for tabletop tidying that includes dataset collection, learning a tidiness metric, and planning object rearrangement on the table.

II. RELATED WORK

Tidying up is an object rearrangement problem occurring in situations where the goal is not explicitly provided. Instead of a specific goal, several research approaches involve expressing goals in natural language [4]–[6], or finding functional arrangements based on user preferences [13], [14]. Additionally, there are studies that directly learn the degree of tidiness as a score function and plan trajectories to achieve a tidied scene [15], [16].

Recent studies such as StructFormer [4] and StructDiffusion [5] find appropriate positions for objects guided by natural language instructions. Both methods take language tokens and object point clouds as inputs to find arrangements that satisfy language conditions. Studies such as [13] and [14] learn user preferences to find organized arrangements without explicit goals. These studies rely more on semantic information rather than visual information of the objects. However, due to the absence of an objective metric for evaluating rearrangement quality, these studies depend on user ratings or various heuristic criteria. In this context, [15] and [16] aim to quantify the degree of tidiness using a learned score function. [15] uses an energy-based model to learn a cost function, which is used to determine placements for missing objects. [16] learns a score function to estimate the likelihood under a target distribution for each task, and subsequently uses this score to train a policy for rearranging objects. Our approach is superior to these methods, as our discriminator can consistently measure tidiness from real-world images, whereas [15] and [16] primarily demonstrate success in simulation or toy examples rather than real-world scenarios.

Among the studies employing MCTS algorithm for rearrangement tasks, [6] uses the MCTS to find trajectories to obtain arrangements that satisfy language conditions. [6] assumes that explicit spatial conditions can be derived based on language conditions. In this paper, instead of relying on language, we propose a method that enables the agent to infer spatial conditions solely based on object information and perform tidying accordingly.

III. PROBLEM FORMULATION

In the tabletop tidying up problem, an agent (in our case, a robot) $\mathcal{M}$ is tasked to rearrange a set of movable objects $O = \{o_1, o_2, \ldots, o_N\}$ to achieve a tidy arrangement. At each timestep, the agent receives a single RGB-D image from a fixed overhead camera. The workspace is planar and all objects are assumed to be rigid bodies. The robot $\mathcal{M}$ interacts with objects through pick-and-place actions, performing arbitrary translations and rotations. In this paper, we will refer to the pick-and-place action as an action.

We assume there is a tidiness-score function Ψ which returns the degree of tidiness given an image of the tabletop with objects. This function assesses whether objects on the table are visually tidied up, considering the types, shapes, and sizes of the objects, and assigns a tidiness score between 0 and 1, where 0 represents a completely messy scene and 1 indicates a well-arranged scene.

For a given set of objects O, the visual observation depends on the pose of the objects. Therefore, we can formulate the tidiness score as $\psi = \Psi(O, P)$, where P denotes the 4-DoF positions of O. Then we can formulate the objective of the table tidying problem as finding the optimal arrangement P^* to maximize the tidiness score:

$$P^* = \arg\max_{P} \Psi(O, P). \tag{1}$$

In this paper, we parameterize a tidiness score function with neural networks θ, as a discriminator Ψ_θ. Then, Ψ_θ is trained to estimate the tidiness score of a table arrangement.

IV. TABLETOP TIDYING UP DATASET

We collect a Tabletop Tidying Up (TTU) dataset which includes both tidied and messy scenes to train a vision-based tidiness discriminator. To cover diverse object arrangements, we define a set of environments E consisting of four environments: Coffee table, Dining table, Office desk, and Bathroom. For each environment $e \in E$, we define a set of objects $O_e \subseteq O_{\text{all}}$ belonging to that environment, where O_{all} is the entire set of objects. Then, we predefine possible combinations of objects within O_e, each consisting of two to nine objects.

In this study, we introduce the concept of a template to encourage automatic collection of well-organized arrangement data. A template is defined as a specific set of spatial relationships between objects, categorized as one of the following: on, under, left, right, front, behind, left-front, left-behind, right-front, and right-behind. Figure 2-a shows examples of templates and their corresponding tidied arrangements for a set of objects $O = \{knife, fork, plate, cup\}$. By defining templates in this manner, we can represent all tidied arrangements that a person might create with specific templates.

We first collect templates for the given object sets and then use these templates to gather tidied scene data. We configure an appropriate combination of objects for each environment and design up to 16 templates for each set of objects. The entire process of finding templates is conducted manually by spawning object models on a table in the PyBullet simulation. We use 3D object models from YCB [17] and HouseCat6D [18]. To collect tidying sequence data, we first created tidied scenes based on templates, then generated untidying sequences by scattering the tidied objects one by one. After sampling a template, we create a tidied scene by augmenting the distances between objects, changing objects within the same category,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

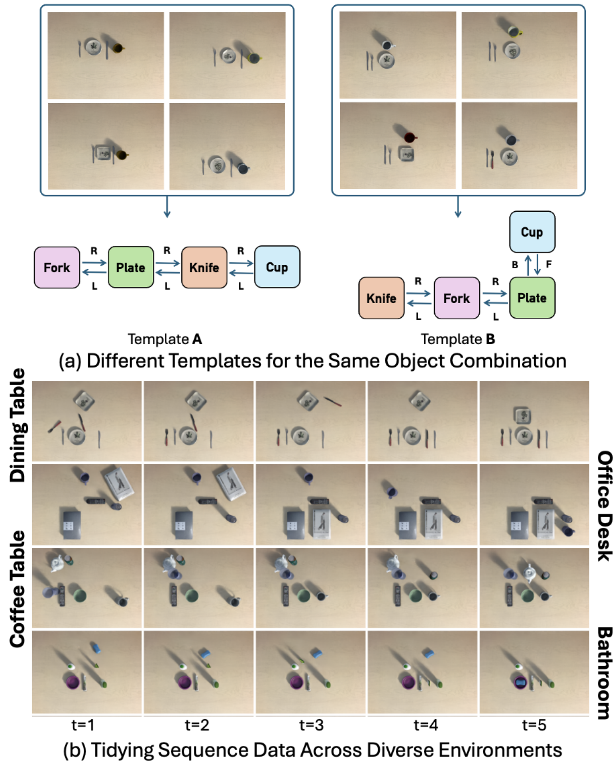

Fig. 2. (a) Arrangements where the fork is to the left of the plate and the knife to the right could all be considered as belonging to template $\mathcal{A}$. Meanwhile, arrangements where both the fork and knife are neatly placed on the left side of the plate would fall under template $\mathcal{B}$. R represents 'right', L stands for 'left', B for 'behind', and F denotes 'front'. (b) The TTU dataset consists of state-action sequences for each environment, ranging from a messy scene (t=1) to a perfectly tidied scene (t=T).

and modifying the central position of the arrangements. This tidied scene becomes the final state s_T of the trajectory, where s_t represents the scene at timestep t, and T denotes the trajectory length. We start from s_T and randomly pick objects to move to random positions on the desk, collecting the untidying sequence of $(s_{T-1}, s_{T-2}, ..., s_1)$. By reversing this sequence, we obtain a tidying sequence from a messy to a tidied table. Finally, we collect a dataset $D = \{\tau_1, ..., \tau_{N_{traj}}\}$, where each trajectory $\tau_i = ((s_1, \psi_1), ..., (s_T, \psi_T))$ consists of a sequence of state and tidiness score pairs. The tidiness scores are given as follows, proportional to the timestep t, with the final state s_T receiving a tidiness score of 1:

$$\psi_t = \frac{t-1}{T-1} \tag{2}$$

As shown in Figure 2-b, we collect tidying sequences in four environments with $T = 5$. Table I lists the numbers of objects, templates, trajectories, and scene data in each environment. There are overlapping objects across the environments, and we utilize a total of 170 objects. In total, we have collected 413 templates and 224,225 scene data including RGB and depth images and object information.

V. Proposed Method

The proposed framework, the Tidiness Score-guided Monte Carlo Tree Search (TSMCTS), consists of two components: (1) training the tidiness discriminator and tidying policy, and (2) planning the tidying up process using MCTS.

TABLE I
Data Collection Across Various Environments

Environment	# Objects	# Templates	# Trajectories	# Data
Coffee Table	93	120	14,880	74,400
Dining Table	105	125	13,245	66,225
Office Desk	43	131	12,865	64,325
Bathroom	29	37	3,855	19,275
Total	170	413	44,845	224,225

We trained a tidiness discriminator and tidying policy using the TTU dataset described in the previous section. The tidiness discriminator learns a score function that evaluates the tidiness score of the current state. The tidying policy is used as a tree policy in the MCTS algorithm to efficiently sample appropriate actions from the entire feasible action space. We trained the tidiness discriminator in a supervised manner and the tidying policy using IQL. Finally, starting from the initial configuration (O, P), we iteratively find actions by planning with MCTS using the tidiness discriminator as a utility function and the tidying policy as a tree policy, until all the objects on the table are tidied up.

A. Tidiness Discriminator and Tidying Policy

The training process of the tidiness discriminator and the tidying policy is illustrated in Figure 3-a. We parameterized the tidiness score function using neural networks $\Psi_\theta : S \mapsto [0, 1]$ and the tidying policy $\pi_\rho : S \mapsto A$, where S and A represent the state space and action space, respectively. In this paper, the state is represented as an RGB image of the table, while the action corresponds to a pick-and-place operation, defined by the target object, placement position, and rotation angle.

From the TTU dataset, we can obtain sequences of state and tidiness score pairs $((s_1, \psi_1), ..., (s_T, \psi_T))$, where T denotes the length of collected trajectories. Here, s_1 is the most messy scene and s_T is the final tidied up scene. Finally, we train the discriminator using pairs of states and score labels, $\mathcal{D}_{Disc} = \{(s, \psi)_i\}$, employing the mean squared error as the loss function:

$$L(\theta) = \mathbb{E}\left[(\Psi_\theta(s_t) - \psi_t)^2\right]. \tag{3}$$

For policy training, we use the Implicit Q-Learning (IQL) method. We use a sparse binary reward as below:

$$r_t = \begin{cases} 1, & \text{if the episode ends } (t = T) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and obtain offline data $\mathcal{D}_{RL} = \{(s_t, a_t, s_{t+1}, r_t)\}$ from TTU dataset. In IQL method, we learns Q-function Q_ϕ, value function V_φ and the policy π_ρ simultaneously. The loss functions for V_φ and Q_ϕ are computed according to the modified TD learning procedure in IQL,

$$L_V(\varphi) = \mathbb{E}_{s,a}\left[L_2^\tau(Q_{\hat{\phi}}(s, a) - V_\varphi(s))\right], \tag{5}$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u < 0)|u^2$ represents the expectile regression loss, with $\tau = 0.7$ used as the default value. $Q_{\hat{\phi}}$ is a target network, which is a lagged version of Q_ϕ.

$$L_Q(\phi) = \mathbb{E}_{s,a,s',r}\left[(r + \gamma V_\varphi(s') - Q_\phi(s, a))^2\right]. \tag{6}$$

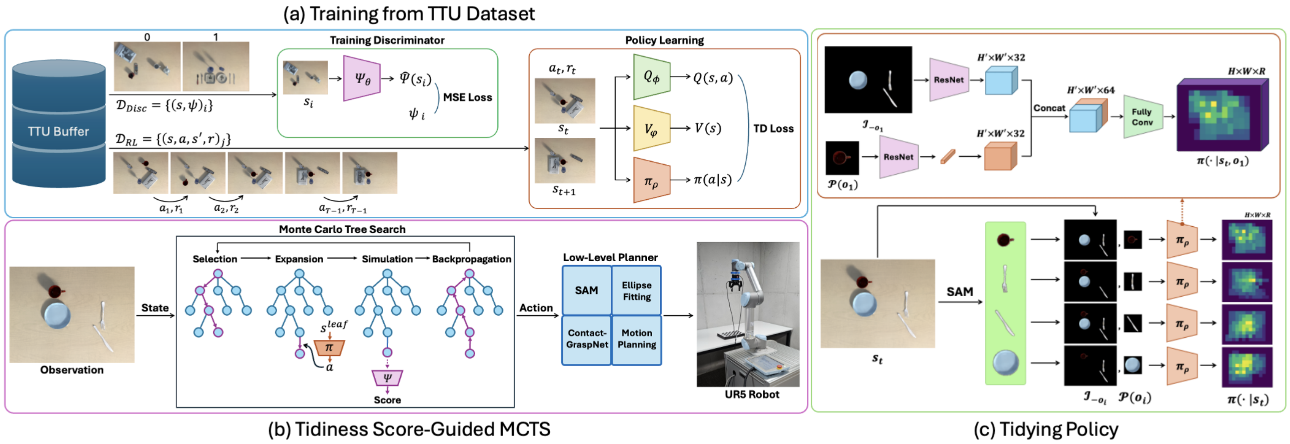

Fig. 3. (a) We train the tidiness discriminator and tidying policy using the TTU dataset. The tidiness discriminator is trained in a supervised manner to predict the tidiness score of the table, while the tidying policy is trained to estimate the action distribution using the IQL framework. (b) During inference, MCTS utilizes the tidiness discriminator Ψ_θ and the tidying policy π_ρ to find the best actions. (c) From the current table image s_t, the policy networks take the table image $\mathcal{I}_{-o_i}$ and the object's patch $\mathcal{P}(o_i)$ as inputs to generate an action probability distribution. The action is defined by the selected object, its placement position, and its rotation.

Then, the policy extraction step can be applied using advantage weighted regression:

$$L_\pi(\rho) = \mathbb{E}_{s,a}\left[\exp(\beta(Q_{\hat{\phi}}(s,a) - V_\varphi(s)))\log \pi_\rho(a|s)\right], \quad (7)$$

where β denotes the inverse temperature.

We employ a pre-trained ResNet-18 as the backbone of the tidiness discriminator, replacing its final fully connected layer with one that predicts a single tidiness score. The tidiness discriminator takes the current table image s_t as input and outputs the corresponding tidiness score $\Psi_\theta(s_t)$. We use the Segment Anything Model (SAM) [19] to remove the background to learn a more consistent score function.

For the tidying policy, two inputs are used for each object o_i, $i = 1, \ldots, N$: (1) the patch image of the object, $\mathcal{P}(o_i)$, and (2) the table image without the object, $\mathcal{I}_{-o_i}$. The policy outputs a probability distribution over pixel positions and rotations for placing the target object. As shown in Figure 3-c, the policy networks extract separate features for the table $\mathcal{F}(\mathcal{I}_{-o_i})$ and the object $\mathcal{F}(\mathcal{P}(o_i))$ using ResNet-18 networks. The 32-dimensional object feature $\mathcal{F}(\mathcal{P}(o_i))$ is then expanded to match the size of the table feature and concatenated with it to form the combined feature, $cat(\mathcal{F}(\mathcal{I}_{-o_i}), \mathcal{F}(\mathcal{P}(o_i)))$. This combined feature is processed through fully convolutional networks to generate an $H \times W \times R$ probability distribution, where R denotes the number of possible rotations. The tidying policy repeats this process for all N objects in parallel to produce an $N \times H \times W \times R$ action probability distribution.

B. Low-Level Planner

When rearranging objects, even if they are placed in the same locations, their arrangement can appear either tidy or disordered depending on their orientations. We observe that objects appear more organized to humans when aligned with the table's x-axis or y-axis. To encourage this, we fit an ellipse to each object's segmentation mask and use its major axis as the default rotation axis.

We discretize the pick-and-place action by dividing the workspace into an $H \times W$ grid and the 360° rotation into R bins. As shown in Figure 4, an action is defined as $a = (o, p)$, where o denotes the target object and $p = (x, y, r)$ represents the placement position. Here, x and y are the pixel coordinates, and r is the rotation index. The default rotation is defined as the orientation where the object's major axis aligns with the table's x-axis.

When the high-level planner outputs a pick-and-place action, the low-level planner generates the corresponding robot trajectory. The agent first receives a top-down RGB image of the table and uses SAM to extract object segmentation masks. Then, we use the least squares method to fit an ellipse to each object and determine its major and minor axes. We use Contact-GraspNet [20] to predict 6-DoF grasping points of the target object. Finally, robot trajectories for grasping and placement are generated using inverse kinematics and the MoveIt planner.

C. Tidiness Score-Guided High-Level Planner

We utilize Monte Carlo Tree Search (MCTS) as a high-level planner. MCTS is a search algorithm to solve decision-making processes for deterministic problems. For the implementation of MCTS, it is necessary to recognize the dynamics from a state s_t to the next state s_{t+1} after an action a_t is performed. Instead of using a separate simulator to get the predicted next state $\hat{s}_{t+1}$, our method generates $\hat{s}_{t+1}$ by directly moving each object's image patch on the initial RGB image. Although $\hat{s}_{t+1}$ may differ from the state achieved by physically performing a pick-and-place action, the approach achieves an 85% success rate in real world experiments, indicating its robustness despite potential errors. Figure 5 shows the process of next state prediction and the sequence of expected and actual states during real world evaluation.

Our high-level policy leverages the trained tidiness discriminator and tidying policy to guide MCTS in finding the most efficient action sequence for tabletop tidying up. The overall inference process of TSMCTS is shown in Figure 3-b. For each timestep t, the agent receives the state which consists of the current RGB-D images. Then TSMCTS builds a search tree with the root node representing the current state s_t. Starting from the initial tree, TSMCTS repeats the following four

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

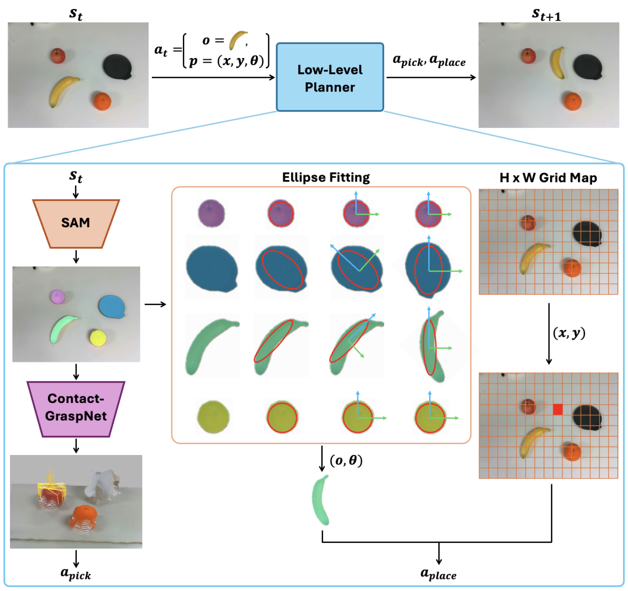

Fig. 4. Given a high-level action specifying which object to pick and where to place, the low-level planner uses the Contact-GraspNet to find a stable grasping point for the object. To place the object in the desired orientation, the initial orientation is determined by applying ellipse fitting to the object mask obtained through SAM, followed by calculating the rotation transformation to determine the placement.

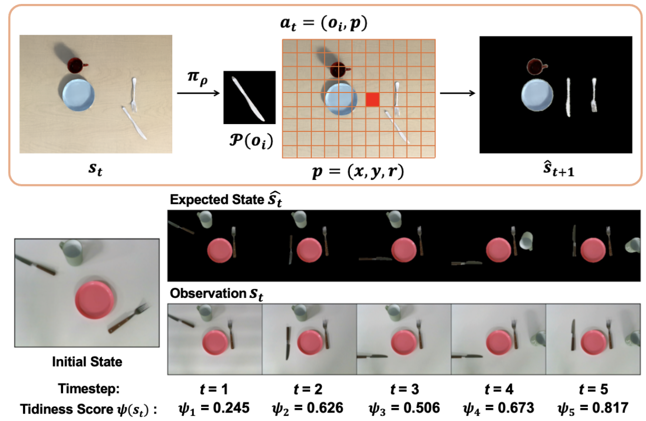

Fig. 5. The upper figure illustrates the process of next state prediction by moving object patches. The lower figure depicts a sequence of TSMCTS evaluations in the real world. The top row presents the predicted states $\hat{s}_t$ by moving image patches from the previous states. The bottom row displays the observed states s_t. ψ_t denotes the tidiness score of each state s_t.

steps K times to complete the tree: Selection, Expansion, Simulation, and Backpropagation.

Selection: Starting from the root node, TSMCTS selects child nodes until it reaches a leaf node. At each node s, TSMCTS selects an action a based on the UCT function, given as follows:

$$U(s, a) = \frac{Q(s)}{N(s, a)} + c\sqrt{\frac{2 \log N(s)}{N(s, a)}}, \tag{8}$$

where c is the exploration term. $N(s)$ denotes the number of visits to node s, and $N(s, a)$ denotes the number of times action a has been executed at node s. $Q(s)$ denotes the cumulative reward of node s, where the reward is assigned during the Backpropagation step.

Expansion: If TSMCTS reaches a leaf node s_{leaf}, it adds a child node to the tree. To expand the tree at the leaf node, we use the trained tidying policy π_ρ to sample actions from the action space, $a \sim \pi_\rho(\cdot|s_{leaf})$, where $a = (o, p)$. As mentioned above, we create the new child node s_{new} by moving the image patch of object o to the position p in the RGB image of s_{leaf}.

Simulation: If an overlap occurs between the moved object and remaining objects during the expansion step, it is treated as a collision and the corresponding node is marked as a failure with a node value of 0. If no collision is detected, we leverage the trained tidiness discriminator to predict the expected value of the expanded node s_{new} as $V(s_{new}) = \Psi_\theta(s_{new})$, where Ψ_θ denotes the tidiness discriminator. We then obtain the outcome $z(s_{new})$ from a random rollout by executing π_ρ until the terminal step $T_{rollout}$. The outcome $z(s_{new})$ is set to 1 if the final state of the rollout is fully tidied up and 0 otherwise.

Backpropagation: TSMCTS backpropagates Q-value updates from the newly expanded node s_{new} back to the root node. For each node s and action a along the path, the Q-value updates are performed as follows:

$$\begin{aligned} &N(s) \leftarrow N(s) + 1, \\ &N(s, a) \leftarrow N(s, a) + 1, \\ &Q(s, a) \leftarrow Q(s, a) + (1 - \lambda)V(s_{new}) + \lambda z(s_{new}). \end{aligned} \tag{9}$$

We use $\lambda = 0.3$, where λ denotes the mixing parameter.

After the tree search is completed, TSMCTS selects the most visited child node of the root node as the best action. Then, the high-level action is converted into robot trajectories by the low-level planner to control the robot.

VI. EXPERIMENTS

A. Evaluation of Tidiness Discriminator

We evaluate whether the trained tidiness discriminator generalizes well to unseen objects and unseen configurations beyond the training data. We divide the 224,225 tidying data into 162,000 training data and 62,225 validation data. The validation data contains unseen objects and templates from the training data. To determine whether a scene is fully tidied up, we define a tidiness threshold ξ, ranging from 0 to 1. During the experiments, a task is considered successful if the tidiness score exceeds ξ. The tidiness threshold is crucial for determining success - lowering ξ increases recall by classifying more scenes as well-tidied but reduces precision due to more false positives.

To determine an appropriate threshold, we analyze the classification performance of the tidiness discriminator across varying tidiness threshold values. The recall and precision measured on the validation set as functions of the tidiness threshold are illustrated in Figure 6. Additionally, we conduct a human evaluation to ensure that the tidiness threshold aligns with human perceptions of tidiness. We present 20 randomly selected sequences from 50 tidying sequences organized by TSMCTS to 17 participants, asking them to choose the scenes they judge to be tidied up. For each sequence, we define the tidiness threshold as the lowest tidiness score among the scenes that participants judge to be tidied up. The average thresholds for each environment are presented in Table III. As a result, people judge that the table is tidied up at an

TABLE II
COMPARATIVE EVALUATION OF PLANNING METHODS IN SIMULATION EXPERIMENTS

	Coffee Table			Dining Table			Office Table			Bathroom			Mixed			Average		
	SR	Score	Length	SR	Score	Length	SR	Score	Length	SR	Score	Length	SR	Score	Length	SR	Score	Length
Direct-BC	10.0	0.571	13.167	2.7	0.402	8.445	6.7	0.496	11.067	16.7	0.576	10.189	2.7	0.452	9.000	7.8	0.499	10.374
Direct-IQL	9.3	0.534	6.417	2.0	0.380	8.000	12.0	0.574	9.764	10.7	0.534	9.056	7.3	0.503	9.583	8.3	0.505	8.564
Greedy Search	68.7	0.887	5.985	71.5	0.896	5.561	80.3	0.902	6.012	78.7	0.883	6.330	72.7	0.905	5.082	74.4	0.895	5.794
TSMCTS-Uniform	73.3	0.874	5.118	82.7	0.878	5.572	88.0	0.890	5.151	89.0	0.900	4.616	84.0	0.895	4.905	83.4	0.887	5.072
TSMCTS-BC	78.0	0.885	5.040	87.3	0.902	5.346	90.0	0.899	**4.555**	86.7	0.897	**4.467**	**92.0**	**0.907**	4.920	86.8	0.898	4.866
TSMCTS	**79.3**	**0.889**	**4.631**	**92.7**	**0.904**	**5.144**	**90.7**	**0.905**	4.778	**89.3**	**0.902**	4.626	90.7	**0.907**	**4.620**	**88.5**	**0.901**	**4.760**

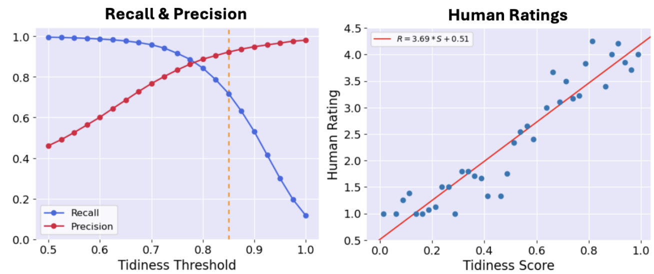

Fig. 6. The left graph shows the recall and precision measured according to the tidiness threshold. The orange dashed line represents the tidiness threshold used in the experiments, $\xi = 0.85$. The right graph shows the distribution of human evaluated ratings according to the tidiness scores. We divide the tidiness score range from 0 to 1 into 40 intervals and average the ratings of scenes within each interval.

TABLE III
TIDINESS THRESHOLD MEASURED BY HUMAN EVALUATION

Environment	Tidiness Threshold
Coffee Table	0.8223 ± 0.1230
Dining Table	0.9017 ± 0.0895
Office Desk	0.8310 ± 0.0892
Bathroom	0.8632 ± 0.0799
Average	0.8486 ± 0.0423

average tidiness score of 0.8486. Looking at the environment, the threshold for the Dining table is higher at 0.9017 compared to other environments. This appears to be because people consider the arrangement more organized when the tableware and cutlery are placed according to their functional uses. Based on these results, we determine that a threshold of 0.85 is an appropriate value. The trained tidiness discriminator achieve a recall of 71.8% and a precision of 92.2% on the validation data using this threshold.

We conduct another human evaluation to verify how well the trained tidiness score reflects the actual perception of tidiness by humans. We sample 10 scene data for each tidiness score interval from 0 to 1 at 0.1 intervals, resulting in a total of 100 scenes. Then, we ask 17 participants to view 15 randomly selected scenes from the 100 scenes and rate the degree of tidiness on a scale of 1 to 5. The correlation between the tidiness score and human ratings is shown in Figure 6. We observe a strong positive correlation between the tidiness score and human ratings. We also find a tendency for the variance in human ratings to increase as the tidiness score rises. This is likely because the standards for tidiness are highly subjective and vary from person to person.

B. Simulation Experiments

We use the PyBullet simulator for the simulation experiments. In the simulator, a workspace table and a UR5 robot are set up. As the initial states, random objects are spawned on the table in random positions and orientations. We use 3D object models from the YCB and HouseCat6D datasets, along with 10 additional object models and four extra object categories not included in the training set of the TTU dataset.

We conduct simulation experiments across five environments by adding a mixed table environment, Mixed, to the original four: Coffee table, Dining table, Office desk, and Bathroom. For each environment, we tested 150 scenarios with varying object compositions and initial placements. We use a tidiness threshold of 0.85, and a scenario is considered successful if a tidied scene is achieved within 10 steps without any collisions. We measured the success rate, the tidiness score of the final state, and the number of steps.

We compare several methods that leverage the tidying policy and the tidiness discriminator:

Direct execution of the learned policy: Direct-BC and Direct-IQL execute the learned policy directly without the MCTS process. Direct-BC uses a policy trained via behavior cloning, while Direct-IQL uses a policy trained via IQL.

Greedy search using the tidiness discriminator: Greedy Search evaluates the tidiness score for each predicted state resulting from all possible actions at the current state and selects the action with the highest score.

TSMCTS with different tree policies: We compare the MCTS process using different tree policies. TSMCTS-Uniform employs a uniform policy, while TSMCTS-BC uses a policy trained via behavior cloning. The proposed method, TSMCTS, uses a tree policy trained with the IQL framework.

The experimental results in Table II showed that direct policy execution often falls into local optima or short-step loops, while Greedy Search struggled in complex scenarios with many objects due to long trajectories. Among TSMCTS variants, the IQL-based tree policy delivered the highest performance, and behavior cloning outperformed the uniform policy, suggesting that stronger tree policies curb unnecessary exploration. To compare performance across environments, TSMCTS showed the lowest success rate and tidiness score in the Coffee table environment, and the highest success rate in the Dining table environment. This is likely because the Coffee table contains a more diverse and complex set of objects, whereas the Dining table consists of more standardized object templates. Notably, it also achieved high success rate and tidiness score on mixed-object configurations unseen during training, demonstrating robust performance across diverse scenarios.

For baseline comparisons, we evaluate TSMCTS comparing StructFormer [4] and StructDiffusion [5]. Both StructFormer and StructDiffusion are algorithms that find arrangements

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

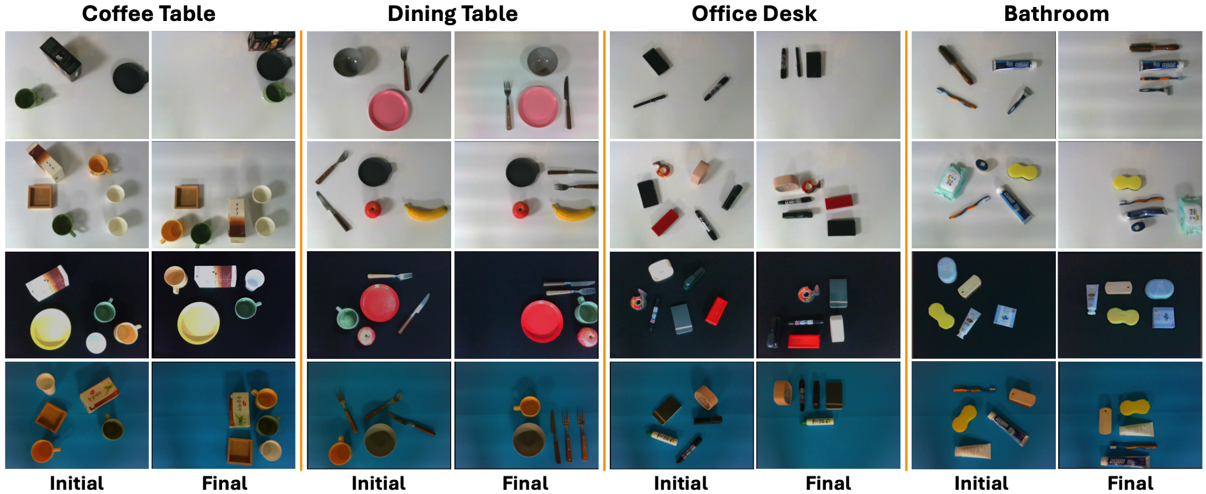

Fig. 7. Examples of tidying up across various object sets. Starting from the initial configurations, TSMCTS successfully tidied up the tables. The demonstration includes various objects such as apples, bananas, clocks, razors, and scotch tape, which are not included in the TTU dataset.

TABLE IV
HUMAN EVALUATION IN SIMULATION EXPERIMENTS

Methods	Distance ↓	Rotation ↓	Number of Operations	NASA-TLX ↓
StructFormer [4]	88.2cm	150.3°	143.1	37.84
StructDiffusion [5]	65.3cm	**101.9°**	116.5	41.86
TSMCTS-Binary	60.3cm	201.0°	104.0	**26.47**
TSMCTS	**57.2cm**	158.1°	**102.9**	27.06

TABLE V
TSMCTS EVALUATION IN THE REAL WORLD

Environment	Success Rate ↑	Tidiness Score ↑	Length ↓
Coffee Table	100%	0.894	4.4
Dining Table	80%	0.840	3.8
Office Desk	80%	0.945	7.0
Bathroom	80%	0.909	5.6
Average	85%	0.897	5.1

matching given conditions, based on language tokens related to goals. While their setup is different from ours, both studies include tasks for organizing a dining table, so we conduct comparative experiments exclusively in the Dining table setting. Additionally, we perform an ablation study on the tidiness discriminator by comparing TSMCTS with TSMCTS-binary. TSMCTS-binary utilizes a tidiness discriminator trained with binary labels from the TTU dataset, where completely tidied scenes are labeled as 1, and all other scenes are labeled as 0.

We conduct a human evaluation to assess how closely each algorithm's results approximate human-perceived tidiness. We ask 17 participants to rearrange 30 randomly selected scenes until they achieve a tidiness. For this purpose, we collect 160 scenes in total, including 20 final outputs and intermediate steps from each algorithm. Participants are allowed to move object patches using keyboard controls, with 1cm translation and 10° rotation per input. We measure the cumulative movement distance, rotation angle, and number of keyboard operations for each scene. Additionally, the participants completed a NASA task load index (NASA-TLX) [21] for each scene. While the NASA-TLX measures workload on six subjective subscales, we report only the three most relevant to our study: mental demand, own performance, and frustration level. The results of human evaluation are presented in Table IV. TSMCTS achieved the shortest movement distance (57.2cm) and fewest keyboard operations (102.9), while TSMCTS-Binary scored lowest on NASA-TLX (26.47). StructDiffusion and StructFormer produced smaller rotation angles, suggesting point-cloud methods better orient objects. Overall, low NASA-TLX scores for both TSMCTS variants indicate that participants feel less task load when tidying the table, implying less effort or stress is required to achieve a tidied arrangement that meets human standards. The minimal movement distance for TSMCTS suggest it positions objects closest to what people consider a tidy arrangement.

C. Real Robot Experiments

We use a Universal Robots UR5 mounted with a Robotiq 2F-85 Gripper at the end effector for real robot experiments. An Intel RealSense D435 camera is mounted on the wrist of UR5 to capture RGB-D images in 480×640 resolution. At each timestep, our agent receives RGB and depth images from the mounted camera at the fixed view point.

We evaluate TSMCTS across four environments: Coffee table, Dining table, Office desk, and Bathroom. TSMCTS considers a scene to be tidied if the tidiness score exceeds the tidiness threshold. A scenario is considered successful if a tidied scene is reached within 10 steps without any collisions. We conduct tidying scenarios with five different object configurations in each environment and measure the tidiness score of the final scene for each scenario. The results of the experiments are presented in Table V, and demonstrations of the final tidied up tables are shown in Figure 7. TSMCTS achieves an average tidiness score of 0.897 and a success rate of 85% across a total of 20 scenarios. In real experimental scenarios, objects not shown in the TTU dataset are also included in the configurations. These results demonstrate that TSMCTS can robustly tidy up even in scenarios with diverse and complex object compositions.

For baseline comparisons, we evaluate TSMCTS comparing StructFormer and StructDiffusion. We conduct comparative experiments solely in the Dining table setting, similar to the simulation experiment. StructFormer and StructDiffusion

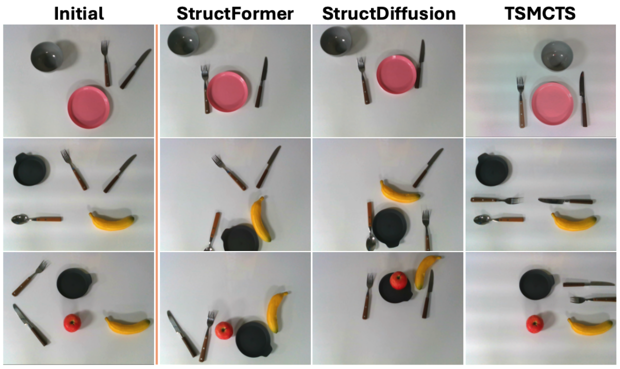

Fig. 8. Examples of tidying up using various methods in the real world. We evaluated StructFormer, StructDiffusion, and TSMCTS on a Dining table setup, starting from same initial configurations.

TABLE VI
REAL WORLD EVALUATION ON THE DINING TABLE ENVIRONMENT

Methods	Success Rate ↑	Length ↓	Collisions ↓
StructFormer [4]	40%	3.6	0.6
StructDiffusion [5]	60%	**3.0**	0.4
TSMCTS	**80%**	3.4	**0.2**

consider a scene tidied if the objects are placed at the intended target positions. In all three methods, a scenario is considered successful if a tidied scene is achieved without any collisions. We evaluate with five different scenarios, and the results are presented in Table VI and Figure 8. In a simple dining table setup with one plate, fork, and knife, StructFormer and StructDiffusion demonstrate good tidying performance. However, in complex configurations with multiple forks or knives, StructFormer and StructDiffusion often fail to find appropriate positions for all objects, leading to overlaps and collisions. TSMCTS, on the other hand, is able to find tidied arrangements even with complex configurations, demonstrating its diverse and robust tidying capabilities.

VII. CONCLUSION

In this paper, we have introduced the TSMCTS framework, a tidiness score-guided Monte Carlo tree search for tabletop tidying up. TSMCTS is a framework that uses a tidiness discriminator to assess current and future table tidying states, generates a search tree according to the tidying policy, and finds the optimal 3 for tidying up. To train the tidiness discriminator and tidying policy, we have collected the TTU dataset, a structured dataset that includes tidying sequence data across various environments. We have shown experimental results that TSMCTS has robust tidying capabilities across various object configurations including unseen objects and duplicate objects. In addition, we have successfully transferred TSMCTS to the real world without any transferring efforts. Despite the satisfactory results, there also exist limitations in the proposed method. Since TSMCTS assumes a 2D arrangement, it cannot perform tidying that involves stacking objects in layers. Additionally, the tidiness discriminator relies on visual information, which often leads to a lack of consideration for the functional uses of objects. In future work, we plan to leverage large language models (LLMs) as guidance to better handle ambiguous cases and resolve scenarios where the functional use or arrangement of objects is unclear.

REFERENCES

[1] E. Huang, Z. Jia, and M. T. Mason, "Large-scale multi-object rearrangement," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[2] A. H. Qureshi, A. Mousavian, C. Paxton, M. C. Yip, and D. Fox, "Nerp: Neural rearrangement planning for unknown objects," in *Robotics: Science and Systems (RSS)*, 2021.

[3] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "Ifor: Iterative flow minimization for robotic object rearrangement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 787–14 797.

[4] W. Liu, C. Paxton, T. Hermans, and D. Fox, "Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[5] W. Liu, Y. Du, T. Hermans, S. Chernova, and C. Paxton, "Structdiffusion: Language-guided creation of physically-valid structures using unseen objects," in *Robotics: Science and Systems (RSS)*, 2023.

[6] H. Chang, K. Gao, K. Boyalakuntla, A. Lee, B. Huang, J. Yu, and A. Boularias, "Lgmcts: Language-guided monte-carlo tree search for executable semantic object rearrangement," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[7] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. M. J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021.

[8] J. Li, D. Li, C. Xiong, and S. Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*. PMLR, 2022.

[9] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning (CoRL)*, 2022.

[10] T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson, "Robotic skill acquisition via instruction augmentation with vision-language models," *arXiv preprint arXiv:2211.11736*, 2022.

[11] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[12] Kostrikov, Ilya, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[13] I. Kapelyukh and E. Johns, "My house, my rules: Learning tidying preferences with graph neural networks," in *Conference on Robot Learning (CoRL)*. PMLR, 2022.

[14] N. Abdo, C. Stachniss, L. Spinello, and W. Burgard, "Robot, organize my shelves! tidying up objects by predicting user preferences," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[15] I. Kapelyukh and E. Johns, "Scenescore: Learning a cost function for object arrangement," *arXiv preprint arXiv:2311.08530*, 2023.

[16] M. Wu, F. Zhong, Y. Xia, and H. Dong, "Targf: Learning target gradient field to rearrange objects without explicit goal specification," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 986–31 999, 2022.

[17] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[18] H. Jung, S.-C. Wu, P. Ruhkamp, G. Zhai, H. Schieber, G. Rizzoli, P. Wang, H. Zhao, L. Garattoni, S. Meier, D. Roth, N. Navab, and B. Busam, "Housecat6d-a large-scale multi-modal category level 6d object perception dataset with household objects in realistic scenarios," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 22 498–22 508.

[19] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[20] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[21] L. Colligan, H. W. Potts, C. T. Finn, and R. A. Sinkin, "Cognitive workload changes for nurses transitioning from a legacy system with paper documentation to a commercial electronic health record," *International journal of medical informatics*, vol. 84, no. 7, pp. 469–476, 2015.