MANER: Multi-agent Neural Rearrangement Planning of Objects in Cluttered Environments

Vivek Gupta, Praphpreet Dhir, Jeegn Dani, and Ahmed H. Qureshi

***Abstract*—Object rearrangement is a fundamental problem in robotics with various practical applications ranging from managing warehouses to cleaning and organizing home kitchens. While existing research has primarily focused on single-agent solutions, real-world scenarios often require multiple robots to work together on rearrangement tasks. This paper proposes a comprehensive learning-based framework for multi-agent object rearrangement planning, addressing the challenges of task sequencing and path planning in complex environments. The proposed method iteratively selects objects, determines their relocation regions, and pairs them with available robots under kinematic feasibility and task reachability for execution to achieve the target arrangement. Our experiments on a diverse range of simulated and real-world environments demonstrate the effectiveness and robustness of the proposed framework. Furthermore, results indicate improved performance in terms of traversal time and success rate compared to baseline approaches. The videos and supplementary material are available at https://sites.google.com/view/maner-supplementary**

***Index Terms*—Task and Motion Planning, Multi-Robot Systems, Deep Learning Methods**

I. INTRODUCTION

OBJECT rearrangement is a fundamental problem in the field of robotics, with diverse applications in warehouse automation, robotic kitchens, and precision manufacturing. The ability to rearrange objects in a desired configuration is crucial for improving efficiency, organization, and productivity in these domains. The problem involves three stages: task planning, path planning, and execution. During the first phase, the robot decomposes the task into sub-tasks while considering the environment state, robot kinematic reachability, and task feasibility. The remaining phases find the robot's motion path and control actions to achieve the given subtasks in the environment. While most of the existing work addresses the rearrangement planning with a robot arm in tabletop environments, the solutions via multi-agent mobile manipulation systems remain largely unexplored.

In multi-agent systems, task planning plays a crucial role in achieving a desired global objective by assigning optimal sequences of sub-goals to each agent. It involves solving a combinatorial optimization problem, which determines the most efficient order of tasks for agents to accomplish. This stage is a vital precursor to path planning and execution. Existing research has proposed solvers to address some of these planning problems [1] [2]. However, these approaches do not consider noisy observations, rely on simplified environments, and assume objects have trackable markers and readily reachable goals, i.e., monotone scenarios. In contrast, the practical scenarios often lack objects with trackable markers and contain non-monotone cases, where objects require multiple relocations before making it to the goal, adding complexity to the problem of rearrangement planning and making vision-based techniques necessary.

In this paper, we present **M**ulti-**A**gent **Ne**ural **R**earrangement (MANER) framework for long-horizon multi-agent object rearrangement planning from visual observations in cluttered environments. The framework analyzes the current and target state of the environment to determine potential pickup objects for each agent. Additionally, it proposes feasible drop-off locations for each agent-object pair. Subsequently, a path planning framework evaluates these pick-place propositions to select the best pick-place sequence, enabling efficient scene arrangement by multiple agents. The key contributions of this paper are:

- a pick and place module to compute the feasible sub-task sequences for each agent, considering robot traversal time, reachability, and collision avoidance.
- a scalable rearrangement planning framework that accommodates the varying number of objects and agents in the scene and tackles both monotone and non-monotone cases.
- a training and testing strategy that enables one-shot sim2real transfer of our neural policies to solve multi-agent rearrangement planning problems in real-world scenarios, relying only on the bird's eye view of the environment.

To the best of our knowledge, MANER is the first learning-based approach that specifically tackles the task of multi-agent object rearrangement planning, accounting for complex rearrangement scenarios where objects may require multiple repositioning steps. Our experimental results demonstrate the effectiveness and superior performance of MANER compared to classical baseline methods and its ability to generalize to real-world scenarios.

II. RELATED WORK

A. Rearrangement Planning

Rearrangement planning has garnered significant attention from researchers due to its applicability in various domains [3]. Tree search- and learning-based approaches have been widely explored to solve this problem. For instance, methods [4], [5] introduce a tree search-based strategy to mitigate the combinatorial complexity in decision-making for rearrangement planning. In a similar vein, Ren et al. [6]

The authors are with the Purdue University, West Lafayette, IN 47907, USA {gupta690, pdhir, jdani, ahqureshi}@purdue.edu

V.Gupta and P.Dhir did the project during their stay at Purdue University.

proposed a rearrangement-based manipulation approach using kinodynamic planning with dynamic horizons. While these tree search-based methods have demonstrated effectiveness to some extent, they exhibit poor asymptotic performance as the environment becomes cluttered with more objects.

In contrast, the learning-based approaches [7] have recently emerged as a scalable tool for addressing the rearrangement planning problem. Qureshi et al. [8] proposed a neural approach that estimates the optimal pick-and-place actions for each task in the rearrangement problem. Their work incorporates reactive object placement strategies and exhibits promising results. Wu et al. [9] introduced a visual foresight model that leverages deep learning to generalize to real-world environments, enabling zero-shot rearrangement planning. However, all the abovementioned methods, including classical and learning-based, have predominantly focused on single-agent manipulation systems with minimal kinematic reachability challenges, overlooking the complexities of multi-agent mobile manipulation systems.

B. Multi-Agent Systems

In the field of multi-agent systems, the concept of coordinating multiple agents to achieve distributive and cooperative tasks has a longstanding history, tracing back to the 1980s [10]. Since then, several approaches have been introduced, broadly falling into search, optimization, and learning-based methods. The search-based methods mainly focus on strategies to reduce the search space for multi-robot task allocation [11]. Recently, Motes et al. [12] introduced a hypergraph representation for multi-robot task and motion planning, demonstrating its effectiveness in generic rearrangement tasks. Gao et al. [13] proposed a dependency graph-guided heuristic search procedure tailored to non-monotone rearrangement tasks, albeit considering only two manipulators. Still, despite advancements, the search-based approaches lack scalability to a large number of objects or robots due to their computational cost.

The optimization-based approaches solve the constraint optimization problem for multi-robot task allocation and scheduling and have surfaced as a promising tool for solving various assembly tasks [1], [14]. However, the existing work considers monotone cases with readily reachable goals, assumes complete information about the environmental states, and is yet to be explored in real-world settings with raw visual observations. In the realm of learning-based methods, the problem of rearrangement planning with multi-agent mobile manipulation systems remains relatively unexplored, except for recent work [2]. This approach also considers only monotone cases where the task is to transport objects to a fixed target. Their intention-based representation and deep reinforcement learning framework showed promise in coordinating multiple mobile robots to achieve such tasks. However, their approach assumes the known location of the robots and objects, i.e., it does not operate under noisy visual observations.

In contrast to the abovementioned approaches, our method solves both monotone and non-monotone rearrangement planning problems. In addition, our framework operates directly from raw visual observation and generalizes to real-world settings via direct sim2real transfer. Besides, the tasks considered in this work are significantly more challenging than previous works, requiring planners to consider goal reachability as goals may initially be occupied, feasible space due to static and dynamic obstacles, and robot kinematics.

III. Proposed Method: MANER

Given a bounded workspace $W \subset SE(2)$, containing $m \in \mathbb{N}$ agents and $n \in \mathbb{N}$ movable objects, and a set of static obstacles, our goal is to find a motion plan for each agent that rearranges the environment to achieve the target configuration.

We define our multi-agent rearrangement planning problem as follows: The target scene of the environment is denoted by X_T, and the current scene at time t is denoted by X_t. The states of objects in the current and target scene images are denoted as $o^t_{\{n\}}$ and $o^T_{\{n\}}$ respectively. Here, $o \in o_{\{n\}}$ represents the object state comprising position and instance label. The multi-agent robot states at time t are denoted as $r^t_{\{m\}}$, and $r^t \in r^t_{\{m\}}$ is the three-dimensional pose of an agent at time t. Let $l_{\{b\}}$ comprise all the free regions in the current scene X_t. The multi-agent object rearrangement planning problem involves pairing each available agent with an object from the scene and allocating a free region for its relocation. Each agent picks its paired object and relocates it to the assigned region. This process is repeated until the target arrangement in X_T is achieved. Therefore, at each time step t, the MANER policy, defined as π, takes as input the current and target scene and outputs the action $\tau^t = \{(r^t_0, o^t_0, l^t_0), \cdots, (r^t_i, o^t_j, l^t_k)\}$, where $i \in [0, m]$, $j \in [0, n]$, $k \in [0, b]$, and each element (r^t, o^t, l^t) is best pairing of a robot at state r^t with pick-up object o^t, and its placement location l^t. In other words,

$$\{(r^t_0, o^t_0, l^t_0), \cdots, (r^t_i, o^t_j, l^t_k)\} \longleftarrow \pi(X_t, X_T)$$

We introduce a function σ that represents an underlying multi-agent motion planner. The function σ takes the given pick-and-place action sequence τ^t and determines if there exist collision-free path solutions for each robot r^t to reach its target object o^t and then the relocation region l^t. The function σ returns $\emptyset$ if any of the required paths are not found in the given time limit. Let E be the environment function that executes the given path $\sigma(\tau^t)$ on X_t and returns the next environment scene X_{t+1}. Furthermore, we assume all robots execute their assigned subtasks synchronously and, after completion, wait for the subsequent subtask assignments. Finally, let F indicate the total traversal time for all robots in executing their assigned tasks in τ^t.

Our objective is to find an near-optimal function π that solves the multi-agent rearrangement problem. We formulate our objective as follows:

$$\min_{\tau^t \sim \pi} \sum_{t=0}^{T-1} F(\tau^t), \quad \text{subject to}$$
$$X_T = E(\sigma(\tau^{T-1}), X_{T-1}) \,\&\, \sigma(\tau^t) \neq \emptyset \; \forall t \in [0, T-1]$$

We aim to generate an action trajectory $\{\tau^0, \cdots, \tau^{T-1}\}$ through the function π that minimizes the overall multi-robot travel time given by F. Additionally, the output trajectory should lead to feasible multi-robot motion planning, i.e.,

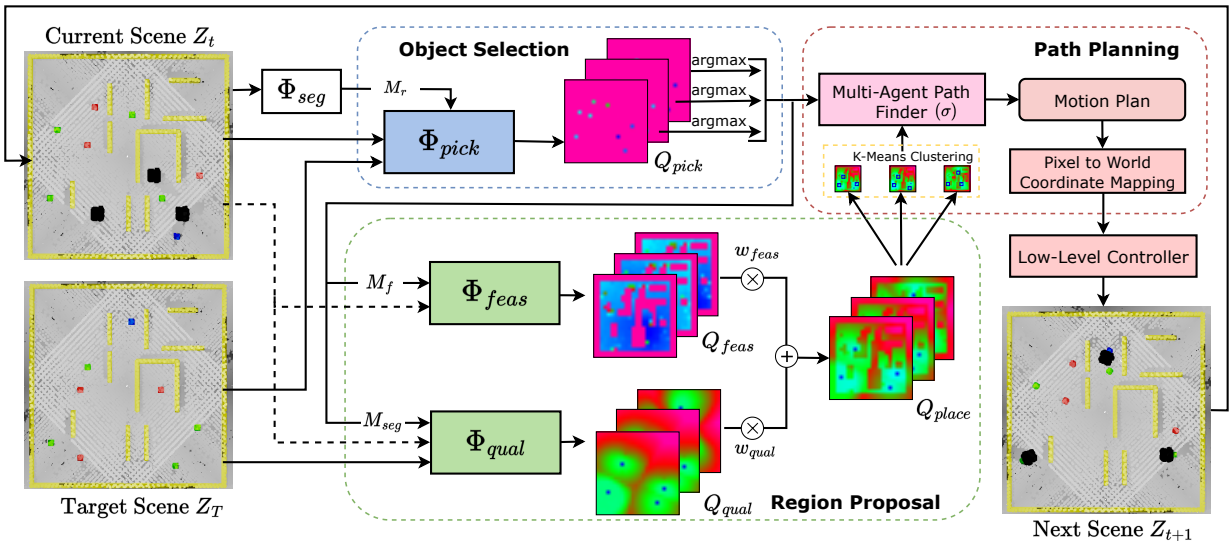

Fig. 1. MANER architecture: Initially, bird's-eye view images of the current and target scenes are captured and processed. These images Z_t and Z_T are then passed through the Object Selection Module, denoted as Φ_{pick}, which generates a heatmap called Q_{pick}. This output is further processed by the region proposal module, which consists of two transformer networks, Φ_{feas} and Φ_{qual} and works together to produce a final heatmap named Q_{place} identifying suitable regions for object placement. The proposed regions are clustered using the k-means algorithm, and then fed into a Multi-Agent Path Planner (σ), which generates a motion plan. Subsequently, the visual coordinates are converted to world coordinates and the motion plan is executed to reach the next state Z_{t+1}. Throughout the planning and execution phases, a low-level controller is utilized to guide the agents and achieve the desired object rearrangement. Note that the darker pixels in the heatmap indicate higher confidence and lighter pixels indicate lower confidence.

$\sigma(\tau^t) \neq \emptyset \; \forall t \in [0, T-1]$, and result in achieving the target scene arrangement X_T. The following sections provide a detailed description of MANER, while Figure. 1 and Algorithm 1 provides an overview of the overall framework.

A. Scene Preprocessing

To capture the entire state X_t of the environment in our environments, we utilize RGBD camera(s). In case of multiple cameras, the captured images are converted to a single point cloud using the intrinsic property K, extrinsic property R_t, and depth information provided by those cameras. We take the bird's eye view image of the reconstructed point cloud to obtain the processed environment state Z_t. Next, we utilize an instance segmentation model Φ_{seg} [15] [16] to extract relevant object states $o^t_{\{n\}}$ from the RGB image Z_t.

B. Neural Models

MANER consists of two neural network modules that enable efficient performance of the object rearrangement task even in cluttered environments. These modules generate the action sequence τ^t that specifies the pick-and-place actions for each agent. These components are described as follows.

1) Object Selection Module: The object selection module is responsible for determining the most suitable object for each agent to relocate. This module takes into account various factors, including the current scene arrangement, target scene arrangement, object states, and the positions of the agents, to identify objects that meet the criteria of being in close proximity to the agents and feasible to be picked.

The module Φ_{pick} takes as input a stacked map $M_{pick} \in \mathbb{R}^{H \times W \times 7}$ which includes the top-down observation of the current scene, denoted as $Z_t \in \mathbb{R}^{H \times W \times 3}$, the target scene $Z_T \in \mathbb{R}^{H \times W \times 3}$, and a map $M_r \in \mathbb{R}^{H \times W \times 1}$ encoding the current state of each agent r in the arena. Using this information, it generates a heatmap $Q_{pick} \in \mathbb{R}^{H_l \times W_l \times m}$, where $H_l = H/p$ and $W_l = W/p$. Each pixel (u, v, r) in Q_{pick} corresponds to a squared patch of size $p \times p$ in the current environment image, representing the confidence level of agent r to successfully transition and pick an object from that location (see Figure 1). The agents are assigned a unique location based on the highest values in Q_{pick}, reflecting the most promising pick-up spots in the scene. To determine the specific object that each agent is prepared to pick, the module utilizes the object states $o^t_{\{n\}}$ extracted from Z_t and the promising pick-up spots for each agent. The final output is a selection of agent-object pairs, i.e., $\tau_{pick} = \{(r^t_0, o^t_0), \cdots, (r^t_i, o^t_j)\}$, where $i \in [0, m]$ and $j \in [0, n]$.

2) Region Proposal Module: This component consists of two interconnected networks: the Feasibility Network and the Quality Network. These networks work together to identify optimal regions for each agent-object pair, taking into account both the feasibility of reaching the region and the importance of the region for object relocation in terms of goal reachability. The networks are discussed below.

a) Feasibility Network: Once an object is selected, the next step involves transferring it to an optimal location that the agent can reach in the shortest possible time. To achieve this, the feasibility network plays a crucial role. The feasibility network Φ_{feas} takes a stacked map $M_{feas} \in \mathbb{R}^{H \times W \times 2}$ as input. This map consists of a binary image of the current scene, denoted as $B_t \in \mathbb{R}^{H \times W \times 1}$, where free locations are represented by 0, and static obstacles and objects in the arena are represented by 1. Additionally, the map includes $M_f \in \mathbb{R}^{H \times W \times 1}$, which specifies the position of the object to be picked by the current agent and that of the other agents

as provided by the Object Selection Module. The location of the current object is denoted by +1, while that of other picked objects is represented by -1.

The feasibility network then produces a heatmap denoted as $Q_{feas} \in \mathbb{R}^{H_l \times W_l \times m}$. This heatmap indicates the feasibility or reachability of different locations on the map for each agent. The feasibility score takes into account factors such as traversal time and location sparsity, aiming to identify regions that can be efficiently accessed while avoiding obstacles and other agents. If an accessible region is densely occupied, the feasibility score is lower compared to regions with fewer nearby objects. This ensures that the agent can transfer the object efficiently to the selected location while navigating around obstacles and other agents.

b) Quality Network: When determining the best regions for transferring an object, we need to consider not only the reachability but also the proximity to possible final locations. For instance, if an object cannot be directly transferred to a final destination and needs to be moved to an intermediate location, regions closer to the final destination should be better suited for it than those farther away. To accomplish this, the quality network Φ_{qual} is employed. The input to this network consists of stacked map $M_{qual} \in \mathbb{R}^{H \times W \times 9}$ consisting of top-down observation of the current scene, denoted as $Z_t \in \mathbb{R}^{H \times W \times 3}$ and target scene $Z_T \in \mathbb{R}^{H \times W \times 3}$, and a segmentation mask $M_{seg} \in \mathbb{R}^{H \times W \times 3}$ of the current scene from the object selection module Q_{pick}. It generates a heatmap $Q_{qual} \in \mathbb{R}^{H_l \times W_l \times m}$, indicating the quality score of each patch for object transfer. This score accounts for the proximity to the target location, aiming to minimize unnecessary movements during the process.

To determine the best transfer regions for each agent-object pair, we employ a weighted sum Q_{place} that combines the outputs from the Feasibility Network and the Quality Network. Q_{place} is computed according to Line 17 in Algorithm 1. The weights $w_f \in (0, 1)$ and $w_q \in (0, 1)$ are used to control the relative importance of feasibility and quality, respectively, and can be adjusted based on the specific task requirements. For our experiments, we set w_f to 0.2 and w_q to 0.8 as a balanced configuration.

Instead of selecting a single region for each agent-object pair, the region proposal module applies techniques such as k-means clustering to the output heatmap Q_{place}, generating a set of k potential transfer regions denoted as $\tau^r_{place} \in l_b$ (Line 23). α is a hyperparameter in the range of $(0, 1)$, used to filter out regions with insignificant values (Line 21). These proposed transfer regions represent possible intermediate or final locations where the agent can successfully transfer the object. In our experiments, we propose three regions for each agent-object pair, i.e., $k = 3$.

C. Path Planning

Once the Object Selection module and Region Proposal module have determined the objects to be picked and the potential transfer regions, the next step is to plan the paths for each agent to execute the rearrangement actions. For our experiments, we employed multi-agent path planning algorithm called AA-SIPP(m) (Anytime Any-angle Search with Safe Interval Path Planning for Multi-Agent Systems) [17]. This algorithm is designed for multi-agent path planning in shared 2D workspaces and can find collision-free paths for multiple agents operating simultaneously.

Algorithm 1: MANER $(X_0,\ X_T)$

1 $Z_T \leftarrow$ ScenePreprocessing(X_T)
2 **for** $t = 0, \ldots, T-1$ **do**
3 $Z_t \leftarrow$ ScenePreprocessing(X_t)
4 $B_t \leftarrow$ BinaryImage(Z_t)
5 $o^t_{\{n\}}, r^t_{\{m\}} \leftarrow$ InstanceSegmentation(Z_t)
6 $\tau_{pick} \leftarrow [], \tau_{place} \leftarrow []$
7 **for** *each agent* $r \in r^t_{\{m\}}$ **do**
8 $M_r \leftarrow$ AgentEncode$(r, r^t_{\{m\}} \backslash r)$
9 $Q_{pick} \leftarrow \Phi_{pick}(Z_t, Z_T, M_r)$
10 $o_{pick}(r) \leftarrow \arg\max_{(u,v)} Q_{pick}(u, v, r)$
11 $\tau_{pick} \leftarrow \tau_{pick} \cup (r, o_{pick}(r))$
12 **for** *each* $(r, o_{pick}) \in \tau_{pick}$ **do**
13 $M_f \leftarrow$ ObjectEncode(o_{pick}, τ_{pick})
14 $M_{seg} \leftarrow$ SegmentationMask(Z_t, o_{pick})
15 $Q_{feas} \leftarrow \Phi_{feas}(B_t, M_f)$
16 $Q_{qual} \leftarrow \Phi_{qual}(Z_t, Z_T, M_{seg})$
17 $Q_{place} \leftarrow w_f * Q_{feas} + w_q * Q_{qual}$
18 $Q^{max}_{place}(r) \leftarrow \arg\max_{(u,v)} Q_{place}(u, v, r)$
19 $S \leftarrow \phi$
20 **for** *each* (u, v) **do**
21 **if** $Q_{place}(u, v, r) > \alpha Q^{max}_{place}(r)$ **then**
22 append (u, v) to S
23 $\tau^r_{place} \leftarrow$ K_Means_Cluster(S)
24 $\tau_{place} \leftarrow \tau_{place} \cup \tau^r_{place}$
25 $\tau \leftarrow$ PathPlanning$(\tau_{pick}, \tau_{place})$
26 $X_{t+1} \leftarrow E(\sigma(\tau), X_t)$

In this path planning phase, we utilize the multi-agent path planner σ to find the near-optimal motion plan for τ that is free from collisions if such a plan exists. Each element in τ is of the form (r, o, l), where r represents the location of an agent, o denotes the assigned object, and l indicates the selected transfer region for the agent-object pair. Given that we have defined multiple possible regions τ^r_{place} for each agent-object pair, our goal is to select the transfer regions $\tau^{max}_{place} \in l_b$ for each agent that results in the shortest collision-free path while maximizing the number of objects successfully relocated to their desired target locations. This process ensures that the action sequences generated by the Object Selection and Region Proposal modules can be executed effectively, enabling the agents to navigate the environment freely without any collisions. The low-level controller then facilitates the execution of the resulting motion plan after pixel-to-world coordinate conversion using the camera's intrinsic, extrinsic, and depth information, leading to the next state.

By incorporating the key components described above, MANER offers a comprehensive solution for multi-agent object rearrangement planning. It effectively addresses the objective of minimizing the overall multi-agent traversal time

while sequencing each agent's tasks, ensuring feasible motion planning, and achieving the target scene arrangement.

IV. Implementation Details

A. Environment Setup

In both simulation and real environment setups, we employ Turtlebot3 Robots, which are built on the ROS standard platform, for object manipulation. To capture the state of the environment at every time step, we utilize RGBD camera(s). For simulation experiments, we use four such synthetic cameras, while for real experiments, we use Intel Realsense D455 camera. Each environment is composed of varying numbers of objects, agents, and static obstacles, each with diverse shapes and sizes. The locations of these objects, agents and obstacles are randomized in each experiment. For both simulation and real-robot experiments, we work with images of size 480x480. We evaluated the effectiveness of our framework by assigning our robots three distinct rearrangement tasks in cluttered environments. These tasks included shuffling objects, sorting objects (including arranging objects in a specific pattern) and random placement of objects. Visual examples of these tasks can be seen in Figure 2. Additionally, we introduce domain randomization in different aspects of the environment to assess the robustness of our framework. Some of these variations are outlined below:

- Size of the arena: 4.5 meters to 5.5 meters.
- Number of agents: 2 and 3.
- Number of objects: 8, 12, and 16.
- Object colors: Red, green, and blue, with Gaussian noise.
- Number of static obstacles: Ranging from 5 to 14.
- Arena color: Gray values ranging from 128 to 255.

The size of agents was fixed at 0.4 meters, while the size of objects was kept fixed at 0.1 meters.

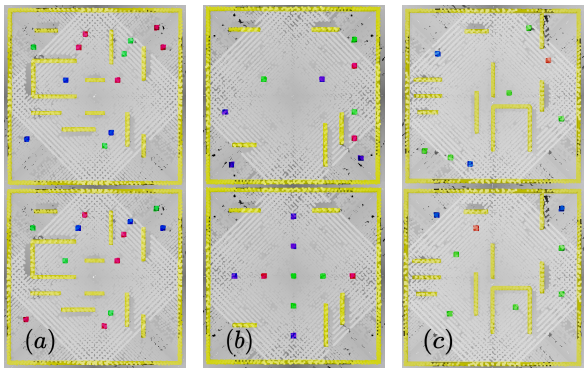

Fig. 2. Start (top) and target (bottom) configuration for different types of rearrangement tasks. These include (a) shuffling objects, (b) sorting objects, and (c) random placement of objects.

B. Data Generation: Training and Testing

We utilized PyBullet module to gather our dataset, comprising 600 distinct environments. Within each environment, we define 20 different configurations, distributed as follows: 40% for shuffling, 30% for sorting, and 30% for random object placement tasks. To obtain accurate ground truth data for our training dataset, we employed specific methodologies. For the object selection module, we utilized the A* algorithm on each agent to compute the shortest path from its location to every object in the environment, detected using instance segmentation. This process generated the ground truth heatmap, which served as a reference for object selection. In order to account for objects that were already positioned at their target locations, we multiplied those locations with a factor of 0.1 in the ground truth heatmap. This adjustment reflected the lower likelihood of these objects being moved in most cases. For the feasibility module, we sampled a set of locations within the workspace. Applying the A* algorithm from potential object pickup locations to the sampled locations allowed us to generate a heatmap. Areas with high density were scaled with a factor of 0.5, considering the lower feasibility for subsequent rearrangement actions. In the quality network module, we adopted a heuristic approach using the Euclidean distance from the closest target location of the pickup object to every location in the workspace. This calculation enabled us to compute the ground truth heatmap, which served as the reference label for the quality network. To increase the size of our dataset and improve network performance, we also applied data augmentation techniques such as vertical and horizontal flips, as well as clockwise and anticlockwise rotations by 90 degrees. From all data samples, we use 80% for training and the remainder for evaluation. Also note that we train our models in the simulation with domain randomization and use our real-world setup only for evaluation purposes to demonstrate the sim-to-real transfer capability of our framework.

C. Training Details

1) Model Architectures: The implementation of our algorithm was carried out using PyTorch. One of the key characteristic of our algorithm lies in the uniformity of the network architectures employed. Each network leverages a transformer network, following the framework outlined in [18]. The foundation of each network is a Fully Convolutional Network (FCN) responsible for encoding the environment input into a latent space. The FCN achieves dimensionality reduction through a sequence of convolutional, ReLU, and MaxPool layers. By applying a sliding window of size $p \times p \times i$ to the input, where i varies depending on the network type (7 for object selection, 2 for feasibility, and 9 for quality), the FCN generates an output of dimensions $H_l \times W_l \times d$. Here, H_l and W_l correspond to the height and width of heatmap, while d represents the latent dimension of the transformer encoder. The value of p is determined by the smallest block that an agent or object can occupy in the image. We set the patch size p to 20 for our experiments. The output of the FCN is subsequently passed through the transformer encoder, which facilitates the learning of connections between local regions in the current and target scenes of the environment. The encoder architecture encompasses a series of multi-headed self-attention (MSA) and multi-layer perceptron (MLP) blocks. To further enhance the model's performance, we employ Dropout, Layer Normalization, and residual connections, following the approaches presented in [18] and [19]. Through its attention

mechanism applied to all patches, the network effectively encodes the significance of individual pixels for the specific task at hand. Lastly, the output of the encoder undergoes a 1×1 convolutional layer, enabling the prediction of probabilities for each patch.

2) Parameters: We use the L2 loss and the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e-9$. We varied the learning rate as proposed in [18] with warm-up steps of 3200. The latent size d of the transformer encoder was set to 512. We apply residual dropout of 0.1 as presented in [18]. Each model was trained for 100 epochs with a batch size of 120. The models were trained on one machine with 3 NVIDIA 3090RTX graphics card.

V. Results

Our evaluation consisted of four sets of experiments. Firstly, we evaluated the performance of our method on various rearrangement scenarios, comparing it against several classical baselines. The number of agents and objects used in these scenarios was consistent with the training data. Secondly, we show the generalization capability of our method by testing it on object rearrangement tasks with unseen numbers of objects and agents. Thirdly, we conducted an ablation study to analyze the individual contributions of each component within the MANER framework. Lastly, we showcased the sim-to-real generalization of our method by applying it to real-world object rearrangement tasks. To quantitatively compare the performance of different methods, we employed the following metrics:

- **Placement Success Rate (SR)**: This metric represents the percentage of successfully placed objects at their desired target positions.
- **Distance Traveled (DT)**: This metric measures the total distance covered by all agents in rearranging the objects from the source configuration to the target configuration.
- **Completion Time (CT)**: This metric quantifies the time spent by the agents to complete the object rearrangement process from the source configuration to the target configuration.
- **Inference Time (IT)**: This metric indicates the total time spent by the model in planning all the rearrangement actions using 1 CPU. The maximum allowable planning time is set to 120 seconds.

Note that the placement success rate is computed across all rearrangement scenarios, while the distance travelled, completion time and inference time are computed only on successful rearrangements.

A. Algorithm Comparison

To evaluate the effectiveness of our approach, we conducted experiments on a range of simulated object rearrangement scenarios. These experiments involved different numbers of objects and agents, specifically 8, 12, and 16 objects with 2 and 3 agents. For each combination of agent and object, we performed 50 experiments encompassing various tasks, as depicted in Figure 2. Tables I and II shows how our method compares to multiple baseline methods. The baselines we evaluated include:

TABLE I
Placement Success Rate. We show the success rate (%) averaged over 2 and 3 agents for different number of objects (8, 12, 16). Higher is better.

Algorithms	# Objects		
	8	12	16
Classical (Random)	76.25	65.33	48.12
Classical (Greedy)	77.63	66.41	53.06
MANER (Ours)	**81.25**	**76.33**	**75.93**

TABLE II
Average Distance Traveled (meters) and Completion Time (seconds). We show the results for different sets of object-agent number pair. Lower is better.

# Objects	Algorithms	2 Agents		3 Agents	
		DT	CT	DT	CT
8	Classical (Random)	41.1	144.7	38.4	97.7
	Classical (Greedy)	35.2	117.4	32.1	85.3
	MANER (Ours)	**24.6**	**82.3**	**24.4**	**57.4**
12	Classical (Random)	64.5	218.4	68.2	166.5
	Classical (Greedy)	51.0	176.8	59.6	160.5
	MANER (Ours)	**36.6**	**116.4**	**37.5**	**87.5**
16	Classical (Random)	77.2	265.8	59.9	144.7
	Classical (Greedy)	65.1	231.6	50.4	123.0
	MANER (Ours)	**48.1**	**159.8**	**39.0**	**98.1**

1) Classical (random): This baseline method is a multi-agent variant of a search-based approach described in [5] for addressing non-monotone instances of object rearrangement. The method employs instance segmentation to detect objects and agents in the start and target scenes. Each agent is assigned a specific object, and the method determines the closest unoccupied target location for each object from the target scene. If the resulting multi-agent path, computed using [17] is free from collisions, the method proceeds iteratively to rearrange the remaining objects. However, if the computed path is obstructed, the method attempts to clear the path by randomly sampling available regions in the environment as potential relocation spots for the obstructing objects. It then moves the objects to the sampled locations. This process continues until a valid rearrangement solution is found or the time limit is exceeded.

2) Classical (greedy): This baseline method also begins with instance segmentation to identify objects and agents in the current and target scenes. The objects are then mapped one-by-one in the scenes using a bipartite graph matching algorithm based on computed A* distances between same-colored objects. Each agent is assigned an object from the start scene. The transfer locations for each agent-object pair are determined based on one-to-one mapping. If the resulting multi-agent path is free from collisions, the algorithm proceeds to rearrange the objects. If the path is obstructed, the method attempts to clear the path by sampling the closest available regions in the environment as potential relocation spots for the obstructing objects. It then moves the objects to the sampled locations. The process continues until a valid rearrangement solution is found or the time limit is exceeded.

Comparison to classical methods: Table I, Table II, and Figure 3 highlight the superior performance of MANER com-

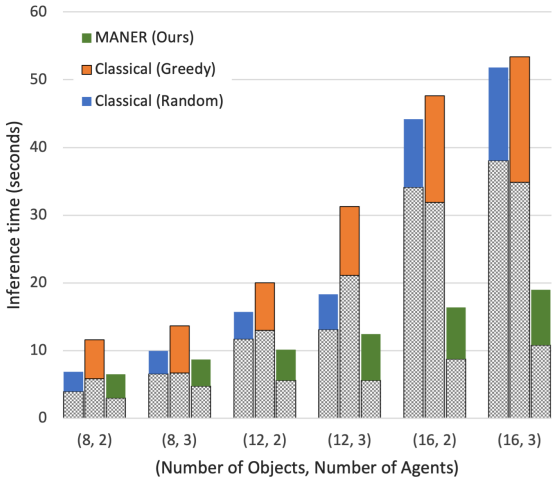

Fig. 3. Inference Time Comparison of Baseline Models and MANER on evaluated dataset with varying numbers of objects and agents. The time taken by path planning component is indicated as gray shaded area. The inference time for MANER increases almost linearly with increasing number of objects and agents, while it grows exponentially for classical models.

pared to the baseline methods. This performance improvement can be attributed to several factors. The classical approach often leads to collisions due to random object selection and placements, especially in cluttered environments. In contrast, MANER utilizes transformer networks and self-attention mechanisms, which effectively capture spatial relationships and dependencies among objects, agents, and static obstacles in the environment. This enables it to identify optimal object-agent pairs and corresponding regions, resulting in a higher success rate and reduced traversal time. Furthermore, the classical approach exhaustively considers all possible rearrangement actions, leading to longer execution times in the worst case. In contrast, the execution time of MANER grows linearly with the increasing number of agents, providing a more efficient solution. Also worth mentioning that most objects which MANER fails to rearrange are due to reachability constraints, as some objects cannot be picked or dropped since objects, agents, and static obstacles are positioned randomly as discussed in IV-A.

Generalization to Different Number of Objects and Agents: Table III demonstrates MANER's generalization capability across different scenarios involving varying numbers of objects and agents. While the training dataset for MANER comprised scenes with 2 and 3 agents and 8 to 16 objects, the evaluation results show that MANER performs effectively in challenging environments with 4 agents and a broader range of 8 to 24 objects. The success rate remains relatively consistent even with the addition of more agents to the scene, and only exhibits a slight decrease as more objects are introduced. Additionally, the total distance traveled by the agents remains similar, while the completion time decreases linearly, indicating that the rearrangement task is evenly distributed among all agents. These findings highlight MANER's robustness in handling scene clutter and efficiently solving complex rearrangement tasks.

TABLE III
GENERALIZATION TO UNSEEN NUMBER OF OBJECTS AND AGENTS.

#	Performance Metrics		
	SR (%)	DT (m)	CT (sec)
8 objects, 4 agents	82.5	25.9	52.6
12 objects, 4 agents	77.16	37.4	73.8
16 objects, 4 agents	73.5	48.5	107.6
24 objects, 4 agents	68.83	65.2	154.6

B. Ablation Studies

We conducted a series of experiments using 12 objects and 3 agents to analyze the impact of different components in our method. Specifically, we examined the impact of k, the object selection network, and the region proposal network. The results are presented in Table IV. The findings indicate the significance of all these components. When only the best region from the region proposal network is selected, the placement success rate drops slightly. When object selection is absent, the model randomly chooses objects for drop-off, resulting in a reduced success rate and increased distance traveled. Without the region proposal, the success rate decreases drastically as random regions are proposed instead of carefully selected regions for object transfers.

TABLE IV
ABLATION ANALYSIS OF THE VARIOUS COMPONENTS OF THE NETWORK

Algorithms	Performance Metrics		
	SR (%)	DT (m)	CT (sec)
MANER	79.9	35.2	89.9
w/ Region Proposal ($k = 1$)	72.5	40.12	99.1
w/o Object Selection	69.5	38.4	96.6
w/o Region Proposal	28.87	58.5	168.1

C. Real Robot Experiments

Finally, we conducted a series of real-world experiments using two TurtleBot3 robots in an arena with a size of 2.4 meters. Due to size constraints on the robots and the environment, the experiments were limited to scenarios with 4 and 6 objects. As depicted in Fig. 4, the robots were tasked with rearranging a scene consisting of 6 objects to achieve a desired target configuration. The real-world tasks presented additional challenges, including noisy scene segmentation and odometry drift caused by factors such as wheel slippage and sensor noise. These challenges occasionally resulted in rearrangement failures. Nonetheless, MANER demonstrated remarkable generalization to real-world settings with IT 9.5 ± 1.6 seconds, DT 19.6 ± 5.4 meters, and CT 180 ± 32 seconds across various rearrangement setups. Additional examples can be found in the supplementary materials.

VI. CONCLUSION, LIMITATIONS, AND FUTURE WORK

We presented the Multi-Agent Neural Rearrangement Planning (MANER) approach that rearranges objects to their desired configurations in cluttered environments. We evaluated MANER on challenging problems and demonstrated its sim-to-real generalizations. One challenge to our approach is the

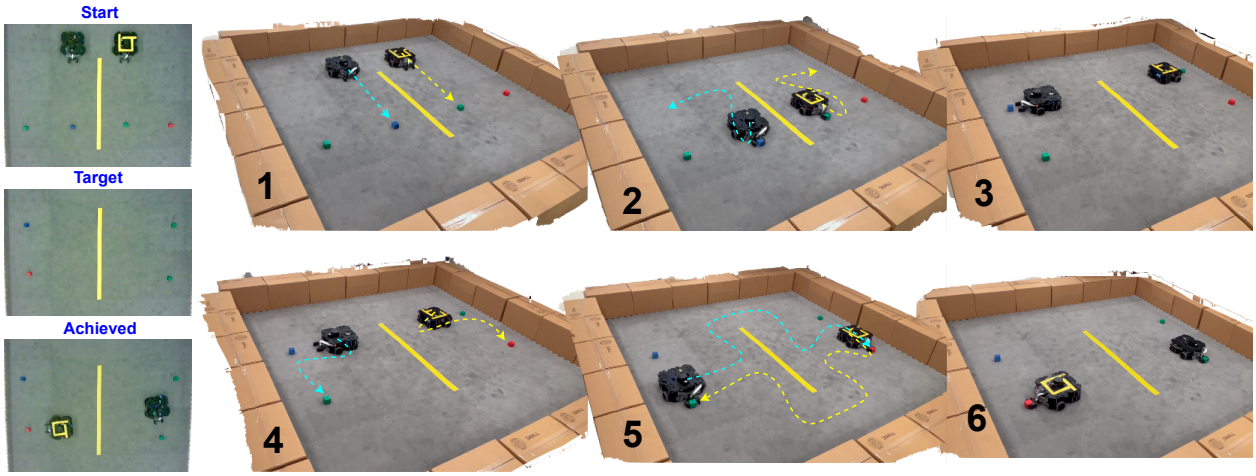

Fig. 4. Example of a real-world experiment with 4 objects and 2 agents. The images on the left show the start, target, and achieved configurations. The images labeled with numbers on the right show the intermediate configurations achieved by agents based on the trajectories from MANER.

precise low-level control since pixel-to-world coordinate mapping can result in precision errors and collisions, leading to failures. Another challenge is from our path planning module, which currently does not tackle orientation constraints during object placements. Hence, for future works, we envision building precise motion planning and control strategies to solve even harder multi-agent object rearrangement tasks.

REFERENCES

[1] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Transactions on Robotics*, 2022.

[2] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, "Spatial intention maps for multi-agent mobile manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8749–8756.

[3] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, *et al.*, "Rearrangement: A challenge for embodied ai," *arXiv preprint arXiv:2011.01975*, 2020.

[4] Y. Labbé, S. Zagoruyko, I. Kalevatykh, I. Laptev, J. Carpentier, M. Aubry, and J. Sivic, "Monte-carlo tree search for efficient visually guided rearrangement planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3715–3722, 2020.

[5] A. Krontiris and K. E. Bekris, "Dealing with difficult instances of object rearrangement." in *Robotics: Science and Systems*, vol. 1123, 2015.

[6] K. Ren, L. E. Kavraki, and K. Hang, "Rearrangement-based manipulation via kinodynamic planning and dynamic planning horizons," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1145–1152.

[7] 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.

[8] A. H. Qureshi, A. Mousavian, C. Paxton, M. C. Yip, and D. Fox, "Nerp: Neural rearrangement planning for unknown objects," *arXiv preprint arXiv:2106.01352*, 2021.

[9] H. Wu, J. Ye, X. Meng, C. Paxton, and G. S. Chirikjian, "Transporters with visual foresight for solving unseen rearrangement tasks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 756–10 763.

[10] L. E. Parker, D. Rus, and G. S. Sukhatme, "Multiple mobile robot systems," in *Springer Handbook of Robotics*, ser. Springer Handbooks, B. Siciliano and O. Khatib, Eds. Springer, 2016, pp. 1335–1384. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_53

[11] M. Levihn, T. Igarashi, and M. Stilman, "Multi-robot multi-object rearrangement in assignment space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5255–5261.

[12] J. Motes, T. Chen, T. Bretl, M. Morales, and N. M. Amato, "Hypergraph-based multi-robot task and motion planning," *arXiv preprint arXiv:2210.04333*, 2022.

[13] K. Gao and J. Yu, "Toward efficient task planning for dual-arm tabletop object rearrangement," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 425–10 431.

[14] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams, "Cooperative task and motion planning for multi-arm assembly systems," *arXiv preprint arXiv:2203.02475*, 2022.

[15] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[17] K. Yakovlev and A. Andreychuk, "Any-angle pathfinding for multiple agents based on sipp algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 27, 2017, pp. 586–594.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.