Scene Mover: Automatic Move Planning for Scene Arrangement by Deep Reinforcement Learning (Supplementary Material)

HANQING WANG, Beijing Institute of Technology
WEI LIANG*, Beijing Institute of Technology
LAP-FAI YU, George Mason University

ACM Reference Format:
Hanqing Wang, Wei Liang, and Lap-Fai Yu. 2020. Scene Mover: Automatic Move Planning for Scene Arrangement by Deep Reinforcement Learning (Supplementary Material). *ACM Trans. Graph.* 39, 6, Article 233 (December 2020), 10 pages. https://doi.org/10.1145/3414685.3417788

1 COMPARISON WITH PRIOR WORKS

In this section, we compare our work with prior robotics works from the problem setting and approach perspectives.

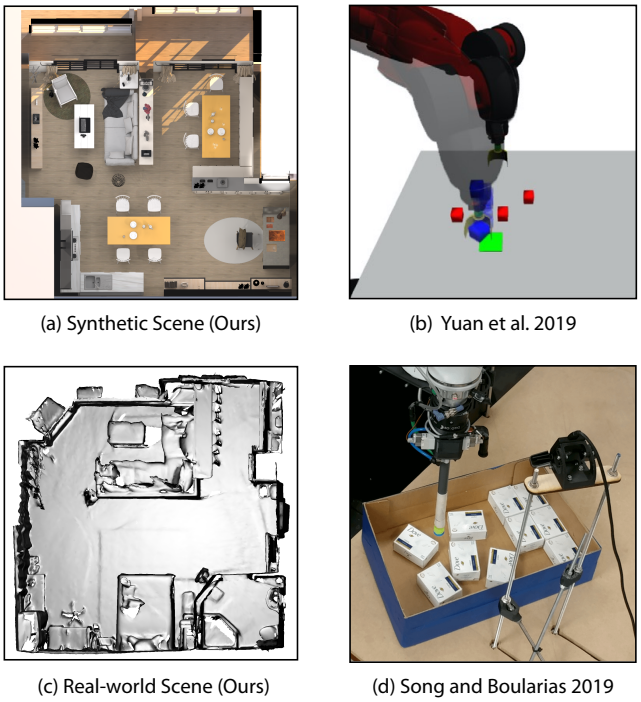
Problem Setting. From the problem setting perspective, we compare different works based on the following factors:

(1) Object Pose: the orientation of objects;
(2) Shape Variation: the shape variation of objects;
(3) Collision: collision detection during moving;
(4) Multiple Objects: handling multiple movable objects;
(5) Target Configuration: whether a target configuration is given;
(6) Scenarios: the testing scenarios.

The detail of the comparision is shown in Table 1.

A rearrangement task generally comes with a certain target configuration of the objects. An alternative rearrangement problem setting is based on object clustering [Song et al. 2019], where each object has a certain target region that it should move to. The latter case can be regarded as having a soft target configuration.

Most of the prior works considered multiple movable objects and collision detection. However, [Haustein et al. 2019; King et al. 2017; Labbé et al. 2020; Song et al. 2019; Yuan et al. 2019] did not consider the pose of objects probably because taking poses into consideration would substantially expand the space of layout states. Many prior works [King et al. 2017; Song and Boularias 2019; Yuan et al. 2019] also did not consider the variation in object shapes, assuming the same object shape for all objects in a scenario. [Haustein et al. 2019; Labbé et al. 2020] assumed simple object shapes such as cylinders and cubes. However, in most real-world scene rearrangement scenarios, the shapes of objects (i.e. furniture) vary a lot. Therefore the scene rearrrangement problem that our approach tackles is generally more complex.



(a) Synthetic Scene (Ours) (b) Yuan et al. 2019
(c) Real-world Scene (Ours) (d) Song and Boularias 2019

Fig. 1. Some scenarios tested in the experiments of prior works and our work. (b) and (d) are synthetic and real tabletop scenarios tested in prior works [Yuan et al. 2019][Song and Boularias 2019]. (a) and (c) are synthetic and real indoor scenes tested in our work. The scene complexity and settings of the prior works and our work are very different. For example, our work rearranges many furniture objects with a variety of shapes; the indoor scenes also have irregular boundaries and obstacles (e.g., walls) within the scenes. Our *Scene Transformer* approach driven by deep reinforcement learning plus MCTS is capable of handling such complexities.

Our work solves the rearrangement problem considering all the above factors. The prior works focus on solving tabletop rearrangement and are evaluated on synthetic scenarios as illustrated in Figure 1 (b) and (d). In contrast, our work is demonstrated on both synthetic and real-world indoor scenes (Figure 1 (a) and (c)), which are more sophisticated due to the larger number of furniture objects present, variations in object shapes, and irregular room layouts and obstacles. Our *Scene Transformer* approach is designed to handle such scene complexities.

*Corresponding author.

Authors' addresses: Hanqing Wang, Beijing Institute of Technology, hanqingwang@bit.edu.cn; Wei Liang, Beijing Institute of Technology, liangwei@bit.edu.cn; Lap-Fai Yu, George Mason University, craigyu@gmu.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
0730-0301/2020/12-ART233 $15.00
https://doi.org/10.1145/3414685.3417788

Table 1. Comparison between different works. Overall, our approach can tackle multi-object real indoor scene rearrangement considering object pose, shape variations, and collision constraints.

Works	Problem Setting & Approach						
	Object Pose	Shape Variation	Collision	Multiple Objects	Target Configuration	Scenarios	Planning Approach
[Yuan et al. 2019]	✗	✗	✓	✗	✓	Synthetic Tabletop	DRL(CNN)
[King et al. 2017]	✗	✗	✓	✓	✓	Synthetic Tabletop	MCTS
[Haustein et al. 2019]	✗	✓	✓	✓	✓	Synthetic Tabletop	RRT+TS
[Song and Boularias 2019]	✓	✗	✓	✓	✓	Synthetic Tabletop	Searching
[Labbé et al. 2020]	✗	✓	✗	✓	✓	Synthetic Tabletop	MCTS
[Song et al. 2019]	✗	✓	✓	✓	✗	Synthetic Tabletop	IL(CNN) + MCTS
Scene Transformer (Ours)	✓	✓	✓	✓	✓	Real/Synthetic Indoor Scene	DRL(CNN+LSTM) + MCTS

Approach. From the planning approach perspective, [Yuan et al. 2019] used deep neural networks to learn how to rearrange a single object through deep reinforcement learning. Most of the existing works [Haustein et al. 2019; King et al. 2017; Labbé et al. 2020; Song and Boularias 2019] solved the multi-object planning problem based on a search framework. However, since the problem setting varied, specific designs were adopted to fit with the problems correspondingly.

For example, [King et al. 2017; Labbé et al. 2020; Song et al. 2019] adopted Monte Carlo Tree Search (MCTS) in their searching. [Haustein et al. 2019] proposed a RRT-based tree searching (RRT+TS) algorithm. [Song et al. 2019] adopted a policy network in the Rollout step of MCTS, which our approach also does similarly. However, our task is different from theirs and both the training paradigm and the network design are different. They trained their policy network (with a CNN backbone) by using random success demonstrations, known as Imitation Learning (IL). For our real-world scene rearrangement planning problem, the scenes are usually much more complicated, making them hard to annotate by random methods. Additionally, sequential decision making is critical due to the complexity of the scene, which calls for a very strong planning agent and training method. To tackle such issues, we propose a powerful neural network with LSTM units to process sequential information, as well as a novel deep reinforcement learning (DRL) paradigm with reward shaping that is effective for solving our problem.

2 NETWORK STRUCTURE

The structure of the Q-network is shown in Fig. 3. The Q-network comprises two parts. The first part is a residual convolutional encoder. It encodes the input (an initial layout and a target layout) to a 4096-d feature vector. There are 10 convolution layers in total. The convolutional layers adopt batch normalization. ELU is used as the activation function.

The second part is LSTM layer followed by a fully-connected layer, which is appeneded to the encoder as a regressor. The 4096-d feature vector passes through the layers and finally outputs the predicted Q-value for each action.

3 ALGORITHM

The pseudo code of the *Scene Transformer* is illustrated in Algorithm 1. It demonstrates an iteration of *Scene Transformer*'s MCTS.

ALGORITHM 1: An iteration of *Scene Transformer*'s MCTS

Input: the root node l_t of the current searching tree , the Q-network $Q_p(\cdot, \cdot)$, maximum simulation depth D, node score's function in selection step $\widetilde{Q}(\cdot)$, node's Q-value $Q_m(\cdot)$, node visit count $N(\cdot)$, function for retrieving the layout represented by the parent's node $p(\cdot)$, simulation function $e(\cdot, \cdot)$, reward function $r(\cdot, \cdot)$

Output: The best action a^*, the root node l'_t of the next iteration's searching tree.

```
for j = 1 to N do
    l = arg max_l Q̃(l);
    a = arg max_a Q_p(l, a), a is a legal action and not be expanded;
    Expand a new node representing l' as a child of the node
      representing l, the edge stores action a;
    sum = 0;
    l_0 = l'
    for k = 1 to D do
        a_k = arg max_a Q_p(l_{k-1}, a);
        l_k = e(l_{k-1}, a_k);
        sum = sum + r(l_{k-1}, a_k);
        if l_k is the target layout then
            break;
        end
    end
    N(l') = 1;
    Q_m(l') = sum;
    while l'! = l_t do
        l' = p(l');
        a* = arg max_a Q_m(e(l', a));
        Q_m(l') = γ Q_m(e(l', a*) + r(l', a*));
        N_m(l') = N_m(l') + 1
    end
end
a* = arg max_a Q_m(e(l_t, a))
l'_t = e(l_t, a*)
return a*, l'_t
```

4 REWARDS DISCUSSION

In this section we discuss the influence of the reward terms. Empirically, a basic reward mechanism is composed of the positive rewards and negative rewards. The positive rewards include the gains received during the task and the reward of finishing the task. The negative rewards include the penalty received during the game. Thus our basic reward mechanism has three terms:

- First-arrival/leave: +/-4

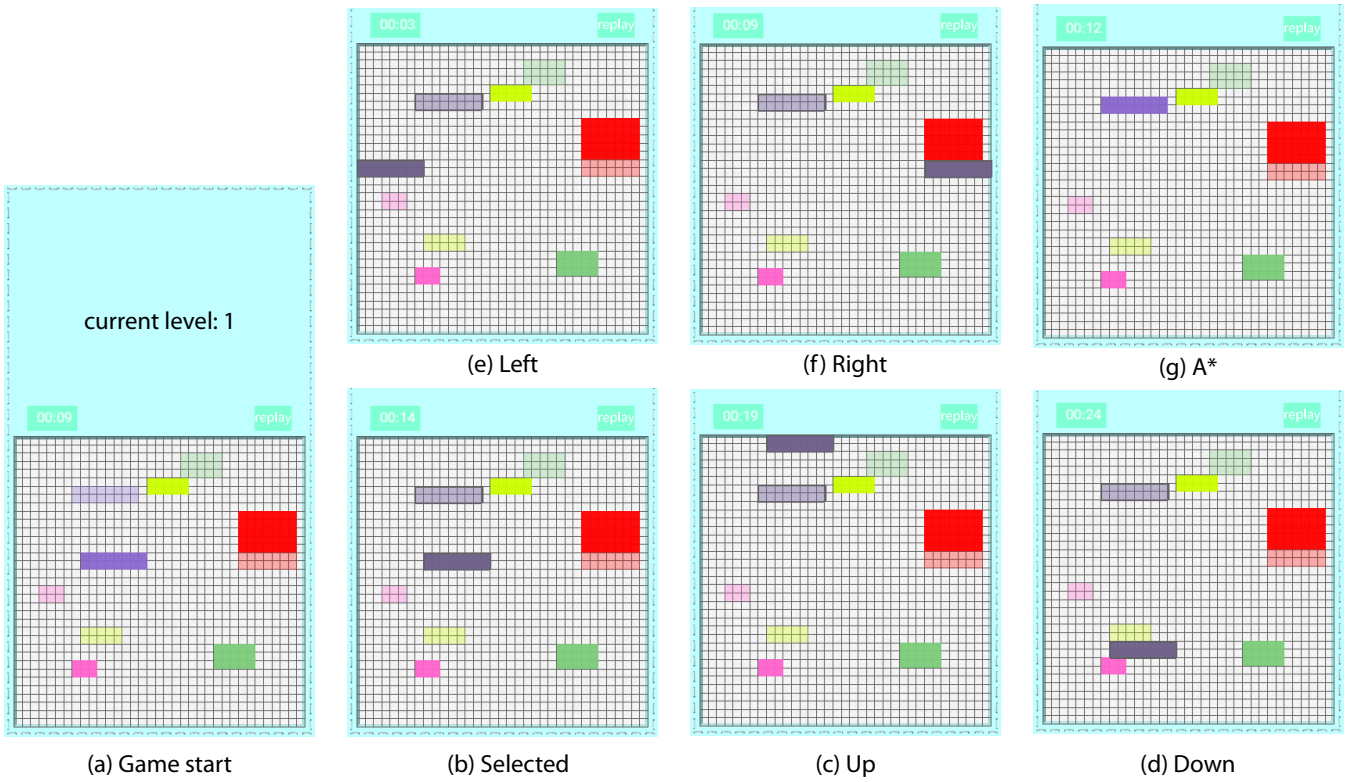


Fig. 2. (a) The start of a game. (b) An object is selected. (c)-(f) The object is slided up, down, left and right as far as it can reach. (g) The user taps on the target position and move the object to it through a path searched by A* Searching.

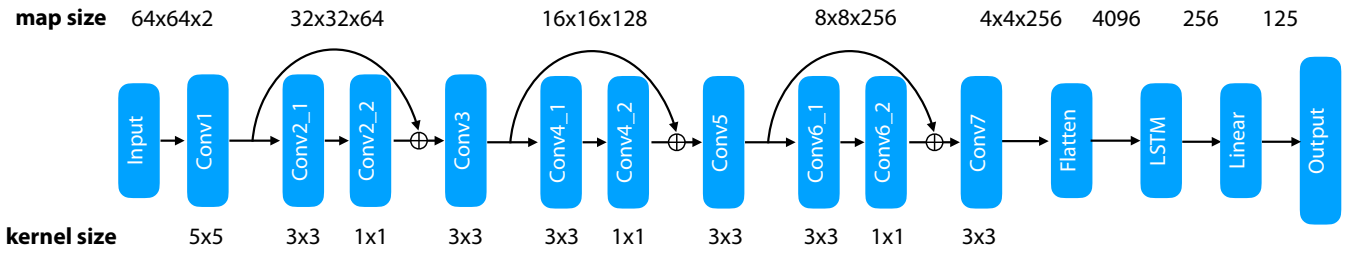


Fig. 3. The structure of the Q-network.

- Success: 50

Basing on those reward terms, we did an ablation study of other terms we use:

- Base: -1
- Repetition: -2
- Multi-arrival/leave: +/-2

We trained the Q-network under 3 different reward settings: 1) without Base term, 2) without Repetition term, and 3) without Multi-arrival/leave term.

We tested each reward setting on completing the move task without MCTS framework. As shown in Table 2, with the complete reward, our network achieves best success rate in both 5-obj. and 17-obj. difficulty levels and comparable success rate in 9-obj. and 13-obj. difficulty levels. The results validate that the terms contribute to the learning of policy.

5 SCENE TRANSFORMER GAME MANIPULATION

5.1 User Interface of the Game for Synthetic Layouts

Fig. 2 illustrates the manipulation of the Scene Transformer game for synthetic layouts. The current positions of objects are indicated by solid colors. The target positions are indicated by the corresponding transparent color.

Table 2. Success rates of network trained with different reward mechanisms in different difficulty levels.

Rewards	5-obj.	9-obj.	13-obj.	17-obj.
w/o. Base	55%	70%	30%	5%
w/o. Repetition	80%	75%	25%	15%
w/o. Multi-arrival/leave	85%	60%	30%	10%
Complete	100%	70%	30%	25%

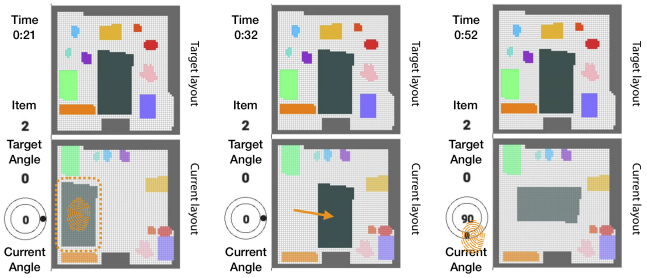


Fig. 4. The finger print represents a tap opertation. The arrow represents a slip operation. (a) Select the item bounded in the dashed box. (b) Drag the selected item. (c) Rotate the selected item.

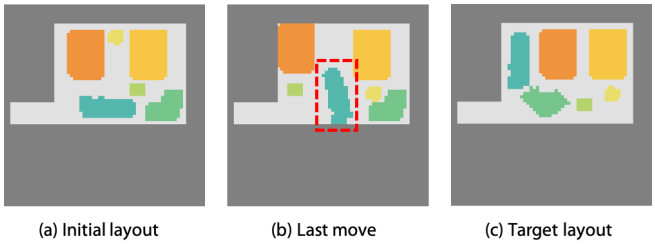


Fig. 5. A failure case. The last move of this user's record is shown in (b).

If a user taps on an object, the object is selected. Then both the current position of the object and the target position are highlighted by thick boxes. So the user can see the target position more clearly.

Under the selection mode, the user can move the selected object by sliding it up, down, left, or right. The user can also tap on the target position. If there is an accessable path from the current position to the target position, the selected object will be moved to the target directly. The accessable path is computed by A^* searching algorithm.

With the selection mode, if the user taps on ground (grey areas) or other objects, the former selection will be canceled.

5.2 User Interface of the Game for Real-World Layouts

The user interface of the game for scanned real-world layout data is shown in Fig. 6. The left bar shows the time used, the information of the selected item including the ID, target angle and the current angle. The top shows the target layout, the bottom shows the current layout.

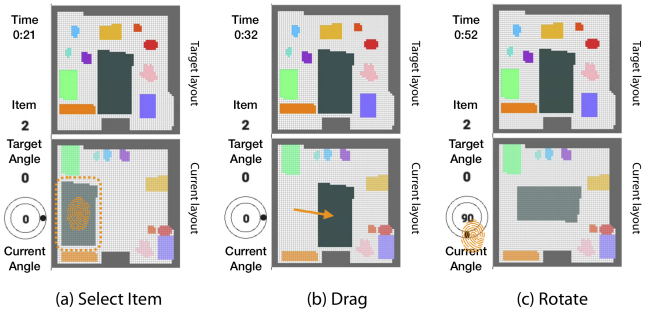


Fig. 6. The finger print represents a tap opertation. The arrow represents a slip operation. (a) Select the item bounded in the dashed box. (b) Drag the selected item. (c) Rotate the selected item.

The item in solid color is at the target position with correct pose, or it dose not reach the target position or has the wrong pose. User can select a item to move by tapping on the item in the current layout. Then user can drag the selected item to somewhere by slipping the screen. User can also rotate the item to the ideal angle by tapping on the wheel of current angle.

6 RATIONALE BEHIND 2D REPRESENTATION

We focus on 2D in this work because: (1) 3D evaluation is much more expensive, causing difficulty in training and testing. (2) It is hard to collect human data in 3D scenes for evaluation due to the difficult 3D manipulation of objects. (3) 2D movements can deal with most scene arrangement scenarios.

7 DETAILS OF EXPERIMENT DATA

7.1 All Layouts used in Comparison Experiments

Fig. 7, 8, 9, and 10 show all used layouts in the comparison experiment. Totally, there are 4 difficulty levels. In each level we generate 20 pairs of layouts. Each pair contains an initial layout and a target layout. In the figures, the rectangles with different colors represent different objects.

7.2 All Real-World Layouts used in Evaluation

All layouts used in our evaluation on real scanned data are shown in Fig. 11. There are 20 layouts in total. The detailed data comparision is shown in Table 3.

7.3 Failure Cases of Participants in Real-World Layouts

To figure out how users failed in some cases, we checked all failure records and found that most participants gave up when the objects blocked the way of other objects. One failure case is shown in Fig. 5. In the last move, the user moved the object in the red dashed box to the middle of the room which then blocked the objects at the bottom-right corner of the room. Those failure cases have an average number of steps of 17.3 which is a little bit higher than the average number of steps of 15.7 in the success cases. It shows that the users might have put efforts to solve the failure cases before they gave up.

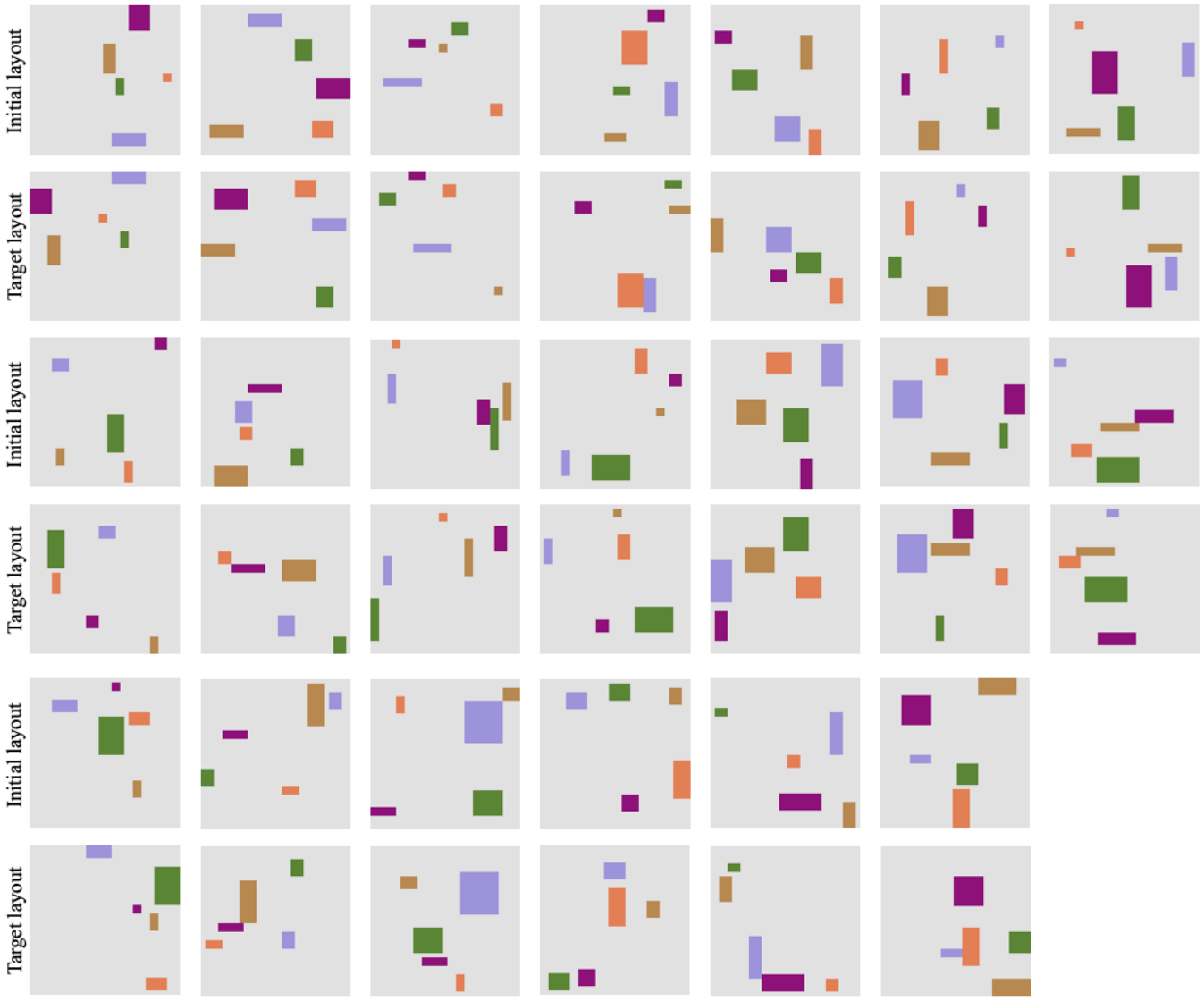


Fig. 7. The initial and target layouts of 5-obj. difficulty level used for the comparison experiments.

REFERENCES

Joshua A Haustein, Isac Arnekvist, Johannes Stork, Kaiyu Hang, and Danica Kragic. 2019. Learning Manipulation States and Actions for Efficient Non-prehensile Rearrangement Planning. *arXiv preprint arXiv:1901.03557* (2019).

Jennifer E King, Vinitha Ranganeni, and Siddhartha S Srinivasa. 2017. Unobservable monte carlo planning for nonprehensile rearrangement tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 4681–4688.

Yann Labbé, Sergey Zagoruyko, Igor Kalevatykh, Ivan Laptev, Justin Carpentier, Mathieu Aubry, and Josef Sivic. 2020. Monte-Carlo Tree Search for Efficient Visually Guided Rearrangement Planning. *IEEE Robotics and Automation Letters* (2020).

Changkyu Song and Abdeslam Boularias. 2019. Object Rearrangement with Nested Nonprehensile Manipulation Actions. *arXiv preprint arXiv:1905.07505* (2019).

Haoran Song, Joshua A Haustein, Weihao Yuan, Kaiyu Hang, Michael Yu Wang, Danica Kragic, and Johannes A Stork. 2019. Multi-Object Rearrangement with Monte Carlo Tree Search: A Case Study on Planar Nonprehensile Sorting. *arXiv preprint arXiv:1912.07024* (2019).

Weihao Yuan, Kaiyu Hang, Danica Kragic, Michael Y Wang, and Johannes A Stork. 2019. End-to-end nonprehensile rearrangement with deep reinforcement learning and simulation-to-reality transfer. *Robotics and Autonomous Systems* 119 (2019), 119–134.

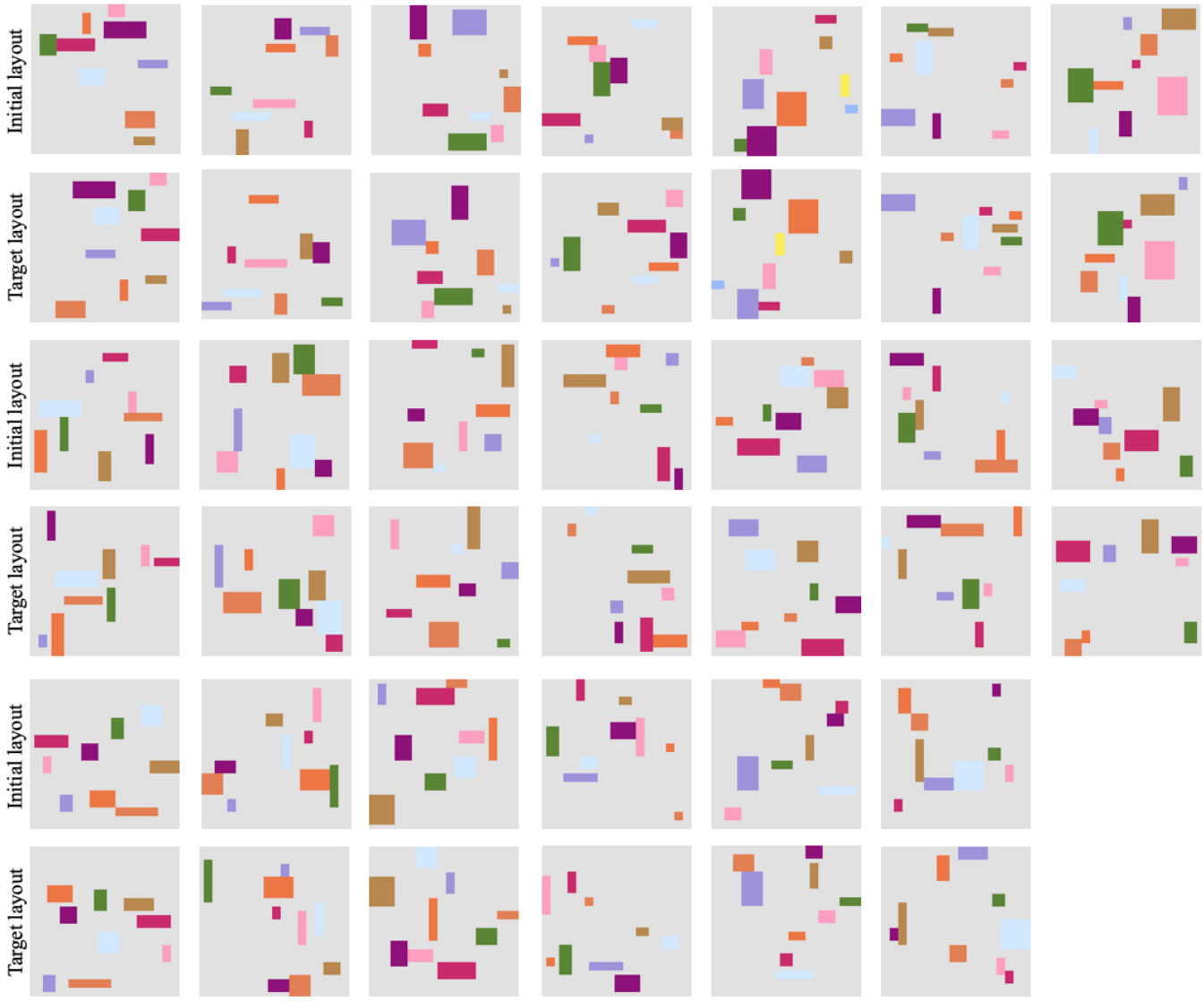


Fig. 8. The initial and target layouts of 9-obj. difficulty level used for the comparison experiments.

Fig. 9. The initial and target layouts of 13-obj. difficulty level used for the comparison experiments.

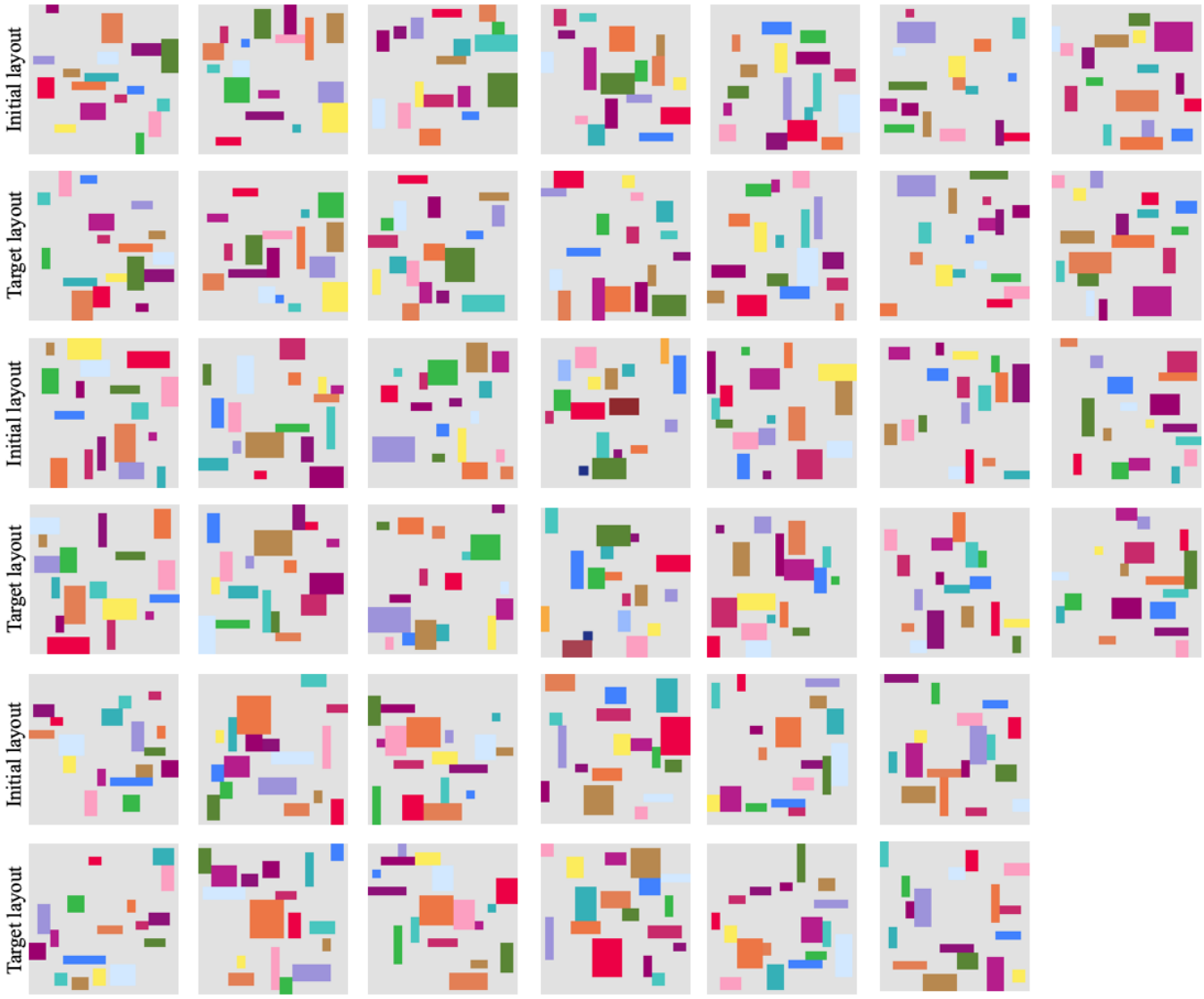


Fig. 10. The initial and target layouts of 17-obj. difficulty level used for the comparison experiments.

Fig. 11. The initial and target layouts of real scanned scenes.

Table 3. The detailed step and trajectory length of *Scene Transformer* and Human.

Scene	Obj.	*Scene Transformer*		Human	
		No. of steps	Travel length	No. of steps	Travel length
1	6	6	233	11.4	805.8
2	10	13	304	23.3	1344.4
3	14	16	63	27.8	642.9
4	9	9	152	12.5	618.8
5	9	8	129	12.4	563.9
6	7	7	179	9.4	567.1
7	11	12	274	14.1	881.5
8	17	9	64	21.1	527.8
9	9	7	131	11.3	581
10	12	11	194	14	598.8
11	14	24	292	32.7	1191
12	6	8	106	26.6	992.6
13	10	43	610	23.4	1178.8
14	13	11	119	23.3	988.4
15	9	10	258	16.4	911.6
16	19	8	137	19.7	645.2
17	9	7	228	10.2	647
18	14	17	226	20.4	807.6
19	11	23	225	13	652.1
20	10	10	103	16.6	604