MOBILE CONSTRUCTION BENCHMARK

Anonymous authors
Paper under double-blind review

ABSTRACT

We need intelligent robots to perform mobile construction, the process of moving in an environment and modifying its geometry according to a design plan. Without GPS or similar techniques, carefully engineered and learning-based methods face challenges to exactly achieve the plan due to the difficulty of accurately localizing the robot while strategically evolving the environment, because common tasks (manipulation/navigation) address at most one of the two coupled aspects. To seek a generic solution, we simplify mobile construction in 1/2/3D grid worlds to benchmark the performance of existing deep RL methods on this partially observable MDP. Our results show that the coupling makes this problem very challenging for model-free RL, and emphasize the need for novel task-specific solutions.

1 INTRODUCTION

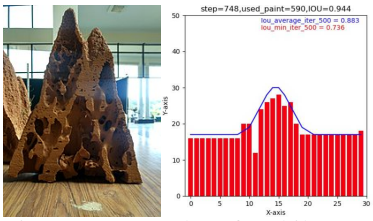


Figure 1: Examples of mobile construction in nature (left), a termite mound section (Wikimedia Commons, 2018), and in our benchmark (right), 1D brick-laying.

Robotic construction is reborn along with AI because of both its growing benefits in time, quality, and labor cost, and the even more exciting role in extraterrestrial exploration. Efficiently achieving this in large scale with flexibility requires robots to have an ability like animal architects (e.g., mound-building termites and burrowing rodents): *mobile construction*, i.e., the process of an agent moving around and simultaneously modifying its surroundings according to a design plan.

Engineering mobile construction with existing techniques is difficult. Needless to mention the unaddressed materials and physics related problems, a fundamental challenge for AI and robotics is *the tight bi-directional coupling of robot localization and long-term planning for environment evolution*. If GPS and techniques alike are not available (often due to occlusions), robots have to rely on SLAM for accurate pose estimation. But in our case the robot needs to take advantage of the bi-directional coupling to strategically change the environment to improve localization, violating the basic static environment assumption in matured visual SLAM methods (Saputra et al., 2018).

Deep reinforcement learning (RL) offers another possibility to tackle this challenge, especially given the recent success of deep model-free RL in game playing and robot control. What if we train deep networks to learn a generic, efficient, but complex policy that controls the robot to strategically build localization landmarks as temporary structures which eventually evolve to the design plan? Based on this, without the loss of generality, we omit the less relevant complexities in the real-world mobile construction such as the physical dynamics between robots and environments, and design a series of simplified tasks in 1/2/3D grid worlds that focus only on the aforementioned challenge.

In this paper, we focus on using those tasks to benchmark some selected deep RL algorithms with state-of-the-art performance in other tasks such as game playing, robot manipulation or navigation. Surprisingly, although these partially observable MDP (POMDP) tasks may seem similar to, if not simpler than, Atari games (Mnih et al., 2013), the results show great difficulties in those algorithms.

In summary, our contributions include: 1) a suite of novel and easily extensible RL tasks as open-source fast Gym environments, which directly connect to important real-world robotic problems; 2) a comprehensive benchmark of baseline models, which demonstrates the unexpected difficulties in these tasks for previously successful deep RL algorithms; and 3) a detailed ablation study revealing insights about the causes of those difficulties as the coupling explained above, which calls for novel task-specific algorithms from this community to solve the problem more effectively.

2 RELATED WORK

Mobile construction robots. Recently we see a rising trend of 3D printing using mobile robots all around the globe for construction and manufacturing (Werfel et al., 2014; Jokic et al., 2014; Ardiny et al., 2015; Nan, 2015; Marques et al., 2017; Buchli et al., 2018; Zhang et al., 2018; Melenbrink et al., 2020). All of those are carefully engineered systems that either assume some global localization ability or only work for specific scenarios, which restricts their feasibility in large scale. Moreover, none of them address the aforementioned challenge from a theoretical perspective.

POMDP solvers. We model mobile construction tasks by POMDP. An intuitive thought is to assume a perfect knowledge of the transition and observation models (although we do not), and then try them on existing offline or online solvers such as SARSOP (Kurniawati et al., 2008) or POMCP (Silver & Veness, 2010). But due to environment modification, both our state and observation spaces are huge, comparing to existing large POMDP tasks such as in Wandzel et al. (2019) with 10^{27} states. Even our simplest 1D task can easily have a much larger state space than the Go game ($3^{361} \ll 100^{100}$ for 100 grids with max height 100). So it is non-trivial to tackle our tasks using model-based methods and model-based RL would also face challenges. We will mainly focus on model-free RL algorithms in this paper and leave the investigation of model-based methods for our future work.

Existing RL tasks. A major contribution of this paper is to stimulate deep RL research with novel tasks that exhibit fundamentally different features than typically benchmarked RL tasks and that are relevant for a real application: mobile construction. Locomotion tasks (Duan et al., 2016) have no need for localization and do not modify the surroundings. Manipulation tasks (Fan et al., 2018; Yang et al., 2019; Labbé et al., 2020; Li et al., 2020) require agents to move objects but usually assume the pose of the robot base is fixed or known, which is challenging in a dynamic environment (Saputra et al., 2018). Visual navigation (Zhu et al., 2017; Gupta et al., 2017; Mo et al., 2018; Zeng et al., 2020) requires localizing the agent but without changing the environment on purpose. Game playing for Atari (Mnih et al., 2013), first-person-shooting (Lample & Chaplot, 2017), and real-time strategy games (Synnaeve et al., 2016; Jaderberg et al., 2019) either have trivial localization or are not evaluated based on the accuracy of environment modifications.

Baseline selection. We create baseline methods from a set of state-of-the-art model-free Deep RL algorithms. Since our tasks live in grid worlds similar to many Atari games, our first choice is DQN (Mnih et al., 2013) which has achieved great success on many Atari tasks with high-dimensional states, benefiting from better Q-learning on representations extracted via deep networks. Another baseline is Rainbow (Hessel et al., 2017), combining six extensions on top of the base DQN, achieving superior performance to any of the individual extensions alone. These include double Q-learning (Hasselt, 2010), prioritized experience replay (Schaul et al., 2016), dueling network architectures (Wang et al., 2016), multi-step TD learning (Sutton, 1988), noisy networks (Fortunato et al., 2017), and distributional reinforcement learning (Bellemare et al., 2017).

Of course, DQN is sub-optimal for POMDP due to its limited ability to represent latent states from long-term history, which could be critical for mobile construction. To address this issue, we add a baseline using DRQN (Hausknecht & Stone, 2015) with a recurrent Q-network. Besides, a sparse reward function may bring additional challenges to our tasks. Hindsight Experience Replay (Andrychowicz et al., 2017) helps off-policy RL algorithms learn efficiently from sparse rewards without complex reward engineering, which is combined with DRQN as another baseline.

In addition, we add two more actor-critic based baselines. One is Soft Actor-Critic (SAC) for discrete action settings (Christodoulou, 2019). Compared with the original SAC (Haarnoja et al., 2018), SAC-Discrete inherits the sample efficiency and tailors the effectiveness for discrete action space which suits our tasks. The other is Proximal Policy Optimization (PPO) (Schulman et al., 2017). Whereas the standard policy gradient method performs one gradient update per data sample, PPO enables multiple epochs of minibatch updates by a novel objective with clipped probability ratios.

3 MOBILE CONSTRUCTION IN GRID WORLD

We formulate a mobile construction task as a 6-tuple POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R}, P \rangle$, in which a robot is required to accurately build geometric shapes according to a design plan P in a grid world. The state space $\mathcal{S}$ is represented as $\mathcal{S} = \mathcal{G} \times \mathcal{L}$, where $\mathcal{G}$ is a space of all possible grid world state G

storing the number of bricks at each grid, and $\mathcal{L}$ is a space of all possible robot locations l in the grid world. At each time step, the robot takes an action $a \in \mathcal{A}$ which is either moving around or dropping a brick at or near its location. Moving a robot will change its location according to the *unknown* probabilistic transition model $\mathcal{T}(l'|l, a)$. Meanwhile, dropping a brick will change the grid world state G at or near l *without any uncertainty*. The robot can make a local observation $o \in \mathcal{O}$ of G centering around its current location, with a sensing region defined by a half window size W_s. $\mathcal{R}(s, a; P)$ is the reward function depending on the design plan $P \in \mathcal{G}$, which is just a goal state of the grid world that the robot needs to achieve.

The aforementioned coupling difficulties in localization and planning are realized via two factors in this setting. First, the partial observability makes robot localization necessary. Second, the environment uncertainty in $\mathcal{T}(l'|l, a)$ simulates real world scenarios where motion control of the mobile robot is imperfect and the odometry is error-prone. We implement this by sampling the robot's moving distance d in each simulation time step from a uniform distribution.

3.1 1D ENVIRONMENT

In 1D environments (Figure 2), the robot will move within a 1D grid world and the ground-truth plan P is a 1D curve. The grid world state $G \in \mathbb{R}^W$ is a vector, where W is the width of the environment. In order to provide more information to the agent, we augment the partial observation o with N_s, and N_b as the observation $o_{env} = \langle o, N_s, N_b \rangle$ of the environment. Here $o \in \mathbb{R}^{2W_s+1}$ is a vector that indicates the current region of the environment observed by the robot in a width-limited window with size $2W_s+1$, and each element of o represents the number of the bricks. N_s is the number of moves the robot has already taken. N_b is the number of bricks already used by the robot. A discrete action $a \in \mathcal{A} = \{0, 1, 2\}$ represents move-left, move-right, or drop-brick respectively. The reward function $\mathcal{R}$ is defined as

$$\mathcal{R} = \begin{cases} 10 & \text{drops brick on } P, \\ 1 & \text{drops brick below } P, \\ -1 & \text{drops brick over } P, \\ 0 & \text{moves.} \end{cases} \tag{1}$$

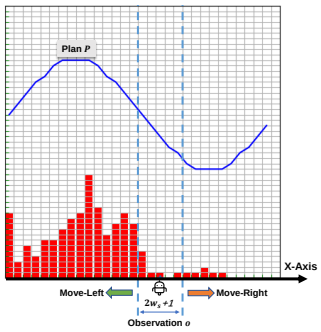


Figure 2: 1D environment: a robot moves along the x-axis and lays bricks (red). The two vertical blue dash lines indicate its sensing region for vector o. The $o = (5, 1, 1, 0, 1)$ in this case. The blue curve is the ground-truth plan P.

Each episode will finish when $N_s = N_{smax}$ or $N_b = N_{bmax}$, where the former is the maximum number of steps and the latter the maximum number of bricks (the integrated area of the plan P within each episode). We set N_{smax} reasonably large to ensure the plan completion.

For the 1D environment, we define two types of tasks: **static** and **dynamic** plan tasks. The static plan task requests the agent to build a static shape P which remains the same for each episode. The ground-truth plan P will vary for each episode for a dynamic plan task. For the static plan task, we consider three types of shape plans $P \in \mathbb{R}^W$: sin function curve, Gaussian curve and Step function curve (see Figure 5a to Figure 5c). For the dynamic plan task, we generate P (Figure 5d) based on the following equation: $P = a\sin(bx + c)$, where $a \sim U(3, 12), b \in \{1, 2, 3\}$, and $c \sim U(-\pi, \pi)$. The coefficients a, b and c are chosen randomly for each episode. For simulating real world condition where robots should have access to the design plan, we append the plan P to the environment observation o_{env} for a dynamic plan task as a 4-tuple $o_{env} = \langle o, N_s, N_b, P \rangle$.

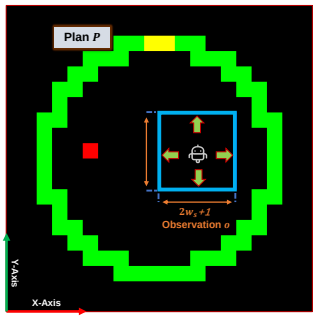


Figure 3: 2D environment: a robot moves in a 2d plane and lays bricks (red/yellow: in/correct laying). The blue box is its sensing region for o, which is a 2d array with size $(2W_s+1)^2$. The green ring is the ground-truth plan P.

3.2 2D ENVIRONMENT

In 2D environment (Figure 3), the robot will move and print binary plans in a 2D grid world. A grid world state $G \in [0, 1]^{W \times H}$ is 2D binary matrix, where W and H are the width and height of the environment. Each element

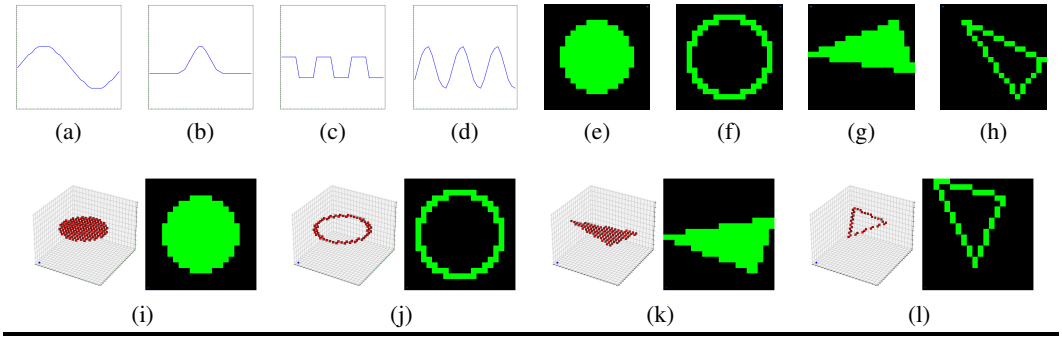


Figure 5: Example plans. The 3D red patches are the top-most surfaces of ground-truth plans.

of G represents if there is a brick or not. The robot observes a 2D region of the environment $o \in [0,1]^{(2W_s+1)\times(2W_s+1)}$. A discrete action $a \in \mathcal{A} = \{0,1,2,3,4\}$ could represent move-left/-right/-up/-down, or drop-brick respectively. The reward function $\mathcal{R}$ is defined as

$$\mathcal{R} = \begin{cases} 5 & \text{drops brick at the correct position,} \\ 0 & \text{drops brick at the wrong position,} \\ 0 & \text{moves.} \end{cases} \tag{2}$$

The 2D environment uses the same stop criteria as in 1D. In addition to the static and dynamic tasks as defined in 1D, the 2D tasks are further grouped in two categories: **dense** and **sparse**. Here, the term dense means the ground-truth plan is a solid shape as showed in Figure 5e and Figure 5g and sparse means a shape is unfilled as showed in Figure 5f and Figure 5h. For dynamic plan tasks, three vertexes of a triangle are randomly picked within the grid world. Note that we tried to add penalty for incorrect printing similar to equation 1, but we find this will cause the agent to always choose moving instead of printing because the latter has higher chance of receiving negative rewards (especially for sparse plans). So we removed the penalty and observed better performance.

3.3 3D ENVIRONMENT

In 3D environment (Figure 4), the robot will move and build a 3D plan within a 3D grid world. The grid world state $G \in \mathbb{R}^{W\times H}$ and observation $o \in \mathbb{R}^{(2W_s+1)\times(2W_s+1)}$ are all 2D matrices, and each element of o represents the number of bricks in that grid. Different from the 1D and 2D environments where the size of the robot is not considered, the dimension of the robot in the 3D environments is introduced (occupying 1 grid) so the robot's motion will be **obstructed** by the built bricks. Therefore, the robot can only build bricks around itself instead of building at its position as in 1D and 2D. So the action will be: $a \in \mathcal{A} = \{0,1,2,..,7\}$, which sequentially represent move-left/-right/-forward/-backward, and drop-brick on the left/right/front/rear side. The reward function $\mathcal{R}$ is defined the same as in the 1D case (see equation 1). In addition to the same stop criteria as in the 1D/2D cases, the game will stop when the robot is obstructed by the built bricks and cannot move anymore. The setup of plans P in 3D is similar to the ones in 2D as shown in Figures 5i-5l.

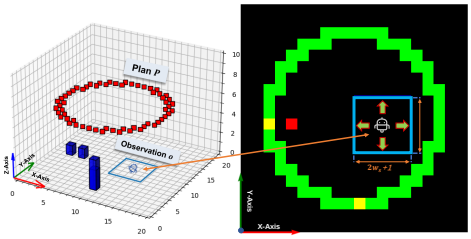


Figure 4: 3D environment: a robot moves in a 2d x-y plane and lays bricks (dark blue cubes). Left/right is the 3D/top view of the environment. The blue box is similar to Figure 3. The red ring in the left image is the top-most surface of ground-truth plan P.

4 BASELINE SETUP

Six state-of-the-art (SOTA) baseline algorithms are considered in our paper. In this section, we explain detail architecture design and hyperparameter setup for each algorithm. Here, we fix the

environment constants as following: 1) half window size W_s is 2 and width of the environment W is 30 for 1D environments; 2) W_s is 3 and W and H are 20 for 2D and 3D environments.

DQN For the static plan tasks, we use an MLP with three hidden layers containing [64,128,128] nodes with ReLU activation function for each layer. Three convolutional layers with kernel size of 3, and ReLU activation functions are used to convert the ground-truth P to feature vector for dynamic plan tasks. We train DQN on each task for 3,000 episodes. Batch size is 2,000, and replay buffer size 50,000. A random search is used to tune the hyperparameters, especially the learning rate. Additionally, instead of only using the current frame, we tried to stack ten frames of historical observations in the replay buffer. This is similar to how the original DQN handles history information for Atari games (Mnih et al., 2013), but no significant difference was found.

DRQN As for the DRQN (Hausknecht & Stone, 2015), we simply add one recurrent LSTM layer to the Q network used in the DQN. The hidden state dimension is 256 for all tasks. We train it for 10,000 episodes with batch size of 64 and replay memory size of 1,000.

DRQN+Hindsight We add hindsight experience replay (Andrychowicz et al., 2017) to the DRQN baseline above. At the end of each training episode, the transitions $\langle o_{env}^t, a^t, \mathcal{R}(s^t, a^t; P), o_{env}^{t+1} \rangle$ of each time step t will be relabeled as $\langle o_{env}^t, a^t, \mathcal{R}(s^t, a^t; G^T), o_{env}^{t+1} \rangle$, where we change the P to the grid world state G at terminate step T. Both of these two transitions will be stored into the replay buffer for optimizing the Q-network. We train this DRQN+Hindsight for 10,000 episodes with batch size of 64 and replay memory size of 1,000.

PPO To benchmark PPO, we use the Stable Baselines implementation (Hill et al., 2018) and train for 10 million time steps with a shared network of 3 layers of 512 neurons with tanh activations. For the hyperparameters, we use the 1D static environment to tune the learning rate, the batch size, the number of minibatches size, and the clipping threshold. We found that the most sensitive parameters were the batch size and the minibatch size and chose the following values: $1\text{e}5$ for the batch size, $1\text{e}2$ for the number of minibatches, $2.5\text{e}{-4}$ for the learning rate and 0.1 for the clipping threshold.

Rainbow For the Rainbow implementation, we used 3 noisy hidden layers (Fortunato et al., 2017) with 128 nodes in each layer, and ReLU nonlinear activation functions. Rainbow has a large set of hyperparameters, as each of the six components adds additional hyperparameters. We used those suggested in (Hessel et al., 2017) as a starting point, but they led to poor results on our specific task and environment designs. As a grid search over such a large hyperparameter space was impractical, we used a random search approach. Based on empirical results, the algorithm was most sensitive to learning rate, as well as V_{min}, V_{max}, and n_{atoms}, which define the value distribution support predicted by the distributional Q-network. Generally, a V_{min} value of -5, V_{max} of 35, and $n_{atoms} = 101$ provided stable performance, and were chosen heuristically based on an approximate range of discounted rewards possible in our environments. We used a prioritized experience replay buffer of size $1\text{e}4$, with priority exponent ω of 0.5, and a starting importance sampling exponent β of 0.4. Additionally, we used multi-step returns with $n = 3$, and noisy network σ_0 = 0.1. Finally, we used a learning rate of $5\text{e}{-5}$ for 1D and 2D environments, and $1\text{e}{-4}$ for 3D environments.

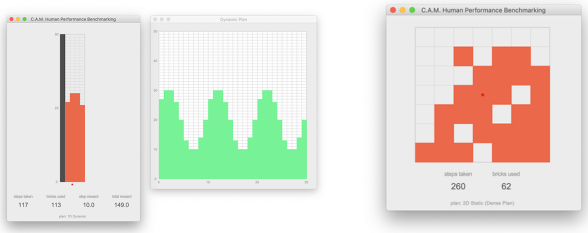
SAC We used the implementation of (Christodoulou, 2019) for SAC in discrete setting which has automatic tuning for entropy hyperparameters and used a learning rate $3\text{e}{-4}$ for target networks for most plans. We use ReLU activation functions for the hidden layers and Softmax for the final layer of the policy network. We use interpolation factor τ=$5\text{e}{-3}$ for target networks and the start steps before running the real policy is 400 with mini batch size 64. We search the main hyperparameters based on 1D static case and we used different network architectures through a random search approach for relatively complex 2D and 3D cases. For 1D plans, we use 2 hidden layers with 64 nodes each for both actor and critic networks. For 2D plans, we use 3 hidden layers with 512 nodes each for the dynamic plan and 3 hidden layers containing [64, 128, 64] nodes for the static plan. 4 layers network architecture containing [64,128,256,128,64] nodes are applied in the 3D cases.

Human For assessing human performance, we made a simple GUI game (Figure 6) that allows a human player to attempt our tasks in identical environments with the same limitations of partial-observability and step size uncertainty. They act as the agent, using the ARROW keys to maneuver,

and SPACE to drop a brick. In static environments, human players are required to complete the task without access to the ground-truth plan, while in dynamic environments, they can reference the current ground truth plan. Additionally, players can toggle between training or evaluation mode. In training mode, they can view per-step reward as well as their cumulative reward over the episode, whereas in evaluation mode, they can only see the number of bricks used and steps taken. For each episode played, players reported their episode-IoU.

5 EXPERIMENTS

5.1 BENCHMARK RESULTS



(a) Training (1D). (b) Evaluation (2D).

Figure 6: Game GUI for measuring human performance. (a) 1D Dynamic plan: user can see the ground truth plan in dynamic environments. (b) 2D Static plan in *evaluation mode*: only step and brick count are shown, while rewards are hidden.

In this section, we presents testing results of all the baselines on our mobile construction task. We use the IoU score as our evaluation criteria which is measured between the terminate grid world state G^T and the ground-truth plan P. For the static plan tasks in 1D/2D/3D, we test the trained agent for 500 times for each task and record the average and min IoU score among 500 tests. For the dynamic plan tasks, the ground-truth plan is randomly generated during the training phase. In order to make our test results comparable among all the baselines, we define 10 groups of coefficients a, b, and c for 1D dynamic and 10 groups of triangles for 2D/3D dynamic task. All algorithms are tested on each group of plans for 200 times and average and min IoU score among 2,000 tests are recorded.

Figure 7 summarizes the quantitative results of all baselines on mobile construction tasks. In general, we see that the performance of each methods drops as the dimension of the environment increases. As mentioned previously, in the 3D environment, the volume of the agent and obstacle is considered, which makes the tasks substantially more difficult. From the results, most of the baselines perform rather poorly on the 3D tasks and do not seem to be able to learn any useful policies in the 3D dynamic plan tasks. This suggests that the 3D task is especially challenging for SOTA RL algorithms. We did an ablation study in Section 5.2 to explore the influence of obstacles on performance. From the results, we can also see that the robot have more difficulties to perform on dynamic plan tasks than on static plans. We posit that the agent lacks the capacity to learn dynamic inputs. DRQN is expected to work best in most of the tasks because it is able to learn more from a long period of history. Interestingly, the HER does not appear to help noticeably for our tasks.

1D tasks: DRQN has the best performance on both static (IoU 0.868) and dynamic (IoU 0.834) tasks (see Appendix Table 2). For the Gaussian function curve, only DRQN and DRQN+Hindsight could learn a general shape. Other algorithms only managed to build a flatten shape. Meanwhile, the DRQN outperforms other approaches and could build all the three peaks of the Sin curve. (See first two columns of Figure 8).

2D tasks: DRQN also achieves the highest IoU (static 0.772 and dynamic 0.537) among all baselines in the static plan tasks. Rainbow gets the best score for the dynamic plan tasks. However, the scores drop dramatically, even close to 0, from static to dynamic plans, which indicates that 2D dynamic plans become much harder for all methods.

3D tasks: we have similarly bad performance in both static and dynamic plan tasks. From the last two columns of Figure 8, we can see that most of the methods can learn to only move in the plane without building any bricks. We hypothesize that the agent learns to only move instead of build bricks in order not to obstruct itself.

Human baseline: we collected 30 groups of human test data, totaling 490 episodes played, and report the average and min IoU showed in the Figure 7. From the feedback, we found that human could learn effective policies (building landmarks to help localization) very efficiently in at most a few hours, which is much less than training an RL model. Interestingly, human performed much better on 2D dynamic plan tasks.

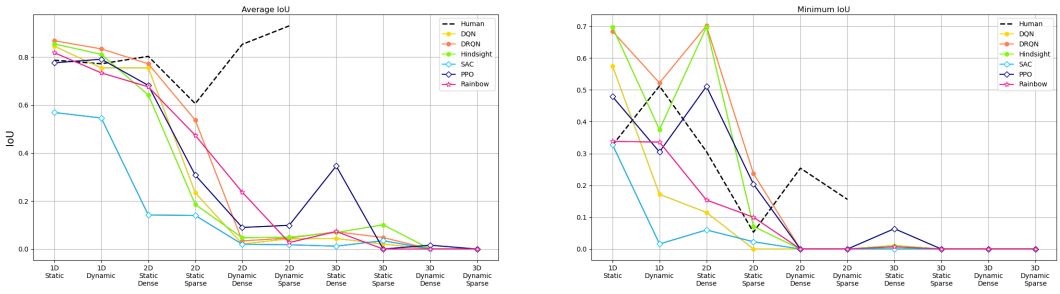


Figure 7: Benchmark results for all the deep RL and the human baselines.

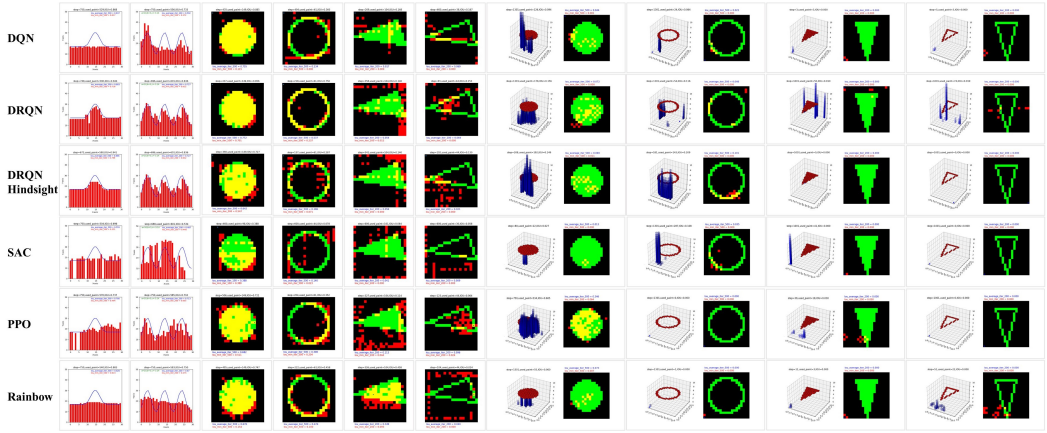


Figure 8: Best testing results of all tasks (the same order as in Figure 7).

Shape	IoU	2D				3D	
		DRQN	+GPS(↑)	-Uncertainty(↑)	+Obstacle(↓)	**DRQN**	-Obstacle(↑)
Dense	Avg	**0.772**	+0.079	+0.073	-0.288	**0.072**	+0.727
	Min	**0.701**	+0.050	+0.144	-0.568	**0**	+0.714
Sparse	Avg	**0.537**	+0.403	+0.239	-0.305	**0.048**	+0.145
	Min	**0.237**	+0.149	+0.539	-0.200	**0**	0

Table 1: Quantitative ablation study results. ↑/↓: expecting better/worse performance.

5.2 Ablation Study

We performed ablation studies to comprehensively analyze the reasons associated to the poor baseline performances in 2D and 3D tasks. We identify three potential challenges: the obstacles in the 3D environments, the lack of localization information, and the step size d uncertainty. We use the DRQN results as the basis of this ablation study and all the ablation experiments were conducted using the same setup described in Section 4. We also use the same test criteria as before.

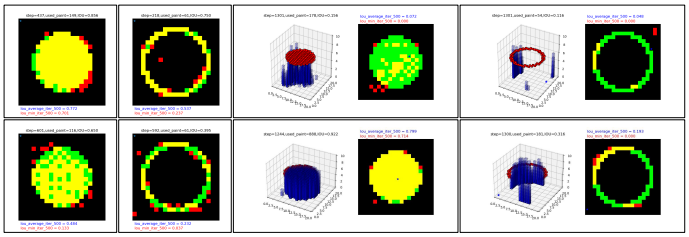
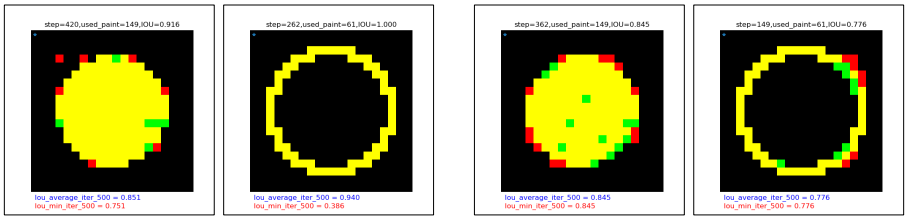


Figure 9: Influences of obstacles. Top rows shows the baseline results on 2D and 3D static tasks. The first two images of bottom row show results of adding obstacle to 2D world. Compared with baseline, the performance drops notably. The last two images of bottom row show results of removing obstacle from 3D world.

Obstacles. To explore the influence of the obstacles, we did two experiments: adding the obstacle mechanism in a 2D task and removing it from the 3D (Figure 9). As showed in Table 1, the



(a) Add GPS. (b) Fixed step size.

Figure 10: Influence of localization and environment uncertainty. Either adding GPS in (a) or removing step size uncertainty in (b) significantly improves DRQN on sparse plan tasks compared with the first two columns of the top row of Figure 9.

performance of the 3D task increases significantly from 0.072 to 0.727 for dense plans. Similarly, when we add obstacles into 2D, the IoU score drops more than 40%. These results suggest that the presence of obstacles is an important reason causing the poor performances on 3D tasks.

Localization. Our mobile construction tasks are intrinsically partially observable and good localization of the robot in its environment becomes a key challenge. The agents are expected to learn implicitly how to do localization with a limited sensing region (partial observation o) of the environment. We hypothesize here that the observed low baseline performances in 2D and 3D worlds can further be explained by the fact that the agents lack the capability of localizing precisely. To test this, we added ground truth location state information $l(x, y)$ into the feature vector as described in Section 4 for DRQN on 2D static tasks. From Table 1, we can see that with the help of position information, the performance increases, especially in 2D sparse task (see Figure 10a).

Environment uncertainty. Besides the limited sensing range, the random step size could be another reason of poor performance of baselines. Therefore, we conduct an experiment where the step size uncertainty is removed. From Table 1, we can see that the IoU increases more than 40%, compared to the baseline on sparse plan (see Figure 10b).

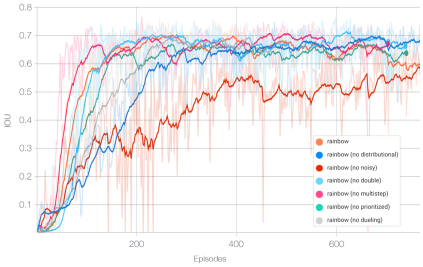


Figure 11: Training history of IoU: the full Rainbow algorithm does not always outperform its pruned configurations.

Rainbow method ablation study. When we applied the complete Rainbow algorithm on top of a base DQN algorithm, we observed an overall decrease in the agent's performance. To better understand the effect of each individual extension with respect to the whole, we conducted six separate runs: in each, we removed one extension from the complete Rainbow algorithm, similar to (Hessel et al., 2017). Figure 11 displays the IoU running average over training in the 2D Static environment for each of these pruned configurations. While the complete Rainbow algorithm initially trains *faster* than all but one of the pruned configurations, over a longer horizon, the top testing performance converges to nearly the same across all configurations. In the case of distributional learning, multi-step learning, and double DQN, removing each individually actually improved top test performance, from 0.676 to 0.685, 0.701, and 0.714 respectively. While this study is not exhaustive, it suggests that the Rainbow algorithm does not easily generalize from the Arcade Learning Environment (Bellemare et al., 2013) (which it was designed for) to our specific tasks.

6 Conclusion and Future Work

We proposed a suite of mobile construction tasks on which we benchmarked the performance of SOTA deep RL algorithms. Our results show that simultaneously localizing a robot while changing the environment is currently very difficult for model-free RL, and we believe solutions with explicit localization and planning modules that are specific to these tasks could be more efficient. Meanwhile, we also plan to further extend our mobile construction task suite with more interesting features such as allowing plan changes within an episode, explicitly adding a landmark placement action, physics-based simulation in continuous worlds, and multi-agent mobile construction.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in neural information processing systems*, pp. 5048–5058, 2017. 2, 5

Hadi Ardiny, Stefan Witwicki, and Francesco Mondada. Construction automation with autonomous mobile robots: A review. In *2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM)*, pp. 418–424. IEEE, 2015. 2

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL http://dx.doi.org/10.1613/jair.3912. 8

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *CoRR*, abs/1707.06887, 2017. URL http://arxiv.org/abs/1707.06887. 2

Jonas Buchli, Markus Giftthaler, Nitish Kumar, Manuel Lussi, Timothy Sandy, Kathrin Dörfler, and Norman Hack. Digital in situ fabrication-challenges and opportunities for robotic in situ fabrication in architecture, construction, and beyond. *Cement and Concrete Research*, 112:66–75, 2018. 2

Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019. 2, 5

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016. 2

Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, pp. 767–782, 2018. 2

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Rémi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. *CoRR*, abs/1706.10295, 2017. URL http://arxiv.org/abs/1706.10295. 2, 5

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2616–2625, 2017. 2

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML*, pp. 1856–1865, 2018. 2

Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems 23*, pp. 2613–2621. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/3964-double-q-learning.pdf. 2

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015. 2, 5

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning, 2017. 2, 5, 8

Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018. 5

Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019. 2

Sasa Jokic, Petr Novikov, Stuart Maggs, Dori Sadan, Shihui Jin, and Cristina Nan. Robotic positioning device for three-dimensional printing. *arXiv preprint arXiv:1406.3400*, 2014. 2

Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008. 2

Yann Labbé, Sergey Zagoruyko, Igor Kalevatykh, Ivan Laptev, Justin Carpentier, Mathieu Aubry, and Josef Sivic. Monte-carlo tree search for efficient visually guided rearrangement planning. *IEEE Robotics and Automation Letters*, 5(2):3715–3722, 2020. 2

Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pp. 2140–2146, 2017. 2

Chengshu Li, Fei Xia, Roberto Martín-Martín, and Silvio Savarese. Hrl4in: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Conference on Robot Learning*, pp. 603–616, 2020. 2

Lucas Galvan Marques, Robert Austin Williams, and Wenchao Zhou. A mobile 3d printer for cooperative 3d printing. In *Proceedings of the 28th Annual International Solid Freeform Fabrication Symposium*. University of Texas, 2017. 2

Nathan Melenbrink, Katja Rinderspacher, Achim Menges, and Justin Werfel. Autonomous anchoring for robotic construction. *Automation in Construction*, 120:103391, 2020. 2

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 1, 2, 5

Kaichun Mo, Haoxiang Li, Zhe Lin, and Joon-Young Lee. The adobeindoornav dataset: Towards deep reinforcement learning based real-world indoor robot visual navigation. *arXiv preprint arXiv:1802.08824*, 2018. 2

Cristina Nan. A new machinecraft: A critical evaluation of architectural robots. In *International Conference on Computer-Aided Architectural Design Futures*, pp. 422–438. Springer, 2015. 2

Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018. 1, 2

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. 2

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pp. 2164–2172, 2010. 2

Richard Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3: 9–44, 08 1988. doi: 10.1007/BF00115009. 2

Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016. 2

Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Wong Lawson LS, and Stefanie Tellex. Multi-object search using object-oriented pomdps. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7194–7200. IEEE, 2019. 2

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2016. 2

Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014. 2

Wikimedia Commons. Section of above ground structure of odontotermes sp. https://commons.wikimedia.org/wiki/File:Section_of_Termite_mound.jpg, 2018. 1

Brian Yang, Jesse Zhang, Vitchyr Pong, Sergey Levine, and Dinesh Jayaraman. Replab: A reproducible low-cost arm benchmark platform for robotic learning. *arXiv preprint arXiv:1905.07447*, 2019. 2

Fanyu Zeng, Chen Wang, and Shuzhi Sam Ge. A survey on visual navigation for artificial agents with deep reinforcement learning. *IEEE Access*, 8:135426–135442, 2020. 2

Xu Zhang, Mingyang Li, Jian Hui Lim, Yiwei Weng, Yi Wei Daniel Tay, Hung Pham, and Quang-Cuong Pham. Large-scale 3d printing by a team of mobile robots. *Automation in Construction*, 95:98–106, 2018. 2

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3357–3364. IEEE, 2017. 2

A APPENDIX

	IoU	1D		2D				3D			
		Static	Dynamic	Static		Dynamic		Static		Dynamic	
				Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse
Human	Avg	0.787	0.772	**0.803**	**0.606**	**0.853**	**0.93**	–	–	–	–
	Min	0.326	0.523	0.306	0.053	0.254	0.156	–	–	–	–
DQN	Avg	0.845	0.755	0.755	0.234	0.021	0.043	0.044	0.021		
	Min	0.575	0.172	0.115	0	0	0	0.001	0		
DRQN	Avg	**0.868**	**0.834**	0.772	0.537	0.034	0.046	0.072	0.048		
	Min	0.684	0.522	0.701	0.237	0	0	0	0		
DRQN+Hindsight	Avg	0.855	0.811	0.642	0.186	0.047	0.049	0.069	**0.101**		
	Min	0.697	0.375	0.697	0.071	0	0	0.011	0		
SAC	Avg	0.569	0.546	0.142	0.14	0.02	0.018	0.012	0.035		
	Min	0.328	0.016	0.06	0.023	0	0	0	0		
PPO	Avg	0.776	0.791	0.682	0.308	0.09	0.099	**0.346**		0.016	
	Min	0.479	0.305	0.511	0.204	0	0	0.064		0	
Rainbow	Avg	0.818	0.734	0.676	0.474	0.237	0.027	0.073			
	Min	0.338	0.336	0.154	0.1	0	0	0.007			

Table 2: Benchmark quantitative results. Empty cells indicate the agents failed at the these task without learning any control policy.