Heuristic Reward Driven Distributed Athlete Trainer

Bao Qingquan [*1] **Cao Jiahang** [*1] **Zheng Yu** [*1]

Abstract

Regular Multi-agent Reinforcement Learning methods are not likely to perform well on environments with sparse rewards or integrated task settings. We expected to explore a heuristic reward function and high-capacity data replay framework to tackle this problem. We mainly study on Olympics Running environment, in which the main challenge is how to perform well in different environments with little positive feedback with a single policy. Using models based on PPO, we construct a parallel framework with efficient data sampling. To deal with the sparse reward, reward-shaping approaches inspired by robotics path planning algorithms called AStar trace potential is utilized. In refining the learning of the reward, we develop an end-to-end routine with the mechanism of curiosity, which performs well on the framework below. Self-play training is also utilized in training routine to achieve better leaning on multi-agent task. Our source code has been released at `https://github.com/COMoER/HRDDAT`

1. Introduction

How to train Reinforcement Learning (RL) model well on an environment with sparse rewards and various type of settings.

Now consider a real-life example: a coach is training a running athlete with poor eyesight, demanding him to keep running on the right path and overcome various of obstacles or opponents, but would only offer huge reward at-one-time when the apprentice reaches the goal. This approach is lack of heuristics and may make the athlete aimless, confused and easy to give up, so traditional training methods often fail to tackle with these problems.

The sense upon may seem weird to occur in reality, but it well describes many stimulative environments of RL applications like game AI and autonomous driving, where the coach is replaced by reinforcement learning model and the athlete and opponent is acted by RL agents. The reward is quantified by value with sparse distribution and the running of athletes is controlled by high-order physical engine systems.

Parallel sampling on multi-task environments is a good way to make agent easier to learn from exploring the environment. Model for single problem is often hard to generalize on various tasks, things would be better if coaches expert at each running maps gather in a staff. Inspired by MAPPO (Yu et al.), we develop a parallel framework of PPO-based model to make use of *transaction* package sampled by *multiply process* and get experiences in multiply tasks.

Besides, intermittent reward may inform the athlete whether he is on the right path towards the goal. A* (Hart et al., 1968) is a heuristic path planning algorithm with simplified search space, from which we construct a task-wise cost map and a simple heuristic reward to encourage stepping forward on the shortest path.

Moreover, we utilized a curiosity-driven approach to make the agent learn a reward end-to-end from its changing observation, where *curiosity mechanism* (Burda et al., 2019) can prevent the agent from trapping in a corner. This model perform the best of our models and shows great stability.

Last but equally important, considering the opponent as a competitive agent, instead of using random-action opponent, self-play (Smith et al., 2021) is also utilized to encourage the agent to run more poisedly than his former self in the policy pool throughout the training.

2. Preliminary

2.1. PPO

Proximal Policy Optimization (PPO) (Schulman & et al, 2017) , an on-policy algorithm, performs well on the basic task in our problem setting almost without hyper-parameters tuning, thus we choose it as our baseline algorithm. And we use the common training strategy of PPO algorithm like Generalized Advantage Estimator (GAE)

[*]Equal contribution [1]SEIEE , Shanghai Jiao Tong University, Shanghai, China. Correspondence to: Wen Ying <ying.wen@sjtu.edu.cn>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

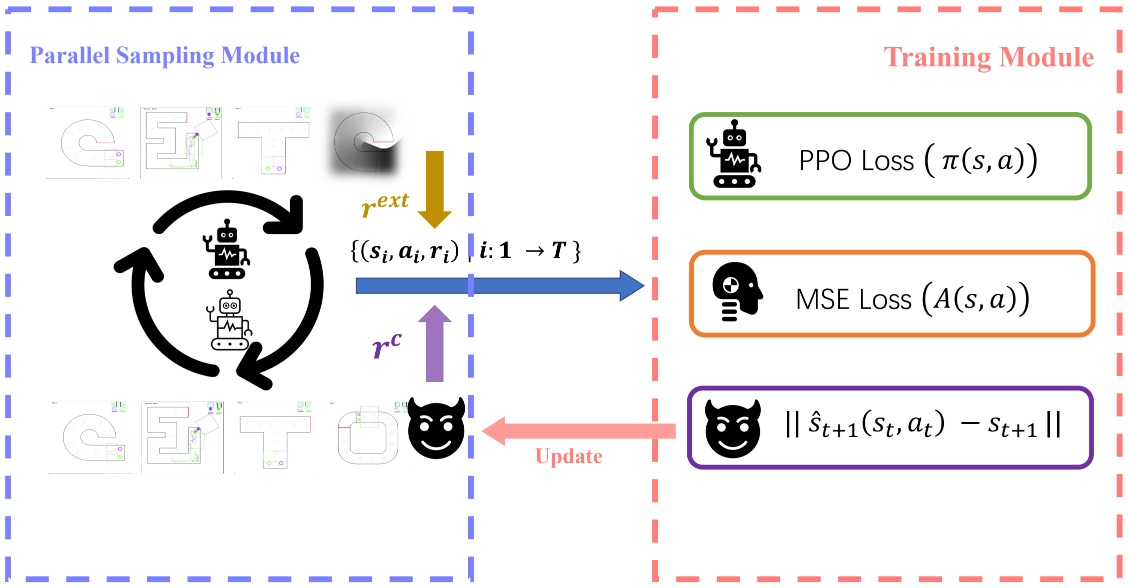


Figure 1. Routine of the parallel framework with curiosity, AStar trace potential reward and self-play training.

(Schulman et al., 2015) in our training to make the training more robust.

2.2. Path Planning Algorithm

There is a basic idea that a heuristic reward should guide the agent towards a shortest path on the map. A* is a popular-used path planning algorithm on connected graph based on map searching. A cost evaluation function is used in this algorithm: cost of point v on a grid map is calculated by the form of $f(v) = g(v) + h(v)$, where $g(v)$ is the calculated number of steps from the start point, and $h(v)$ is the evaluated steps to the goal, which is estimated by Euclidean distance in the coordinate system.

From the core code of the environment we can get the coordinate of the start point and take the middle of the finishing line as the goal point. Map searching from start point to goal point would diffuse towards the shortest path, as next neighbor of points to update the evaluation function is set around the point where value of f increases.

3. Methodology

3.1. Rule-based Reward dependent to A* trace

Using A* we find both each of the shortest path on each map starting on each of the two optional initial agent place. To simplify the calculation, the maps are firstly divided into a coarse-gird level. Then dilation is applied on the wall to avoid the agent of being too close to the obstacles and a small erosion near the start point in avoid of being blocked.

Consider a path $((x_0, y_0), (x_1, y_1), \cdots, (x_{T-1}, y_{T-1}))$, we define a rule-based reward based from a 'trace potential' φ denoting on which step of the path the agent is. At the step n the location of the agent changes from (x_t, y_t) to (x_{t+1}, y_{t+1}). In order to avoid the potential from informing too much importance, a proportion $1/G$ is multiplied on the item of potential.

$$r_t^{as} = \begin{cases} +100, & t = t_{max} \; agent \; wins \\ -100, & t = t_{max}, \; agent \; loses \\ (\varphi_{t+1} - \varphi_t)/G - 1, & otherwise \end{cases} \tag{1}$$
$$where \; \varphi_t = \arg\min_i \left[(x_t - x_i)^2 + (y_t - y_i)^2 \right]$$

3.2. Parallel Framework

To improve the sampling efficiency, we design a parallel framework for training. This framework will create several worker processes to interact with the environment and thus make the amount of sampled data for one time updating be the number of parallel worker processes times of the amount in single worker framework. In our problem setting, this means collecting the experiences of more tasks and perspectives for single updating, which will greatly increase the speed of convergence and improve both generalization performance and absolute performance.

3.3. Curiosity: inner reward

To tackle the sparse reward, we introduce curiosity mechanism, a kind of inner rewards, to encourage the exploration of the agent. We follow the formulation in (Burda et al., 2019). At timestamp t the agent gets the observation x_t, takes the action a_t, and gets the next observation x_{t+1}. Curiosity rewards the exploration, i.e., how informative such transition (x_t, a_t, x_{t+1}) is. To specify, we use the following two networks: (1) an embedding network turning observation x into a compact, sufficient and stable representations $\phi(x)$; (2) a forward dynamic network to predict the *suprisal* of the next observation, i.e., $p(\phi(x_{t+1})|x_t, a_t)$. Finally, we get the inner reward $r_t^c = -\log p(\phi(x_{t+1})|x_t, a_t)$.

In detail, we use *random features* as the function $\phi(\cdot)$, which shows great performance in (Burda et al., 2019) and use mean squared-error(MSE) to model the inner reward, i.e., $r_t^c = \|f(x_t, a_t) - \phi(x_{t+1})\|_2^2$, where $f(\cdot, \cdot)$ is the forward dynamic network modeled by neural networks. With such settings, we let the final reward $r_t = \alpha^{ext} r_t^{ext} + \alpha^c \cdot r_t^c$, where α^{ext}, α^c are hyper parameters controlling the weight ratio between the external and inner reward, to avoid the curiosity reward converging to stable value while training which will cause the exploration losing direction.

While training, the MSE mentioned above will be added to loss function to minimize it in order to make the inner reward of explored transition decrease and thus force the agent explore the new transition and try to get larger total reward.

3.4. Self-play

In this zero-sum game, it's natural to introduce self-play when training our agent. Instead of using a specific agent, e.g. a random agent, self-play sets the opponent sharing the same strategy when training an agent.

4. Experiments

4.1. Olympics Running Environment

Olympics Running is an typical partial observation multi-agents environment. The task is to control one agent with fixed initial location to compete with the other agent in the map which is randomly picked from the different 11 game maps and target is to reach the goal line faster than the opponent. The game will end when anyone reaches the line or the game has been played for 500 steps. This is a typical partial observation problem, for both agent could only observe the objects in the front 25×25 rectangle area. The action is two dimension, the amount and angle of the force added on the agent and they are in order bounded by the range between -100 and 200 and the range between -30 and 30. Instead of directly predicting the continuous action, discrete action space is designed to approximate the continuous space, which divided each of the range of force amount and the range of force angle into 6 bins and the discrete action space is exactly the Cartesian product of the two set of bins whose dimension is 36.

4.2. Basic Network Architecture

Multi-Layer Perceptron(MLP) is treated as the architecture of the deep network to fit the policy and value function with Adam as the optimizer. The 25x25 observation will be flattened as the input of both policy and critic network.The output of policy network is the distribution of the discrete actions defined in 4.1 and the output of value network is a scalar of estimated value.

4.3. Basic Training Setting

The loss function of our network is the weighted sum of value loss, policy loss and the policy entropy loss. Value loss is the MSE loss between the GAE computed advantage value and the value estimated by the critic network, the policy loss is as Equation 2, the updating equation of PPO, and the policy entropy loss is the negative of the entropy of the discrete action distribution predicted by the policy network.

$$\begin{aligned} J(\pi_\theta) = &- \mathbb{E}_{s \sim d^{\pi_{\theta_k}}} \mathbb{E}_{a \sim \pi_{\theta_k}(\cdot|s)} \\ &\Bigg[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \right. \\ &\left. \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_{\theta_k}}(s,a) \right) \Bigg] \end{aligned} \tag{2}$$

In order to smooth the scale difference of data, we suppose to perform the following normalization in training like common PPO training setting. We conduct each of the following experiments with/without the following normalization to verify the performance of normalization in each experiment setting.

- **Data Normalization**: We run two random agent in all kinds of maps for 200 rounds of games, then calculate the mean and the standard deviation in each pixel, and use these to normalize the observations when training.
- **Reward Normalization**: We divide the rewards by a running estimate of the standard deviation of the sum of discounted rewards, because the rewards are sparse, non-stationary, and diverse. By doing this, we can stabilize our training process.
- **Advantage Normalization**: We normalize the advantages in a batch to follow a standard normal distribution.

In training, the sampled transitions will be shuffled into several mini batches and update the parameters of the network in order with all the mini batches repeatly for several times.

The common hyper parameter setting is provided in the appendix.

The table below shows evaluation results of different models. Note that:
(1) most of the model structures are tested both with and without normalizations in Sec.4.2, models with # in the name means that the better of the two is shown.
(2) Win rates are shown as (our model : opponent). w_r indicates result v.s. **r**andom agent and w_b for that v.s. Jidi **b**aseline.

Table 1. Performances of models on single and parallel framework. -sf means maps are shuffled during training.

Model	Data norm	w_r	w_b
Single	False	28 : 10	23 : 33
Single-sf	False	28 : 13	7 : 46
Single-sf	True	8 : 12	5 : 47
Parallel-sf#	**False**	**43 : 7**	**26 : 35**

Table 2. Performance of models on **parallel** framework with A*, curiosity and without reward shaping. Shuffle map is used in default.

Model	Data norm	w_r	w_b
AStar#	False	14 : 22	1 : 54
Single-AStar#	False	18 : 24	8 : 62
Curiosity	**True**	**59 : 8**	**43 : 27**
Curiosity	False	30 : 14	7 : 53
Curiosity-AStar#	False	47 : 6	23 : 36
Curiosity-self-play#	True	47 : 8	36 : 22

4.4. Baselines and Benchmarks

Besides calculation of rate during training, we also supply simulative competition results given by an evaluation script based on Jidi official source code. The opponents in the script include a random agent and a baseline agent and a PPO-trained agent from Jidi. 100 episodes of the game on mixed maps are rendered and the win rate of our models versus both of the opponent is shown in Table 1 and 2.

4.5. Experiments of Training Framework

In training, we first design experiments to figure out the influence on the generalization performance of the trained agent by sampling data from either the fixed map or randomly picked map each episode (shuffle map).

In addition, we compare the difference of performance and efficiency between the training single sampling process and parallel sampling process based on the shuffle map setting. Shuffle map setting enables the sampled data from multiply tasks and to sample the data in multiply perspective, we make half of the rollout environment control the first player and the other half control the second player.

From Table 1 we can see that

- Though single process sampling converges well on single task (task 1), the trained policy is not able to generalized on other tasks.
- When it comes to training on shuffled map, training would fail due to the sophisticated task and so do on evaluation.

Luckily, parallel framework helps us tackle this problem by improve the efficiency of sampling and data utilizing, leading to a great improvement on shuffled map training and evaluation.

To show the time efficiency of sampling, we measure the average time cost of one step (include sampling and model updating) under same hardware conditions using different number of rollouts. From Figure 2 and Table 7 in the appendices we know the increase of number of sampling process dose not linearly increase the time cost. e.g. 16 rollouts only consume 56.3% more time than 4 rollouts but get 300% more data. Parallel framework brings a greatly high efficiency gain with little effect to the convergence of the model.

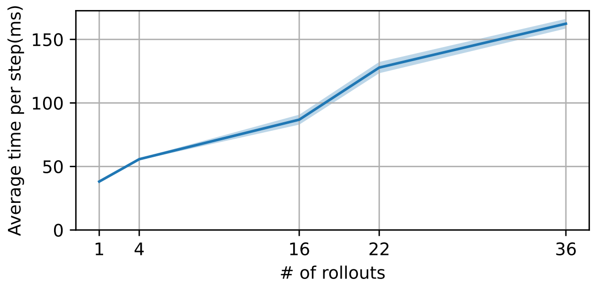


Figure 2. Increasing of time cost by number of rollouts.

4.6. Experiments of Reward Shaping

The naive reward shaping is to give each step reward -1 and at the end of the game, if no winner, still give reward -1 but if anyone wins the game, give a greatly positive reward 100 to the winner and a greatly negative reward -100 to the loser. We conduct the following experiences to compare with the naive reward shaping which is exactly sparse.

4.6.1. Rule-based Reward dependent to $A^\star$ trace

Using the rule-based reward shaping method, $A^\star$ trace potential we try to construct a denser space of reward to reveal the right path to the agent.

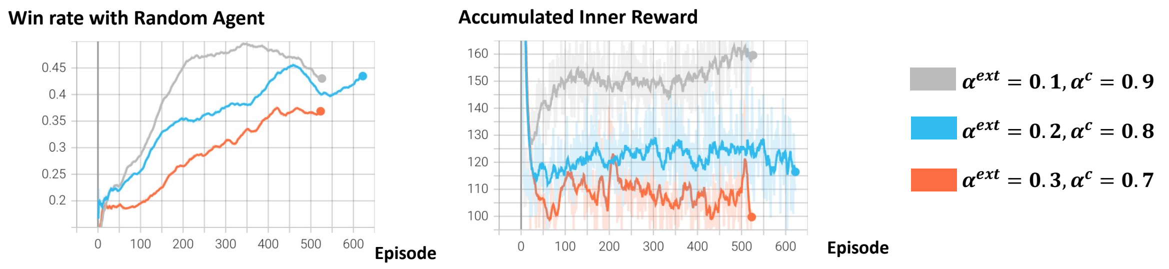


Figure 3. Performance of models with curiosity on different α^{ext}, α^{c}, statistics are calculated till convergence. The ablation study indicates that the best choice is 0.1 for the external reward and 0.9 for the curiosity reward.

Surprisingly, the performance of models with $A^{\star}$ stay poor on either single or parallel frameworks. The training tends to be unstable and may convergent to a bad policy. Considering the routine of generating $A^{\star}$ reward, we consequent this phenomena to the coarse-grained resolution, which makes the agent hard to make progress on the potential as we demanded implicitly. This conflict would further confuse the agent in the hard environment.

However, compromise has made due to costmaps with high resolution would cost unaffordable memory especially under implement of parallel frameworks. This analysis inform us that hand-made reward-shaping approaches would be restricted to practical arguments, and raises the importance of leaning a fair reward end-to-end.

4.6.2. Curiosity Driven Inner Reward

We in advance explore the effect of using inner reward which is driven by curiosity instead of pre-designed reward. As shown in 3.3, we define the reward of agent is the weighted sum of external reward, and conduct the ablation study of the weights of the two rewards. And the loss of the experiments which uses curiosity driven inner reward will be added the extra curiosity loss of the data in the mini-batch. Without loss of universality, we seperately use deafult and $A^{\star}$ reward shapping as inner reward.

As figure 3 shown, the accumulated inner reward, which is the sum of the inner reward within an episode, will descend first and then increase to the convergence point. This indicates that introducing curiosity driven inner reward will exactly encourage the exploration. At first, the model will badly predict the next observation and give big inner reward for explore new area, but minimizing curiosity loss will make the explored transition giving few inner reward which forces the agent which wants to maximize the reward trying to explore the new transition. This is exactly a max-min game and finally converges to equilibrium and then the addition of external reward will take an important role of the further training of agent.

4.6.3. Experiments of the self-play and the combination of $A^{\star}$ and curiosity

On one hand, we study the performance of self-play combined with curiosity in our problem setting. The result, however, shows self-play does not necessarily improve the final competitive game. We doubt that the current RL training paradigm and thereby agent policies are not good enough to introduce the self-play, and self-play misguides our agent. On the other hand, we add the external handcraft $A^{\star}$-inspired rewards to the inner curiosity-driven rewards, but the result is also no better than the original curiosity version.

5. Conclusion

In this paper, we try to overcome the reward sparsity in the diverse MARL environment. We show that the parallel framework can effectively increase the data sampling efficiency and stabilize the PPO updating process, while curiosity-driven inner rewards alleviate the sparsity and can take full use of all successful and failing data. Besides, we also try other handcraft reward shaping like $A^{\star}$ trace potential, however, it shows no improvement partly due to the restrictions to hyper parameters and space complexity. In the future work, how to automate the weight ratio between the external reward and the inner reward, and how to build an online curiosity mechanism such that many agents can explore more diverse interaction situations can be carefully considered.

Acknowledgements

We sincerely appreciate the assistance from Prof.Ying Wen and TA Xihuai Wang. They provide the training framework code for the baseline algorithm and the guidance during our research, which help us to make progress in our experiments.

References

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

Schulman, J. and et al, F. W. Proximal policy optimization algorithms, 2017.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Smith, M. O., Anthony, T., and Wellman, M. P. Iterative empirical game solving via single policy best response. 2021. doi: 10.48550/ARXIV.2106.01901. URL `https://arxiv.org/abs/2106.01901`.

Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. The Surprising Effectiveness of MAPPO in Cooperative, Multi-Agent Games. pp. 25.

Appendix

A. Hyper Parameter Settings

Table 3. Common Hyper Parameters Setting for All Experiments

Hyper-parameter	Value
GAE lambda	0.95
PPO clip epsilon	0.2
maximum of gradient norm	0.5
gamma	0.99
learning rate of policy network	0.0001
learning rate of critic network	0.0008
policy loss weight	1
critic loss weight	1
entropy loss weight	0.001
PPO updating times	5

Table 4. Hyper Parameters Setting for the Experiment of Parallel Framework

Hyper-parameter	Value
number of rollouts	22
shuffle map	true
normalizations	[true,false]

Table 5. Hyper Parameters Setting in Generating A* Costmap and Calculating A*-based Reward.

Hyper-parameter	Value
grid resolution	200
wall dilation (pixel)	12
start point erosion (pixel)	8
max step of trace (G)	5

Table 6. Hyper Parameters Setting in Generating A* Costmap and Calculating A*-based Reward.

Hyper-parameter	Value
curiosity loss weight	1
learning rate of curiosity network	0.0005
curiosity reward weight	[0.9,0.8,0.7]
external reward weight	[0.1,0.2,0.3]

B. Statictics of parallel time cost

Table 7. Time per step under different number of rollouts.

# of Rollouts	Time per step (ms)
1(Single)	38.13 ± 0.01
4	55.73 ± 0.70
16	86.84 ± 3.91
22	127.84 ± 4.43
36	162.36 ± 3.81

C. Visualization of $A^\star$ costmap

In figure 4, We have visualized the $A^\star$ trace and the corresponding costmap in Olympics Running map 2, to demostrate the rule-based reward dependent to $A^\star$ trace.

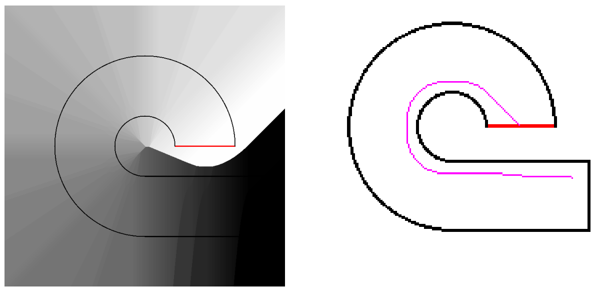
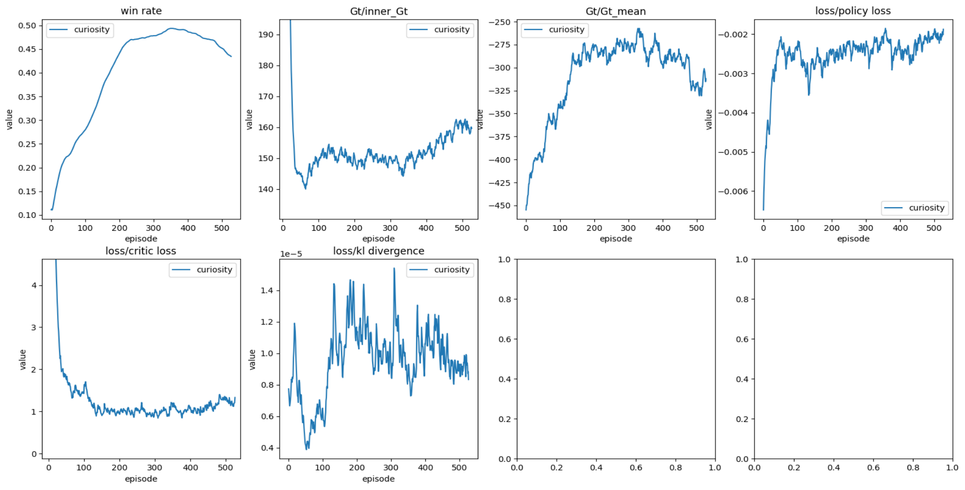


Figure 4. the $A^\star trace$ and the corresponding costmap of map2, the higher pixel value means the higher potential value

D. Training Details

Figure 5 shows the values of metrics while training in our best model (Curiosity with 0.1 α_{ext} and 0.9 α_c).KL divergence is computed between the before updating policy network output distribution and the after updating policy network output distribution of each mini-batch updating.


win rate
Gt/inner_Gt
Gt/Gt_mean
loss/policy loss
loss/critic loss
loss/kl divergence
curiosity
value
episode
1e−5