Differentiable Learning of Scalable Multi-Agent Navigation Policies

Xiaohan Ye; Zherong Pan; Xifeng Gao; Kui Wu; Bo Ren
Published in: IEEE Robotics and Automation Letters (Volume: 8, Issue: 4, April 2023)

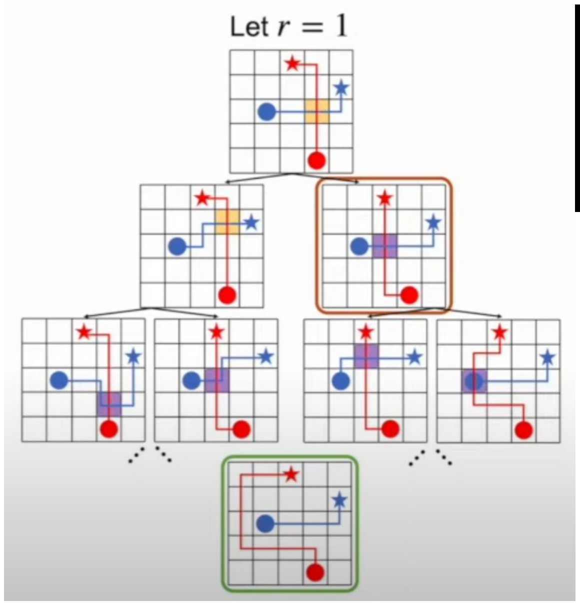
INTRODUCTION

INTRODUCTION

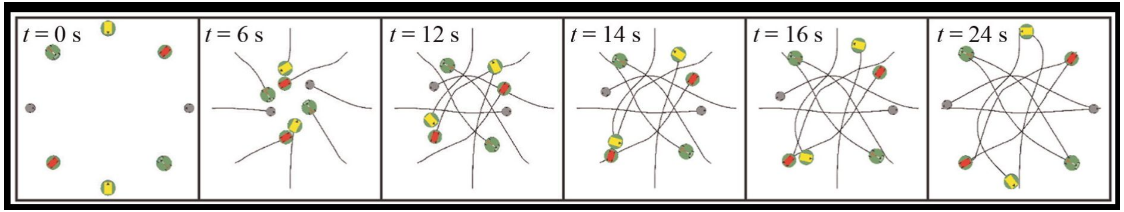
- Multi-agent systems find various robotic applications and there has been an ever-increasing interest in the automatic learning of multi-agent behaviors.
- Complex behaviors can be automatically acquired, e.g., by Multi-Agent Reinforcement Learning (MARL) algorithms, through a gazillion of reward- or curiosity-guided random behavior explorations.
- Specifcally, (MA)RL requires a huge number of environment-interacting experiences, and supervised learning algorithms rely on a dataset of groundtruth control signals.
- We propose the frst differentiable and scalable learning method for collision-free multi-agent navigation policies.

RELATED WORK

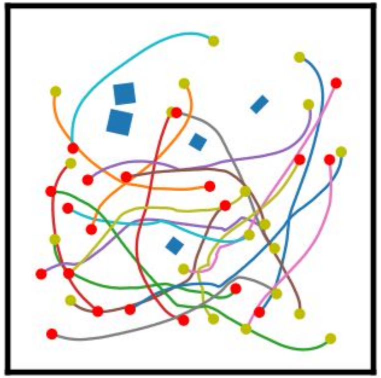
Multi-agent navigation



centralized robot routing algorithm



decentralized real-time navigation algorithms



learning-based semi-centralized navigation policy search (the average RL training cost is around 10 hours on average, as reported by [1])

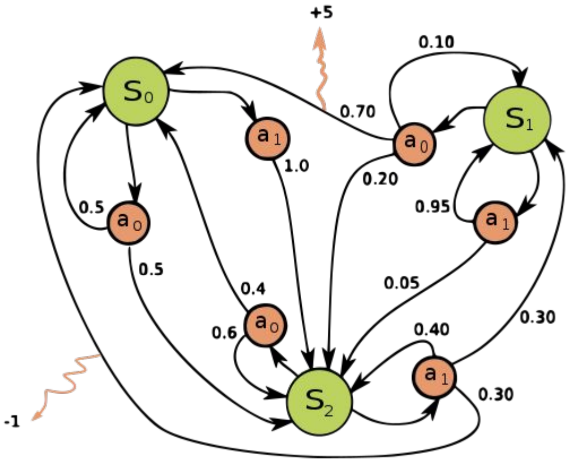
PROBLEM FORMULATION & BACKGROUND

- the same spherical shape with a unifed radius of r
- any pair of two agents do not overlap at any time instance

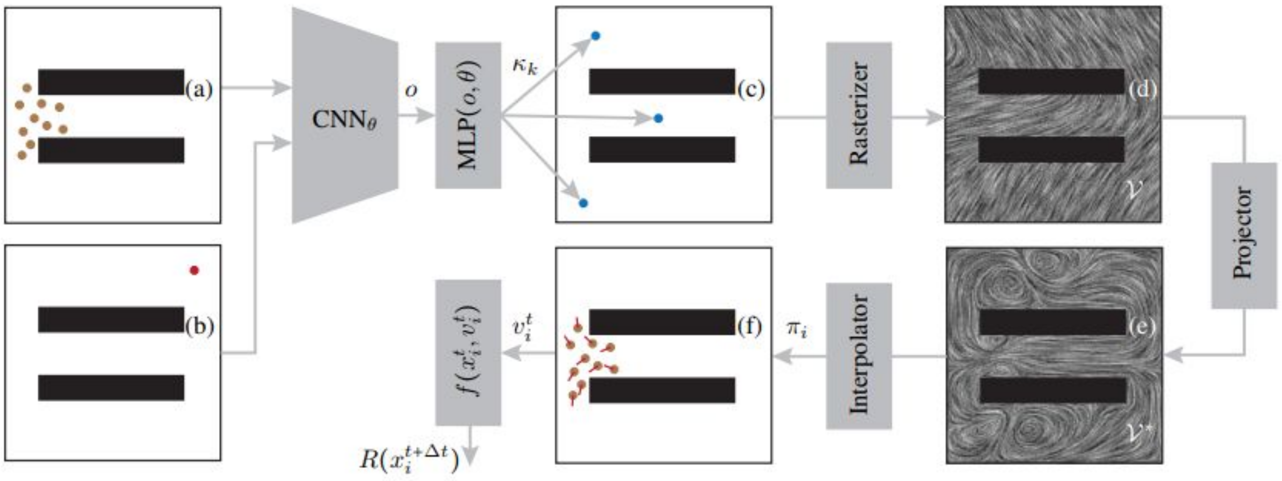
$$\forall \alpha \in [t, t+\Delta t] : \begin{cases} \mathrm{dist}(x_i^\alpha, x_j^\alpha) \geq 2r \\ \mathrm{dist}(x_i^\alpha, o_j) \geq r \end{cases}, \qquad (1)$$

- the following Markov Decision Process (MDP)
- navigation is encoded in a reward function R
- the state transition function f and the policy πi

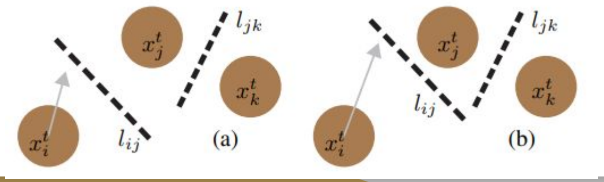
$$\underset{\theta}{\operatorname{argmax}} \mathbb{E}_{\tau_i \sim (f, \pi_i), x_i^0 \sim \mathcal{S}} \left[\sum_{x_i^t \in \tau_i} \gamma^t R(x_i^t) \right], \qquad (2)$$

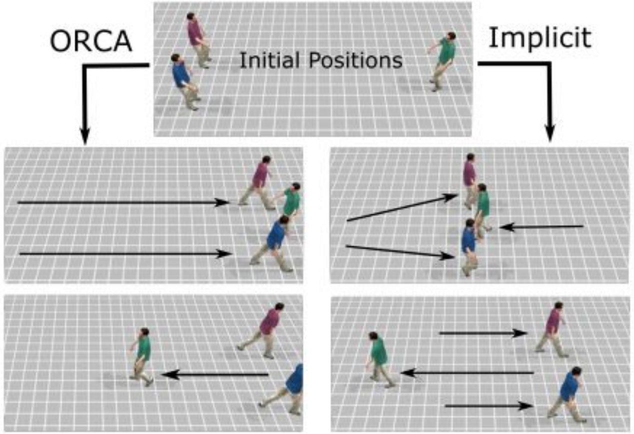


DIFFERENTIABLE & SCALABLE POLICY SEARCH



- Our method consists of novel designs of functions f, R, and πi
- we tried to use an autodifferentiation system [36] to re-implement the ORCA algorithm
- ORCA and its variants confne each agent to independent feasible subdomains so that linear programming problems can be solved separately.
- Although this method signifcantly lowers the computational overhead, it prevents gradient information from propagating to neighboring agents





- we turn to the more recently proposed IC algorithm

$$x_1^{t+\Delta t}, \cdots, x_N^{t+\Delta t} \triangleq \underset{x_1, \cdots, x_N}{\operatorname{argmin}} E(x_i, x_i^t, v_i) \quad (3)$$

$$E(x_i, x_i^t, v_i) \triangleq \sum_i \frac{1}{2\Delta t^2} \|x_i - x_i^t - v_i^t \Delta t\|^2 + \sum_{i \neq j} U(x_i, x_j),$$

where $U(x_i, x_j)$ is a stiff potential function defined as:

$$U(x_i, x_j) \triangleq \frac{1}{\|x_i - x_j\| - 2r}, \quad (4)$$

- should be satisfed at any time instance α ∈ [t, t + Δt], which could be achieved via continuous collision check.

$$U'(x_i, x_j) \triangleq \frac{1}{\underset{\alpha \in [0,1]}{\operatorname{argmin}} \|(1-\alpha)(x_i - x_j) + \alpha(x_i^t - x_j^t)\| - 2r}.$$

- Unfortunately, it is well-known that the distance between line segments is non-differentiable
- we propose an “inconsistent” optimizer
- the line-search algorithm takes care of the entire time period [t, t+Δt] by ensuring that each search step represents a collision-free linear sub-trajectory.
- As a result, the entire trajectory generated by the optimizer is exactly piecewise linear.

$$U(x_i, x_j) = \frac{1}{2}\sum_{l=1}^{\infty} l^3 \max\left[0, \frac{l_0}{l^{\frac{1}{4}}} - (\|x_i - x_j\| - 2r)\right]^2, \quad (5)$$

Kernel-Based Policy Parameterization

- divergence-free constraints can already prevent a considerable portion of local, inter-agent collisions or boundary penetrations.
- Rotating motions are generated by the following kernel:
- Here β and d control the strength of swirl and motion velocity,
-
- $$\kappa(p,\phi) \triangleq d\exp\left(-\alpha\|p-p_0\|^2\right) \quad \phi \triangleq \left(\alpha\, d\, p_0\right). \qquad (6)$$
-
- the accumulated velocity feld V is then rasterized onto a dense grid. To further enforce the divergence-free condition

$$\mathcal{V}^* \triangleq \underset{\mathcal{V}^*}{\operatorname{argmin}} \frac{1}{2}\int_x \|\mathcal{V}^*(p) - \mathcal{V}(p)\|^2 \quad \text{s.t.} \nabla\cdot\mathcal{V}^* = 0, \qquad (8)$$

$$\mathrm{MLP}(o, \theta) = \left(\phi_1 \cdots \phi_K \right). \tag{9}$$

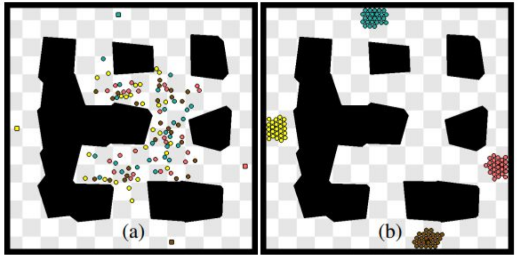
Layer	Kernel	Stride	#Filters/#Neuron	Activation
Conv_1	(7,7)	1	8	MaxPool.+ReLU
Conv_2	(7,7)	1	12	MaxPool.+ReLU
Conv_3	(5,5)	1	16	MaxPool.+ReLU
Conv_4	(3,3)	1	20	MaxPool.+ReLU
MLP_1	/	/	128	ReLU
MLP_2	/	/	5K	Sigmoid

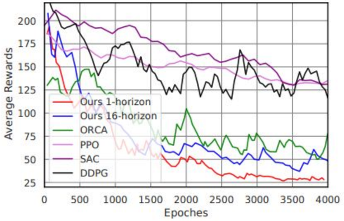
- During each iteration of training, we sample a batch B and optimize R over a receding horizon of H timesteps

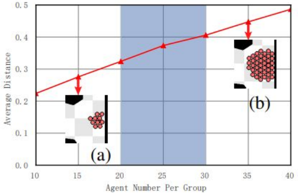
$$\theta \leftarrow \theta + \alpha_{\mathrm{lr}} \nabla_\theta \sum_{x_i^t \in \mathcal{B}} \sum_{h=1,\cdots,H} \gamma^t R(x_i^{t+h\Delta t}). \tag{10}$$

EVALUATION

- We compare our method with three model-free RL baselines (PPO [46], SAC [47], and DDPG [48]) to train our policy using the same reward function.







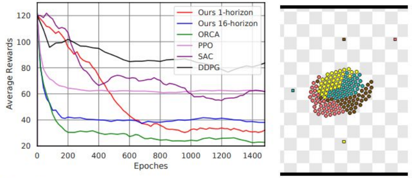
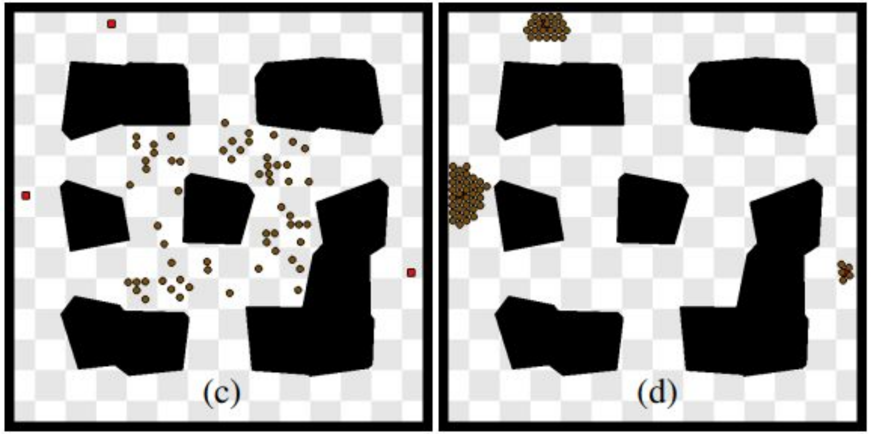
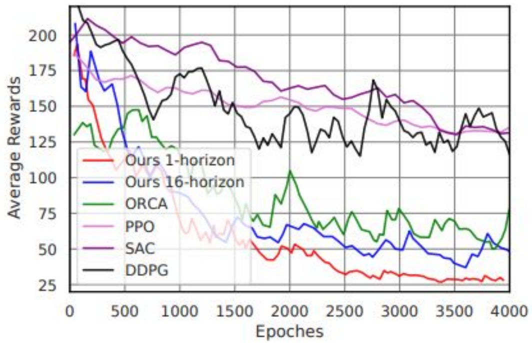


Fig. 5: Comparison as Figure 3 in a congested scenario (right).

$$R(x_i^{t+\Delta t}) = \min_{j=1,\cdots,M} \mathrm{dist}_j(x_i^t) - \min_{j=1,\cdots,M} \mathrm{dist}_j(x_i^{t+\Delta t}), \quad (12)$$





CONCLUSION & LIMITATION

- We present an end-to-end differentiable learning algorithm for multi-agent navigation tasks
- we show that our method outperforms the model-free RL algorithm by more than one order of magnitude