Milestone Report: Social Navigation with RL

Yun-Jin Li
Technical University Munich
MSc. Robotics, Cognition & Intelligence
Munich, Germany
yunjin.li@tum.de

Kagan Kücükaytekin
Technical University Munich
MSc. Robotics, Cognition & Intelligence
Munich, Germany
kagan.kuecuekaytekin@tum.de

I. Baseline

We took the method developed by Chen et. al. [1] as baseline and started improving on their work. Our repository also starts as a fork from their work and can be found here: https://github.com/kkaytekin/tum-adlr-ss22-05/. Even though their model performs well on simple configurations, we found some issues that might be interesting to investigate further.

II. Approach

1) Representation: We follow the same robot-human representation in [1], [2].

$$\begin{aligned} s &= [p_x, p_y, v_x, v_y, r, g_x, g_y, v_{pref}, \theta], \\ w_i &= [p_x, p_y, v_x, v_y, r], \end{aligned} \tag{1}$$

where s represents the robot state and w_i represents the state of the pedestrian or obstacle i. p, v, g, r, θ represent in order: the position, velocity, radius, the goal coordinate and the orientation. For rectangular obstacles, we consider the rectangle as its minimum circumscribed circle. In the case of limited field of view, the ground truth values in w_i are provided if the agent is observed. Else, an approximation based on the last visible state is provided.

2) Reward: We use the same reward criteria as [1].

3) Learning: We apply the same architecture in [1], but we slightly change the interaction module in order to take the obstacles and limited field of view into account. Besides, we follow the same process by first training the model with imitation learning and then proceeding into the value network training.

III. Extensions to the Baseline

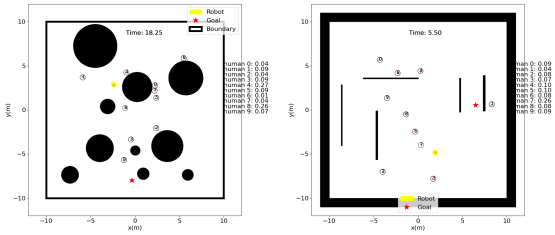
A. The complexity of the simulation world

The scenarios in the baseline environment are too simple, and they under-represent the complexity of real-world environments. In order to make the simulation close to the real-world, we decided to improve the current baseline by the following means.

1) Humans: After reaching their goals, the humans will be given a new goal instead of stopping.

2) Robot: Instead of fixed start and goal positions for all scenarios, the robot will have random start and goal positions for every new scenario.

3) Static obstacles: In reality, there are always both dynamic and static obstacles existing in the same time. As the result, we randomly generate circular or rectangular obstacles to make the environment more complicated as illustrated in the figure 1.



(a) Circular scene (b) Rectangular scene

Fig. 1: Both type of static obstacles environment

4) Boundary: In order to prevent the robot from choosing the easy solution of going around all obstacles, we added a boundary to our simulation.

B. Limited field of view

The baseline method assumed that the robot uses Lidar to perceive its environment, which is expensive and not widely available. We instead decided to assume that the robot uses single RGB-D camera with a 126-degree field of view. This makes the problem even harder and brings additional challenges for our robot.

We start by taking a step back and building up onto the base simulation without adding obstacles for initial experiments. We use the same architecture used for the base method, but we modify the state vector of the observed agents. For the time steps an agent is unseen, we replace the observation with one of the following approximations:

Stationary: Provide the last seen velocity and position, i.e. no approximation.

Continuing: Update the last known position by $time\ step\ size \times last\ velocity$ for each time step the agent is unseen.

Slowing Down: Same as the previous method, but in addition, decay the velocity by a predefined rate prior to every update step.

Stationary Expanding Bubble: Do not update position and velocity, but increase the radius of the unseen agent as it stays unseen.

Moving Expanding Bubble: Combine "continuing" and "stationary expanding bubble".

C. Extension of the current architecture

We modified the input interface of the original SARL algorithm [1] to make sure it could be trained in the new environment.

IV. Result

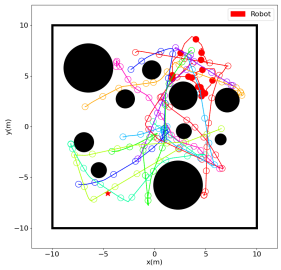
A. Randomized circular obstacles

In table I, we specify the training setting and the result. The result implies that the robot tries its best to avoid collisions but ends up resulting in the so-called "freezing" point in most of the scenarios illustrated by figure 2a.

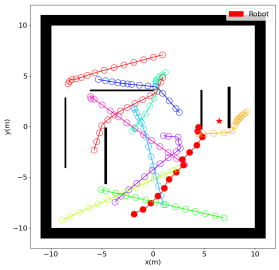
Training Parameter				
Train Episodes	Memory	Humans	Obstacles	Imitation Episodes
4700	100000	10	10	3000

Evaluation			
	Success	Collision	Rewards
RL	0.13	0.13	-0.1052
ORCA	0.09	0.86	-0.1351

TABLE I: Circular obstacles training results



(a) Robot freeze in the circular obstacles environment



(b) Robot collide with the rectangular obstacle

Fig. 2: Trajectories in different environment setting

B. Randomized rectangular obstacles

In table III, we specify the training setting and the result. We can see that our model can actually perform slightly better than ORCA in a more complicated environment. However, unlike the cases in circular obstacles environment, the collision rate in rectangular obstacles environment is pretty high. This might result from the bad parametrization for the rectangular obstacles.

Training Parameter				
Train Episodes	Memory	Humans	Obstacles	Imitation Episodes
10000	100000	10	5	3000

Evaluation			
	Success	Collision	Rewards
RL	0.29	0.71	-0.0599
ORCA	0.25	0.69	-0.0796

TABLE II: Rectangular obstacles training results

C. Limited field of view

For the limited field of view case, we see that the robot develops intelligent behaviour after 10k episodes of training. It tries to keep every pedestrian at sight before reaching halfway to the goal. It also understands that, after reaching halfway, the pedestrians will continue towards their own goals. As long as it keeps looking at the goal direction, it will avoid collisions after this point. For the "stationary" method, robot always watches the pedestrians as they pass and does not take risks. Because it has no assumptions for the trajectories of pedestrians, taking risk usually ends up with penalty. With the heuristic assumptions, the robot can take more risk to obtain higher rewards and lower navigation times.

Training Parameters				
Train Episodes	Memory	Humans	Obstacles	Imitation Episodes
10000	100000	5	None	3000

Evaluation				
	Success	Collision	Nav. Time(s)	Rewards
Stationary	**1.00**	**0.00**	11.79	0.2916
Continuing	0.99	0.01	10.76	0.3214
Slowing Down	0.97	0.01	12.06	0.2663
Stat. Exp. Bub.	0.98	0.02	11.25	0.3013
Moving. Exp. Bub.	0.99	0.01	**10.55**	**0.3259**

TABLE III: Limited field of view training results

D. Next steps

The current model can only accept a fixed number of obstacles (humans + obstacles), so we would like to come up with a new parametrization approach to solve this problem. Besides, we plan to add noise model for the observation. Furthermore, in our first experiment, it seems that the effect of imitation learning is not so helpful, we decide to investigate this further. Last but not least, we want to combine all of our environment setting together and also try out some other reinforcement learning strategies. If we have enough time left, we will lift our simulation to 3D.

References

[1] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[2] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.