Implementation of SARL* Algorithm for A Differential Drive Robot in a Gazebo Crowded Simulation Environment

Seif Eddine Seghiri, Noura Mansouri, Ahmed Chemori

Implementation of SARL* Algorithm for A Differential Drive Robot in a Gazebo Crowded Simulation Environment

1st Seif Eddine Seghiri
LARC Laboratory
University of Constantine 1
Constantine, Algeria
seifeddine.seghiri@umc.edu.dz

2nd Noura Mansouri
LARC Laboratory
University of Constantine 1
Constantine, Algeria
nor_mansouri@yahoo.fr

3rd Ahmed Chemori
LIRMM Laboratory
University of Montpellier
CNRS, Montpellier, France
Ahmed.Chemori@lirmm.fr

***Abstract*—Because of the stochasticity in people's behaviors, autonomous navigation in crowded environments is critical and challenging for both the robot and people evolving around. This paper deals with the implementation and effectiveness evaluation of the Socially Attentive Reinforcement Learning star algorithm, namely SARL*, which is an extended version of the state-of-the-art socially compliant navigation algorithm SARL. It introduces a dynamic local goal resetting mechanism. The Simulations were conducted in the Robot Operating System (ROS) and the Gazebo simulator is used to test the human-aware navigation in different scenarios. Simulation results illustrate the efficiency of SARL* in terms of navigation around people in a socially acceptable manner. Nevertheless, it could not navigate efficiently when the goal position is located behind static or quasi-static obstacles.**

***Index Terms*—SARL*, Social Robotics, ROS, Deep Reinforcement Learning, Navigation**

I. INTRODUCTION

A robotic framework is often constructed around three main entities, including (i) guidance, (ii) navigation and (iii) control (GNC) systems. These systems mainly interact with each other through the transmission of data and signals. In its basic form, GNC is a reference model (the guidance system), a measurement/sensor system (the navigation system) and a controller (the control system) [1]–[4].

Mobile robot navigation has been intensively researched as a basic problem in robotics. Today, more service robots are being created to function in human-robot coexisting situations, including, autonomous vehicles, smart wheelchairs, luggage collecting and hospitality robots [5]–[10].

In traditional navigation frameworks typically, Collision avoidance modules treat dynamic obstacles as static, such as the Dynamic Window Approach (DWA) [11] or simply focus on the next step of action based on certain interaction rules, such as Reciprocal Velocity Obstacle (RVO) [12] and Optimal Reciprocal Collision Avoidance (ORCA) [13]. Because these strategies prevent collisions by passive reaction and typically rely on manually programmed functions to assure safety, it has been revealed that they lead the robot's movement to be awkward, short-sighted, and dangerous.

In congested environments, a robot must be able to observe, interpret and predict the behavior of the surrounding pedestrians in order to move in a socially acceptable manner. In the human aware navigation problem, the integration of human motion prediction with robot motion planning remains a difficult problem. One of the existing approaches consists to plan after prediction, i.e., to choose a safe path after predicting the future trajectories [14]. To anticipate pedestrian trajectories, some hand-crafted models (e.g., constant velocity model [15], discrete choice model [16], social force model and its variations [17], [18]) and data-driven approaches (e.g., social LSTM [19], [20], social GAN [21], [22]) have been presented.

However, the significant stochasticity of the crowd behavior frequently affects the computing cost and reliability of pedestrian trajectory prediction. Such planning-after-prediction approaches are still hard for practical implementations in the context of human-aware navigation for mobile robots, which demands both security and time efficiency [23].

Deep Reinforcement Learning (DRL) is another category of algorithms for human-aware navigation that includes human motion prediction into the decision-making process, i.e., the robot learns from experiences to comprehend crowded environments and encodes crowd-robot interaction in the navigation strategy. Recent studies [24]–[29] shown greater ability to create crowd-aware navigation rules, with the Socially Attentive Reinforcement Learning (SARL) algorithm obtaining cutting-edge performance for an effective DRL based human-aware navigation.

II. PROBLEM FORMULATION

Let us consider a robot and n humans as agents, in a 2D workspace. Each of these agents has an observed states such as position $p = [p_x, p_y]$, velocity $v = [v_x, v_y]$ and radius r (i.e. agents are represented as circles in their workspace). For each agent the rest of the states, such as the orientation θ, goal position $g = [g_x, g_y]$ and the preferred speed v_{pref}, are not observed by other agents.

The robot's full state at time t can be defined as $\mathbf{S}_t = [p_x, p_y, v_x, v_y, \theta, g_x, g_y, r, v_{pref}]$, and the k-th human's observable state at time t can be defined as $\mathbf{O}_t^k = [p_x^k, p_y^k, v_x^k, v_y^k, r^k, v_{pref}^k]$. To make the state representation more general, the current and goal positions (p, g) are replaced by the distance between them and denoted d_g. The distance between the robot and the k-th human is denoted by d^k. As adopted in [26] the new state is defined as follows:

$$\begin{aligned} \mathbf{S}_t &= [d_g, v_{pref}, v_x, v_y, r] \\ \mathbf{O}_t^k &= [d^k, p_x^k, p_y^k, v_x, v_y, r^k, r^k + r] \end{aligned} \tag{1}$$

A joint state of all $(n+1)$ agents at time t is constructed by concatenating the state $\mathbf{S}_t$ with the all of the $\mathbf{O}_t^k$:

$$\mathbf{J}_t = [\mathbf{S}_t, \mathbf{O}_t^1, \mathbf{O}_t^2, ..., \mathbf{O}_t^n] \tag{2}$$

Differential drive robot is controlled by linear and angular velocity commands (i.e. actions $\mathbf{a}_t$) according to a specified navigation policy $\pi(\mathbf{J}_t)$: $\pi(\mathbf{J}_t) = \mathbf{a}_t = \mathbf{v}_t$. At each time the robot is awarded by $R(\mathbf{J}_t, \pi(\mathbf{J}_t))$. The form of the reward function is as following:

$$R(\mathbf{J}_t, \mathbf{a}_t) = \begin{cases} -0.25, & \text{if } d_{min} < 0; \\ 0.5\ (d_{min} - d_c), & \text{if } 0 < d_{min} < d_c; \\ 1, & \text{if } d_g = 0; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Where d_{min} represents the smallest separation distance between the robot and people during the decision interval Δt, and d_c is the shortest comfortable distance that humans can handle.

The optimal value of $\mathbf{J}_t^*$ at time t can be formulated as:

$$V^*(\mathbf{J}_t) = \sum_{i=0}^{K} \gamma^{i.\Delta t.v_{pref}} .R(\mathbf{J}_t, \mathbf{a}_t^*) \tag{4}$$

where $\mathbf{a}_t^*$ is selected according to a given optimal policy $\pi^*(\mathbf{J}_t)$, K is the total number of decision steps from the state at time t to the final state, Δt is the decision interval between two actions $\mathbf{a}_t$, and $\gamma \in [0, 1]$ is a discount factor in which the preferred speed v_{pref} is introduced as a normalization parameter.

Maximizing the cumulative reward yields the optimal policy, which is as follows:

$$\begin{aligned} \pi^*(\mathbf{J}_t) = \underset{\mathbf{a}_t \in \mathbf{A}}{\arg\max}\, R(\mathbf{J}_t, \mathbf{a}_t) + \gamma^{\Delta t.v_{pref}} \\ \int_{\mathbf{J}_{t+\Delta t}} P(\mathbf{J}_{t+\Delta t} \mid \mathbf{J}_t, \mathbf{a}_t).\mathbf{V}^*(\mathbf{J}_{t+\Delta t}) d\mathbf{J}_{t+\Delta t} \end{aligned} \tag{5}$$

with $\mathbf{A}$ is the action space $P(\mathbf{J}_{t+\Delta t} \mid \mathbf{J}_t, \mathbf{a}_t)$ is the transition probability from $\mathbf{J}_t$ to $\mathbf{J}_{t+\Delta t}$ when action $\mathbf{a}_t$ is selected.

Given that pedestrian intentions are difficult to foresee and that the length Δt is quite short, it is reasonable to suppose that pedestrians walk at a constant speed within the time interval $[t, t+\Delta t]$. As a result, we can predict the future states of humans and approximate the next combined state using a constant velocity model: $\mathbf{J}_{t+\Delta t} \leftarrow propagate(\mathbf{J}_t, \Delta t, \mathbf{a}_t)$.

The computation of the optimal strategy defined in (5) is simplified as follows:

$$\pi^*(\mathbf{J}_t) = \underset{\mathbf{a}_t \in \mathbf{A}}{\arg\max}\, R(\mathbf{J}_t, \mathbf{a}_t) + \gamma^{\Delta t.v_{pref}} .\mathbf{V}^*(\mathbf{J}_{t+\Delta t}) \tag{6}$$

III. Deep Reinforcement Learning (DRL)

In Reinforcement Learning (RL), value functions $\mathbf{V}(\mathbf{J}_t)$ can be evaluated in a variety of ways. One approach is to utilize tables to record the values for each state $\mathbf{J}_t$ or action-state pair $(\mathbf{J}_t, \mathbf{a}_t)$. However, this method, does not scale with the sizes of state and action spaces. Another approach is to employ neural networks to estimate value functions or, to generate an action distribution given input states. Artificial neural networks can be viewed as universal function approximators [30], which means that they can represent arbitrarily complicated mappings between spaces if properly trained.

The trained deep neural network takes as inputs the state $\mathbf{J}_t$ and some features that could be collected from agents (e.g. range measurements), and outputs the estimated value function namely Value Network.

The training of the value network used in the SARL* algorithm is summarized by the Algorithm 1.

Algorithm 1 Value Network Training

Initialize experience replay memory E with demonstrations;
Initialize the value network N with memory E;
Initialize the target value network $N' \leftarrow N$;
for episode = $1, ..., M$ **do**
  Initialize $\mathbf{J}_0$ randomly;
  repeat
    $\mathbf{a}_t \leftarrow EpsilonGreedyActionSelection()$;
    $value \leftarrow R(\mathbf{J}_t, \mathbf{a}_t) + \gamma^{\Delta t.v_{pref}}.\mathbf{N}'(\mathbf{J}_{t+\Delta t})$;
    $state \leftarrow \mathbf{J}_{t+\Delta t}$;
    Enrich experience $E \leftarrow (state, value)$;
    Optimize value network N with experience E;
    $t \leftarrow t + \Delta t, \mathbf{J}_t \leftarrow \mathbf{J}_{t+\Delta t}$;
  until $\mathbf{J}_t = \mathbf{J}_{end}$ or $timeout$;
  if $episode \mid TargetUpdateInterval$ **then**
    Update the target value network $N' \leftarrow N$;
  end if
end for
return N;

IV. SARL* Algorithm

The difference between SARL and SARL* is that SARL* introduces a dynamic goal resetting due to the SARL algorithm's training conditions [23], [29]. The DWA local planner [11] is used in each Δt to reset a local goal which is considered as a target goal for the SARL planner. The algorithm keeps controlling the robot to move towards the local goal till it reaches the global goal of the robot.

Furthermore, the learning of SARL algorithm does not consider the static nor the quasi-static obstacles in the 2d environment. Therefore, they introduced a procedure for checking whether the best selected action $\mathbf{a}_t^*$ according to the optimal policy $\pi^*(\mathbf{J}_t)$ will drive the robot to an obstacle or not. The action will be aborted if it will drive the robot to an obstacle and the robot will choose rotating in place in order to find another valid optimal action.

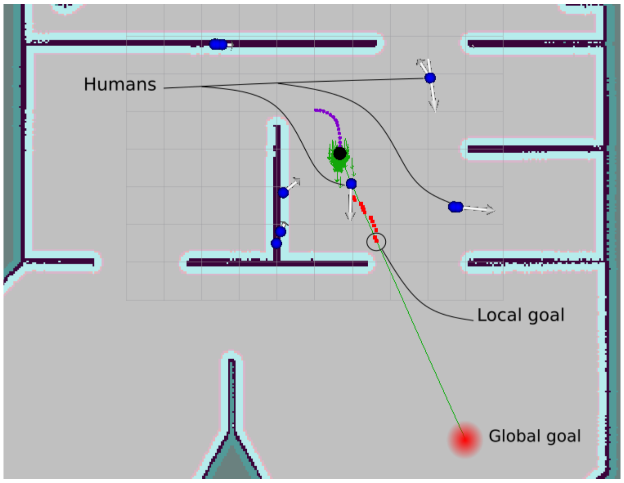

Fig. 1. Illustrative view of the robot, humans, global and local paths in the 2d map environment. The optimal action is selected with respect to the red points (local goals). Eventually, the robot will arrive at its final destination, while avoiding humans navigating around based on the SARL policy.

Algorithm 2 SARL*

Initialize $\mathbf{A}$;
Load the pre-trained value network N;
Build the $2D\ map$;
Set global goal position $\mathbf{g}_{global}$;
while $d_g > goal\_tolerance$ **do**
  Update $\mathbf{J}_t$;
  $GlobalPlan \leftarrow Dijkstra(p_t, \mathbf{g}_{global}, map)$;
  $\mathbf{g}_{local} \leftarrow FindLocalGoal(p_t, GlobaPlan, d)$;
  $\mathbf{a}_t \leftarrow \arg\max_{\mathbf{a}_t \in \mathbf{A}_s} R(\mathbf{J}_t, \mathbf{a}_t) + \gamma^{\Delta t.v_{pref}}.\mathbf{N}(\mathbf{J}_{t+\Delta t})$,
  with $\mathbf{g} \leftarrow \mathbf{g}_{global}$,
    $\mathbf{A}_s = FindSafeActionSpace(p_t, \mathbf{A}, map)$
end while
return N;

As it can be seen from the ROS RViz application depicted on Fig. 1, humans are represented by the blue dots. The robot is moving towards the red dots (local goals), so that it will eventually reaches the global goal.

V. Description of the Simulation Environment

The mobile robot used for simulation is the Turtlebot2, as illustrated in Fig. 2. It is a low-cost differential-drive mobile robot built out of a Yujin Kobuki mobile platform plus sensors and actuators such optical encoders and DC motors. It is equipped with a Microsoft XBOX 360 Kinect along with a Hokuyo lidar sensor.
The robot's simulation ROS package integrates the Kobuki platform with the lidar and Kinect ROS modules. Using the ROS navigation stack, these modules enables simulation and implementation of many mobile robotics tasks such as perception, localization, mapping, navigation, and control.

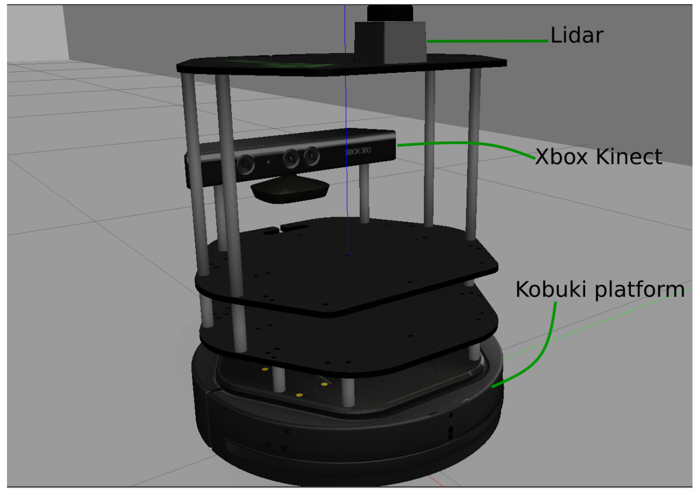

Fig. 2. View of the Turtlebot2 Differential drive mobile robot used in simulation including its basic components and sensors.

The Robot Operating System (ROS) melodic is installed on an 8GB of RAM and i5 CPU PC running under Ubuntu 18.04 operational system along with Gazebo simulator, enables the creation of a realistic human-robot coexisting environment. The simulation environment depicted in Fig. 3 could be a good testing place where there are humans navigating alongside the robot. They move randomly in the environment and do not follow a certain navigation policy.
Each human is tracked using the people ROS package [31]. The robot and the humans states $\mathbf{S}_t,\ \mathbf{O}_t^i$ are used to update the joint state $\mathbf{J}_t$. The SARL* algorithm is then started controlling the robot to move towards its goal while avoiding humans and obstacles.

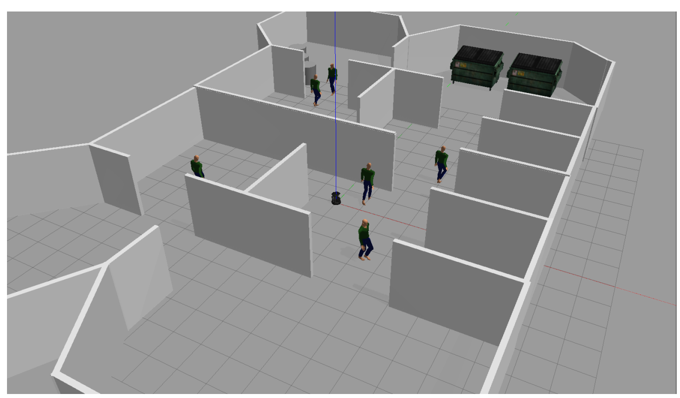

Fig. 3. Top view of the human-robot simulation environment used to test the SARL* navigation efficiency and robustness.

VI. Numerical simulation results

In order to test the efficiency of the SARL* algorithm, we conducted the simulation in two steps. In the first one, we give the robot a goal position in front of it, and near its current position. In the next step, the goal position is placed behind and a little bit far from the robot. Each of these two scenarios is tested by running the algorithm 5 times and then another 5 times while placing new static obstacles along each trajectory in such a way that the robot will encounter them. These obstacles are not present in the 2D occupancy grid map. The time limit for the navigation is defined as $t_{max} = 200sec$. It is valuable to mention that the robustness of SARL* is also considered, by controlling the simulated humans to randomaly walk in different speeds rather than navigating according to the same training conditions such as circle crossing or the Optimal Reciprocal Collision Avoidance (ORCA) [13].

A. Scenario 1

In this scenario, the robot is located initially at $p_0 = [0.0, 0.0]$ and receives a goal position $p_g = [6.22, 0.51]$. The SARL* policy will drive the robot from p_0 to p_g while avoiding humans. Fig. 4 illustrates how SARL* is a good human-aware navigation policy in terms of finding the solution, time and path length. We also indicate on Fig. 4, the time required for the robot to arrive at the goal position at each run.
The two curvatures generated by the SARL* policy are resulting from the humans passing beside the robot, where it successfully navigates through them. The same scenario is

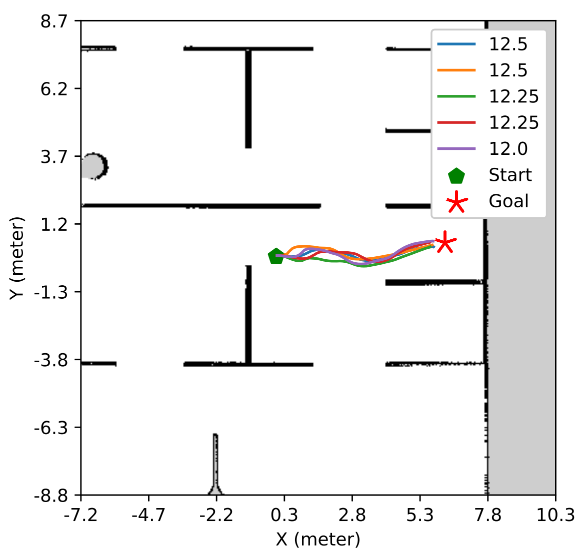

Fig. 4. Illustration of the trajectories that the robot takes for each one of the 5 simulations. At the top right corner, the time in seconds taken by the robot while navigating using the SARL* policy is indicated.

running with a static cylindrical obstacle placed along the trajectory (i.e. along the global path). Fig. 5 Illustrates the length of the trajectory taken by the SARL* policy to converge towards the goal position and each curve is associated with the time required to arrive at the final state.

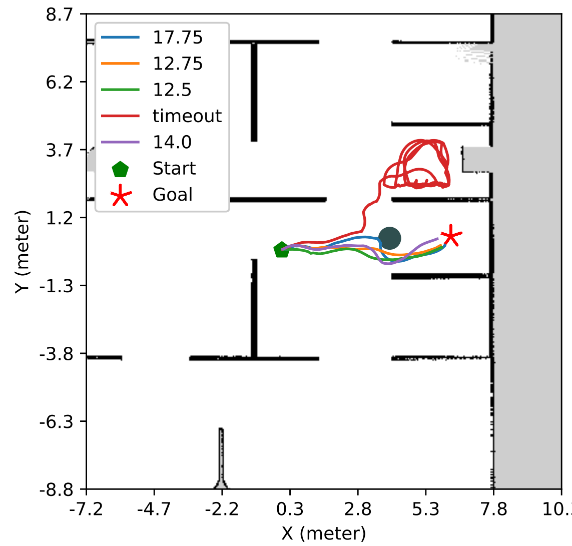

Fig. 5. Illustration of the trajectories taken by the robot to reach the goal pose, for each one of the 5 executions. The time in seconds taken by the robot obeying the SARL* policy is indicated. Where, the cylindrical obstacle is placed at $p_{obs} = [4.0, 0.44]$.

B. Scenario 2

This scenario will be more challenging than the previous one, due to the distance between the initial robot's pose $p_0 = [0.0, 0.0]$ and the target position $p_g = [-6.00, -8.92]$. Like in the first scenario, each curve is associated with the time taken by the robot's navigation. In this first part of the second scenario the robot is placed at p_0 and receives the target pose p_g. The SARL* algorithm will then drive the robot towards p_g. Another challenging problem in this part is that the robot will encounter an L-shaped obstacle as depicted in Fig. 6. The results will be illustrated through Fig. 7. The figure describes the trajectories taken by the robot to go from p_0 to p_g. In addition to the time required for it, in each run.

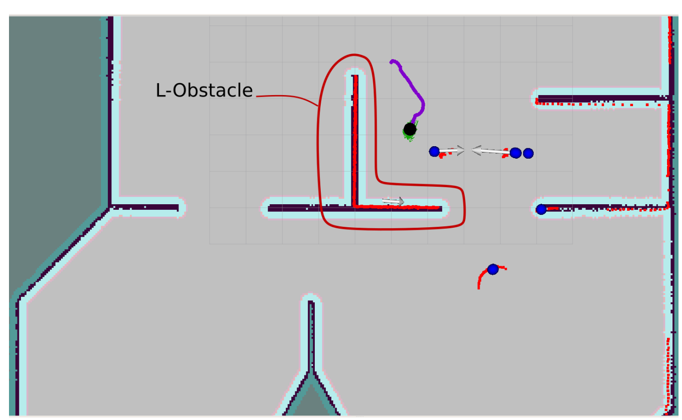

Fig. 6. Illustration of the challenging L-shaped obstacle that could make the robot get stuck.

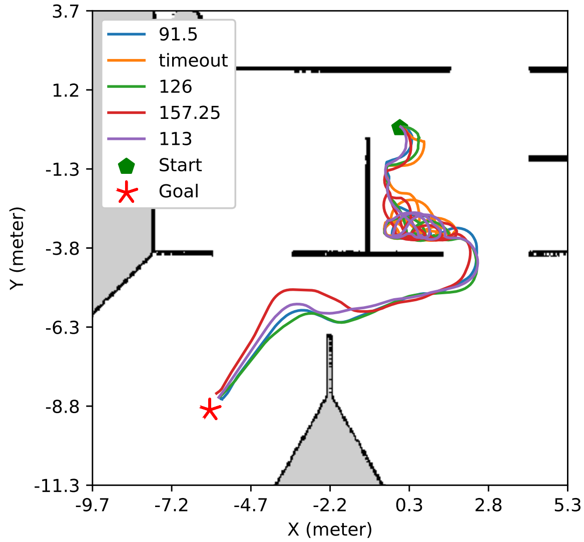

Fig. 7. Illustration of the 2^{nd} scenario trajectories taken by the robot for each one of the 5 simulations. The travel time in seconds is indicated in the top left corner.

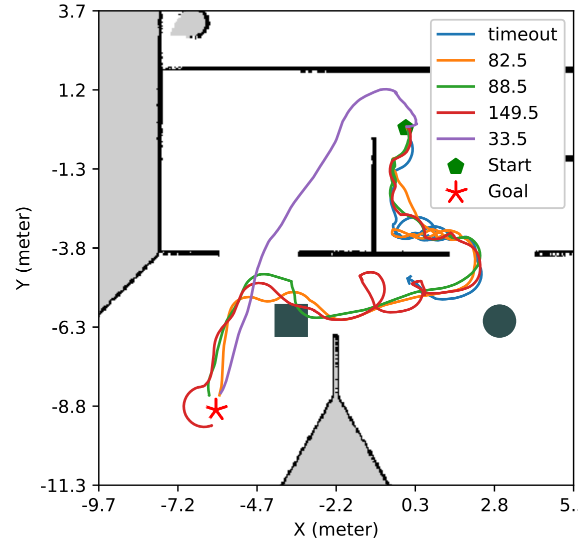

Fig. 8. Illustration of the trajectories that the robot takes for each of the 5 simulations. The time in seconds taken by the SARL* policy to successfully arrive at the desired position is also indicated in the top right corner. The robot pushed away the cubic obstacle, which can be seen where some trajectories are drew on top of the obstacle.

It can be seen from Fig. 7 that the robot makes a lot of detours inside the L-shaped wall. This is because the SARL* navigation dynamic goal resetting mechanism is not taking the occupancy map in account. Therefore, it will obviously try to move toward the dynamic local goal even if it is placed behind and obstacle. The second part of this scenario is that a cylindrical obstacle and cubic obstacle are placed at $p^1_{obs} = [2.96, -6.11]$ and $p^2_{obs} = [-4.12, -5.6]$ respectively. The navigation policy will drive the robot like in the first part of the scenario. However, it has to avoid the new obstacles.
Fig. 8 is an illustration of the trajectories and the their time taken by the robot while following the SARL* navigation policy.
The time required for the robot to reach its goal is highly dependent on how many humans it encountered during its travel. One can imagine a scenario where the robot avoids a human by taking a right turn. Yet, if the robot is trying to avoid a human to the right, the right side has to be clear (i.e. it could get stuck). This explains the reason behind the high variance in the travel time.
The robustness of SARL* is quite acceptable. Nevertheless, in each one of the simulation scenarios when new obstacles are added to the environment, the robot did collide with them one or two times. This can bee deduced from the trajectories overlapping obstacles in Fig. 5 and Fig. 8. However, obstacles are not fixed and the robot could push them away from its path and continue moving along its trajectory.
The approach proposed by Li et al. in [23] is only effective when the goal location is unobstructed. It can be seen in Figs. 7 and 8, the detours made by the robot that are caused by the L-shaped wall which is a common obstacle in all real life human-robot coexisting indoor environments.

VII. Conclusions and Future Work

We evaluated the effectiveness and robustness of SARL* in dealing with real-life socially indoor scenarios. Following a considerable amount of simulations, we found that SARL* policy is an effective technique for socially acceptable navigation in crowded environments. However, a 2D cost map of the surroundings is indispensable. To effectively deploy a socially attentive reinforcement learning-based human-aware navigation in real-world applications, static, quasi-static and dynamic obstacles must be integrated into the network training approach. As a future work, we suggest that the reward function design and the training environment must consider both social and indoor environmental information, such as walls and doors.

References

[1] D. Maalouf, V. Creuze, and A. Chemori. A novel application of multivariable L1 adaptive control: From design to real-time implementation on an underwater vehicle. In *Proc. IEEE/RSJ Int. Conf. Intel. Robots and Systems*, Algarve, Portugal, 2012.

[2] T. Salumae, A. Chemori, and M. Kruusmaa. Motion control of a hovering biomimetic 4-fin underwater robot. *IEEE Journal of Oceanic Engineering*, 44(1), 2019. DOI: 10.1109/JOE.2017.2774318.

[3] J. Campos, E. Monroy, H. Abundis, A. Chemori, V. Creuze, and J. Torres. A nonlinear controller based on saturation functions with variable parameters to stabilize an AUV. *International Journal of Naval Architecture and Ocean Engineering*, 11(1):211–224, 2019. DOI: 10.1016/j.ijnaoe.2018.04.002.

[4] J. Guerrero, J. Torres, V. Creuze, and A. Chemori. Observation-based nonlinear PD control for robust trajectory tracking for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 45(4):1190–1202, 2020. DOI: 10.1109/JOE.2019.2924561.

[5] S Chandramohan and M Senthilkumaran. Intelligent automatic guided vehicle for smart manufacturing industry. In *Advances in Materials and Manufacturing Engineering*, pages 337–343. Springer, 2021.

[6] Kung-Jeng Wang and Darwin Santoso. A smart operator advice model by deep learning for motion recognition in human–robot coexisting assembly line. *The International Journal of Advanced Manufacturing Technology*, pages 1–20, 2021.

[7] Aidin Ferdowsi, Ursula Challita, Walid Saad, and Narayan B. Mandayam. Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 307–312, 2018.

[8] Louis Lecrosnier, Redouane Khemmar, Nicolas Ragot, Benoit Decoux, Romain Rossi, Naceur Kefi, and Jean-Yves Ertaud. Deep learning-based object detection, localisation and tracking for smart wheelchair healthcare mobility. *International journal of environmental research and public health*, 18(1):91, 2021.

[9] Alexander Wilkinson, Michael Gonzales, Patrick Hoey, David Kontak, Dian Wang, Noah Torname, Sam Laderoute, Zhao Han, Jordan Allspaw, Robert Platt, et al. Design guidelines for human–robot interaction with assistive robot manipulation systems. *Paladyn, Journal of Behavioral Robotics*, 12(1):392–401, 2021.

[10] Aarni Tuomi, Iis P Tussyadiah, and Jason Stienmetz. Applications and implications of service robots in hospitality. *Cornell Hospitality Quarterly*, 62(2):232–247, 2021.

[11] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[12] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE, 2008.

[13] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[14] Yusheng Peng, Gaofeng Zhang, Xiangyu Li, and Liping Zheng. Stirnet: A spatial-temporal interaction-aware recursive network for human trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2285–2293, 2021.

[15] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *2009 IEEE 12th International Conference on Computer Vision*, pages 261–268. IEEE, 2009.

[16] Gianluca Antonini, Michel Bierlaire, and Mats Weber. Discrete choice models of pedestrian walking behavior. *Transportation Research Part B: Methodological*, 40(8):667–687, 2006.

[17] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[18] Siming Fang, Zhengjiang Liu, Xinjian Wang, Jin Wang, and Zaili Yang. Simulation of evacuation in an inclined passenger vessel based on an improved social force model. *Safety Science*, 148:105675, 2022.

[19] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[20] Yusheng Peng, Gaofeng Zhang, Jun Shi, Benzhu Xu, and Liping Zheng. Srai-lstm: A social relation attention-based interaction-aware lstm for human trajectory prediction. *Neurocomputing*, 2021.

[21] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018.

[22] Hao Zhou, Dongchun Ren, Huaxia Xia, Mingyu Fan, Xu Yang, and Hai Huang. Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction. *Neurocomputing*, 445:298–308, 2021.

[23] Keyu Li, Yangxin Xu, Jiankun Wang, and Max Q-H Meng. Sarl star: Deep reinforcement learning based human-aware navigation for mobile robot in indoor environments. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 688–694. IEEE, 2019.

[24] Zhengxi Hu, Yingli Zhao, Sen Zhang, Lei Zhou, and Jingtai Liu. Crowd-comfort robot navigation among dynamic environment based on social-stressed deep reinforcement learning. *International Journal of Social Robotics*, pages 1–17, 2021.

[25] Sunil Srivatsav Samsani and Mannan Saeed Muhammad. Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):5223–5230, 2021.

[26] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[27] Yu Fan Chen, Michael Everett, Miao Liu, and Jonathan P How. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017.

[28] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[29] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[30] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

[31] Caroline Pantofaru. http://wiki.ros.org/leg_detector.