Mobile Robot Navigation Strategies Through Behavioral Cloning and Generative Adversarial Imitation Learning

Kevin Levrone Rodrigues Machado Silva and Rodrigo Calvo
Department of Computer Science, State University of Maringa,
5790 Colombo Avenue, 87020-900, Maringa-PR, Brazil

Keywords: Autonomous Robots, Mobile Robots, Behavioral Cloning, Generative Adversarial Imitation Learning, GAIL, Supervised Imitation Learning, Robot Navigation, Braitenberg Vehicles, Machine Learning, Inverse Reinforcement Learning, Imitation Learning.

Abstract: The conception of robots able to navigate autonomously through several environments remains one of the main challenges to be overcome in the robotics research. The wide use of machine learning techniques as imitation learning has obtained efficient performance in this research field. The autonomous navigation is essential to carry out many kinds of task, which it can reduce the time and computational cust. One of the mechanisms to a robot be autonomous is observe the behavior of the other. Therefore, it is proposed in this research the development of a strategy of navigation based on Generative Adversarial Imitation Learning (GAIL) for the learning of the navigational behaviors of distinct mobile robots with different locomotion strategies. The CoppeliaSim simulator is used to build virtual scenarios and execute experiments to gather samples for the strategy training. Another strategy will be developed based on the behavioral cloning, which will also be trained in some environments with the same samples used in GAIL. Regression error metrics in addition with the comparison of the paths generated by the strategies in each scenario will be considered as evaluation methods. The obtained results are then discussed along with the potential future works.

1 INTRODUCTION

In a navigation system, the achievement of complete autonomy by agents remains one of the main challenges to be overcome in the study branches of robotics and automobiles. In this context, machine learning techniques have relevant role in the generation of autonomous agents. The most traditional techniques are based on Reinforcement Learning (RL) and Imitation Learning (IL) and found in the literature.

The GAIL was proposed in (Ho and Ermon, 2016) as an approach that uses concepts from Generative Adversarial Networks (GAN) coupled with the principles of Inverse Reinforcement Learning (IRL) to enable the extraction of an optimal policy through behavioral demonstrations that would come from an expert agent and then input to the training stage.

Although approaches using GAIL got successful results to imitate behaviors of an expert, a wide research regards that issue is recent in order to reproduce the navigational behavior of a mobile robot. Following this trend, the proposal of this project includes the development of a GAIL-based navigation strategy that could learn and reproduce the behaviors demonstrated by two different models of land mobile robots, where each of them will use a different navigation approach that will be applied in several environments.

Another strategy based on BC will be developed to compare its performance and the GAIL-based strategy performance. The BC–based strategy is also able to learn the behaviors demonstrated by agents in the same proposed environment. In addition, it is pretended to evaluate the robustness of strategies when distinct environments and robots are considered. Data related to the current state of the robot in a environment and the adopted action for the next state will be collected in order to use them as input data for GAIL and BC training.

After training period, it is expected that both techniques are able to provide actions that lead a robot to reproduce the demonstrated behaviors previously by an expert. The BC and GAIL implementations are imported from Imitation (Wang et al., 2020), an API that provides clean implementations of imitation and reward learning algorithms. Therefore, there is the possibility to ascertain the level of suitability of the use of both implementations in custom environments that are not present in Gym, the standard API for reinforcement learning that also has a collection of reference environments (Brockman et al., 2016).

The remaind of that text is organized as follow. Works related to that current research are presented in Section 2. In Section 3, information about the tools used, the methodology and the development of the strategies are addressed. The obtained results are analyzed and discussed in Section 4. Finally, the Section 5 presents the conclusion with a brief summary of the research and the achieved results, as well as possible future works.

2 RELATED WORK

This section presents some approaches that inspirated and fundamented the current research. The approaches are reported by chronogical order.

In the 1980s, (Braitenberg, 1986) prepared a hypothetical and pioneering study based on animals and insects, whose instinctive behaviors in their natural habitats influenced the author to describe these behaviors by means of psychological language in order to assimilate them to the behaviors that would be expected from a robot navigating an environment according to the hypothetical model. This assimilation resulted in the strategy known as the Braitenberg Algorithm.

In the 2010s, Generative Adversarial Networks (GAN) was presented as a technique in which a neural network becomes capable of generating samples from various types of data distribution, such as images, music, speech, and prose (Goodfellow et al., 2014). Subsequently, GAIL was developed in (Ho and Ermon, 2016), described by the authors as a general framework used to extract a policy directly from various data.

In the year 2015, the Trust Region Policy Optimization (TRPO) was conceived in (Schulman et al., 2015), an approach considered as an improvement of the baseline policy gradient. Already in 2017, (Schulman et al., 2017) proposed the Proximal Policy Optimization (PPO) as a policy gradient technique with superior performance to TRPO by having lower computational cost and a simpler implementation. After its inception in 2016, applications of GAIL in solving different problems and variations of the original algorithm have been proposed in the literature. In (Kuefler et al., 2017), the authors designed an extension of a GAIL for training recurrent policies in order to predict and simulate the behavior of human drivers accurately and which, in the end, showed robustness when dealing with perturbations in trajectories.

In 2019, (Zolna et al., 2019) presented TRAIL as a variation of GAIL to improve the poor performances obtained using the traditional algorithm in situations where the discriminating network occasionally focuses on irrelevant features. Another variation of GAIL was proposed to generate modeling of Car-Following behaviors and driving styles capable of simulating such behaviors accurately with respect to human drivers (Zhou et al., 2020). Next year, a method was created for learning human navigational behavior by inserting sample human trajectories into crowded public spaces in GAIL (Fahad et al., 2020), in which such trajectories were recreated by the authors in a 3D simulator. In (Zuo et al., 2020), a deterministic version of GAIL was designed to complete the motion planning task of a robot with higher speed and stability during learning compared to stochastic GAIL.

In (Couto and Antonelo, 2021), two variations of GAIL were generated to provide autonomous navigation for a vehicle located in a virtual urban environment of the CARLA simulator. Both versions succeeded in imitating the expert agent demonstration, but the one that used BC to increase the loss function showed superior training stability and time convergence. Another adaptation of GAIL was made to generate robustness improvement of the original technique (Pham et al., 2021). The original algorithm has also been adapted to generate locomotive autonomy to RAMIS systems, and has obtained promising results with tests in virtual simulations and on a real RAMIS platform (Pore et al., 2021).

In turn, the development of Co-GAIL emerged in (Wang et al., 2022) to enable the learning and extraction of a human-robot collaboration policy through behavioral samples of human-human collaborations. For the validation of Co-GAIL, scenarios such as a 2D strategy game and a delivery task between a human and a robot were simulated, and in addition, it was also tested on a task with an active human operator. A multi-agent GAIL was devised to solve the problems of sparse reward and local convergence when using a model trained by a reinforcement learning algorithm as the controller of an AUV (Mao et al., 2022). In the same year, a secondary study was elaborated with emphasis on the use of machine learning in path planning and control of autonomous mobile robots (Xiao et al., 2022).

3 NAVIGATION STRATEGIES

As mentioned in the Section 1, this research aims to develop two strategies for mobile robot navigation that are based on BC and GAIL. The idea is to use each strategy to a learning robot perform actions similar to those of the expert robot, and consequently travel the defined paths in a similar manner to the expert robot.

The performance of each strategy is evaluated and compared with each other in order to define which one has the behavioral policies closest to the expert's policy and to observe the level of success that the model of each strategy would have in leading the learning agent through the paths defined in the different proposed scenarios. To ascertain the adaptability of each strategy, simulations are also realized with four kinds of agent-environment interactions. As previously mentioned, it is also desired to verify the feasibility of integrating the Imitation algorithms in customized environments external to the Gym toolkit.

3.0.1 Behavioral Cloning

Behavioral Cloning (or Supervised Imitation Learning) is one of the most popular IL approaches. The task to be solved is treated as a supervised learning problem in which behavioral samples coming from the expert are grouped for training a model that, according to (Sheh et al., 2011), generalizes the examples of actions received as input. At runtime, the agent's controller uses it to evaluate new situations during its journey in the environment with the intention of performing the same types of actions that the expert would perform in that context.

The main objective during the training phase in a BC task is to minimize the loss function $L(a^*, \pi_\theta(s))$, where a^* represents an action of the expert and $\pi_\theta(s)$ symbolizes an action coming from the learner (model). The value returned by the loss function is inversely proportional to the correctness of the action generated by the learner. According to (Antonelo and Schrauwen, 2010), the accumulation of cascading errors and the need for a high amount of data to avoid the composition error caused by the co-variable change are the main negative factors in applying this approach.

3.0.2 Generative Adversarial Imitation Learning

GAIL was conceived in (Ho and Ermon, 2016) as a new learning technique based on the IRL and GAN approaches, in which the occupancy measure (distribution of pairs of states and actions performed by an agent while exploring its environment via a policy π) of an expert agent ρ_{π_ε} represent the true dataset, while the measure of a learning agent ρ_{π_θ} is equivalent to all the data created by a generative neural network.

A discriminating net receives the pairs of states and actions (pairs (s,a)) from ρ_{π_ε} and ρ_{π_θ} to determine which of them are false or true (i.e., whether they belong to ρ_{π_θ} or ρ_{π_ε}, respectively). In each iteration of training, the discriminator sends feedback to the generative net based on the determinations made for each pair obtained. The generative tends to improve as it receives feedback, in order to create pairs that are increasingly difficult for the discriminator to distinguish.

The goal of GAIL is to promote constant competition between these two nets, so that ρ_{π_θ} is similar to ρ_{π_ε} when the stopping criteria for training is reached. In this way, the generating net is able to provide pairs (s,a) very close to those that belong to the original dataset. In the literature it is common to use the policy gradient methods TRPO and PPO for training a generative network, while an MLP can be used as the discriminating network. More information about GAIL can be found at (Ho and Ermon, 2016).

3.1 Strategies Development

The CoppeliaSim simulator (Rohmer et al., 2013) was used for the construction of the simulation scenarios and execution of the experiments in its 4.3 release using dynamic engine Bullet 2.83 and balanced dynamic setting. To assist in the construction of the strategies, the BC and GAIL implementations contained in the 0.2 release of the Imitation API (Wang et al., 2020) were used for training and the consequent extraction of the policies. The development flow for each strategy is shown in Figure 5.

In each proposed scenario, a simulation is performed in which the expert robot records its pairs (s,a) in a log file while performing actions to travel along a given trajectory. The pairs are used as input for training each strategy. If the agent is a Braitenberg vehicle, a filter must be applied to its log file to remove identical pair (s,a) that would cause redundancy, noise and increase training time.

In the next step, the expert file is used as the training dataset for the model referring to the desired navigation strategy. Several experiments containing different parameters and hyper-parameters for the algorithms are carried out continously for the two strategies in each of the scenarios until the best results is achieved for each model, considering the available resources of the Imitation API. Upon finding such results, a simulation is realized with the model itself as

Table 1: Minimum and maximum values for the state space and action space of each scenario.

Scenario / Parameter	S1 e S2	S3 e S4
observation_space_lows	[-7.11, -4.72, -3.14]	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
observation_space_highs	[7.11, 4.72, 3.14]	[0.1, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05]
action_space_lows	[0.0, 0.0]	[0.0, 0.0]
action_space_highs	[2.0, 2.0]	[4.5, 4.5]

Table 2: Algorithms parameters in each scenario.

	Parameters	Scenarios			
		S1	S2	S3	S4
BC	batch_size	6	12	219	556
	total_timesteps	20.000	60.000	40.000	500.000
GAIL	demo_batch_size	6	12	219	556
	total_timesteps	80.500	500.000	2.500.000	10.000.000
PPO	batch_size	64	56	219	278
	n_steps	64	112	438	556
	ent_coef	0.0	0.001	0.001	0.001

the driver for the learning robot's actions. During the driving, the pairs (s,a) provided by the model are also registered in a file so that its actions are compared with the expert's actions by means of the proposed evaluation methods.

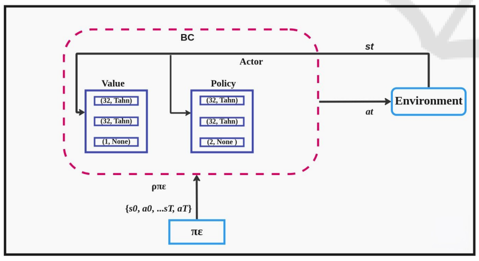

Figure 1: Imitation's BC network architecture and its interaction with the environment.

Next, the simulation scenarios proposed for this work are described. Each of them has a continuous action space and state space. The ideal policy to be extracted and the environment for each scenario are deterministic. The learning of the agents is always model-free. As the robot is always a suspension state when it reaches its final state, the tasks are exclusively episodic and with a finite horizon for each scenario.

The BC and GAIL implementations are configured according to the architectures present in Figures 1 and 2. Aiming to find an equilibrium for all pro-

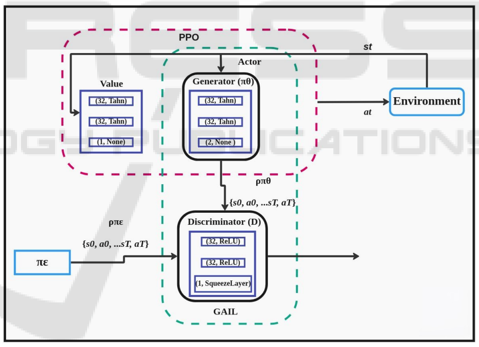

Figure 2: Imitation's GAIL network architecture and its interaction with the environment. The PPO is used as the generative network.

posed scenarios and thus avoid possible overfitting or underfitting after training, all experiments in this work use two hidden layer MLPs with 32 units each for both BC and GAIL. The activation functions used, the outputs generated by each MLP, and the interaction of each approach with the environment are represented in Figures 1 and 2. For all MLPs, the policy and value neworks are shared.

3.1.1 Scenarios 1 and 2

For scenarios S1 and S2, a 750 m² rectangular 3D environment was built with walls around it to prevent the robot from falling off the scenario floor. The virtual version of Pioneer-P3DX that is included in CoppeliaSim is used as the deterministic agent for both scenarios, which moves through them via commands sent from the API to the simulator. A Pioneer state s is represented by a vector of real numbers [x, y, $R_z(\gamma)$] that represents the coordinates and orientation angle in radians of the agent. An action a is composed by joining the velocities of the robot's two wheels into a vector [v1, v2]. Table 1 presents the minimum and maximum values considered for the space and action states. From the expert, 6 and 12 pairs (s,a) were respectively recorded in log files for S1 and S2, where each pair was considered essential for a effiicency performance of each model after training. The paths defined for the expert to follow can be visualized in Figures 7 and 8.

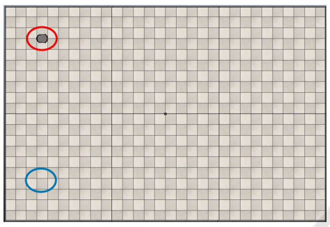
Figure 3: Environment of Scenario 1 with Pioneer-P3DX in its starting position delimited by the red dot and with orientation to the right side. Scenario 2 is nearly identical to the first one, but the robot's starting position is located inside the blue dot with the orientation turned upside.

3.1.2 Scenarios 3 and 4

For scenarios S3 and S4, a rectangular environment similar to the predecessor scenarios was built, but with a size of 375 m², walls positioned on all four sides and other parts of the environment. The robot used for these scenarios is the ePuck. CoppeliaSim's version of ePuck has a light sensor, a distance sensor, and a native implementation of a Braitenberg algorithm, which together cause the robot to move independently along a given path while identifying and avoiding possible obstacles.

The light sensor assists in planning a path by constantly checking how close the ePuck is to a light source, simulating the fear instinct (Braitenberg, 1986). If the ground traversed by the agent is dark (i.e. partial or total absence of light), the stimulus received is insufficient and no changes is set in wheel speed. However, the light coming from the ambient floor stimulates the sensor whenever the robot tries to go out of the path, resulting in a change in the speed of the wheels to make it follow the path orientation again.

If the distance sensor identifies possible obstacles in the vicinity of the agent, signals are sent to change the speed of the wheels in order to perform the appropriate evasive maneuvers. A black path was built to be traveled by the ePuck until it almost reaches a complete turn through it, representing the S3. The reason for the chosen color is for the functioning of the light sensor stimulus and, consequently, the fear instinct that is an essential part of the algorithm. S4 is similar to S3, but four obstacles were added (Figure 4) to check not only the movement inside the path, but also the ability of the strategies to identify them and lead the agent to realize necessary detours in a similar way to the expert robot.

For a Braitenberg vehicle, a state s of the ePuck is represented by a vector of 11 floating-points and 32-bit , in which the first 3 represent the data received by the light sensor and the remaining ones depict the value of each of the 8 beams of the distance sensor. Again, an action a is composed of a vector containing two numeric values referring to the speeds of each wheel of the agent.

The parameters and hyperparameters that returned the best performances in each scenario are shown in Table 2. Only the parameters and hyper-parameters that were changed are mentioned in the table, while the others used implicitly in the algorithm were ignored because they kept their default values defined by Imitation (Wang et al., 2020). The allowed numerical range for the states and actions of S3 and S4 are also in Table 1. A total of 219 and 556 pairs (s,a) of the expert were recorded for S3 and S4, respectively. Looking at the relationship between the total number of pairs and the number of features, it can be concluded that all the scenarios proposed in this work are low-dimensional.

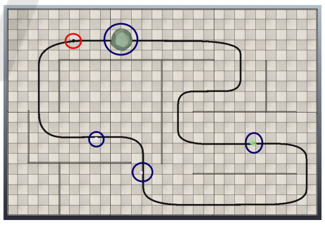
Figure 4: Environment of Scenario 4 with ePuck in its starting position. The red circle shows the initial position of the agent while the blue ones mark the location of each obstacle inserted. Scenario 3 is nearly identical to S4 but it has no obstacles.

3.2 Evaluation Methods

The Mean Squared Error (MSE) and Mean Absolute Error (MAE), two traditional error metrics of regession type machine learning problems, are used to make an accurate evaluation of which strategy is able to generate actions closer to the ones executed by the

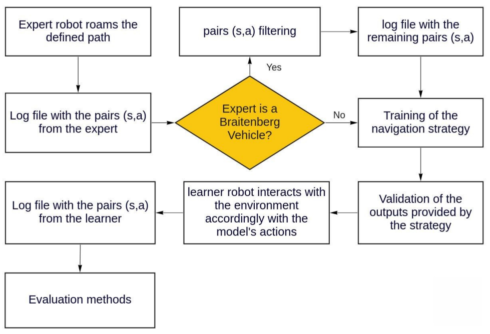
Expert robot roams the defined path
pairs (s,a) filtering
log file with the remaining pairs (s,a)
Yes
Log file with the pairs (s,a) from the expert
Expert is a Braitenberg Vehicle?
No
Training of the navigation strategy
Log file with the pairs (s,a) from the learner
learner robot interacts with the environment accordingly with the model's actions
Validation of the outputs provided by the strategy
Evaluation methods

Figure 5: The development flowchart.

expert. The calculations performed in MSE and MAE are represented, respectively, in the equations 1 and 2. In both, N symbolizes the total of pairs (s,a) of a scenario, y_i is the optimal action coming from the expert and $\hat{y}_i$ represents the action predicted by the trained model of the strategy.

$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2 \tag{1}$$

$$\frac{1}{N}\sum_{i=1}^{N}|y_i-\hat{y}_i| \tag{2}$$

Considering the characteristics of the proposed strategies, it is possible to state that a comparison between the paths executed by the learner and the expert is as important as measuring the error between the actions recorded by both. The metrics are able to inform which of the strategies have the lowest error rate, but they can not accurately determine whether a model that provides outputs considered poor or average can still make the learning robot follow the trajectory determined by the expert in a similar manner.

It is only by comparing the pairs (s,a) of the expert with those of the learner that it will be possible to analyze whether the path taken by the latter agrees with that of the former, regardless of how close the outputs generated by the learner's model are. Next, the pairs of the expert and the models are displayed in graphs generated in order to perform the comparison.

4 RESULTS

The construction of the proposed scenarios was feasible due to the resources available in CoppeliaSim (Figures 3 and 4). The BC and GAIL algorithms from the Imitation API were successfully integrated into each scenario. It can be stated that the adaptability of the API in environments external to Gym was sufficient, so the entire development flow presented in Figure 5 could be performed in each scenario.

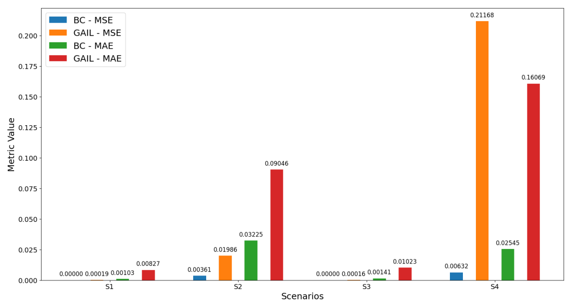

Figure 6: Comparison of MSE and MAE for BC and GAIL strategies in each scenario.

4.1 Error Metrics

In general, both strategies generated similar actions to those that were generated by the expert, as can be observed in Figure 6. Although GAIL presented a considerably low MSE and MAE for all scenarios, BC managed to obtain even lower values and, consequently, more similar actions to those that were produced by the expert in each of them. Note that, with the exception of S4, GAIL's MSE was closer to that of the BC while the MAE distance between the two was greater.

4.2 Path Comparisons

Observing Figure 7, it is possible to affirm that GAIL obtained a satisfactory learning in S1 by driving the robot along the same path defined by the expert even if in the final state change, it directed the agent to a state a little higher than the ideal one. The BC was able to learn the locomotive behavior. Moreover, the strategy steered the robot to a higher state, but closer to the original one when it is compared to the one GAIL had done.

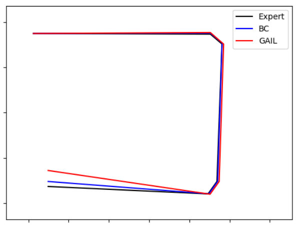

Figure 7: Path of Scenario 1 executed by the expert and the trained BC and GAIL models.

In S2, GAIL performed similarly to the expert for most of the trajectories. The differences in the actions returned by the model in some of the states resulted in slight deviations in orientation along the path, but they did not prevent the agent from reaching its destination. In turn, the BC model reproduced the locomotion along the trajectory in an identical manner to that of the expert, as can be seen in Figure 8. In S3 (Figure 9), both BC and GAIL performed satisfactorily and were able to reproduce the path taken by the ePuck guided by a Braitenberg algorithm. Note that the higher amount of MSE and MAE obtained for the GAIL strategy (Figure 6) did not prevent it from successfully leading the agent along the defined path.

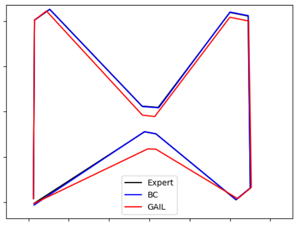

Figure 8: Path of Scenario 2 executed by the expert and the trained BC and GAIL models.

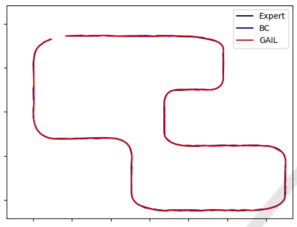

Figure 9: Path of Scenario 3 executed by the expert and the trained BC and GAIL models.

From the results of the fourth scenario in Figure 10, it can be seen that the BC was able to travel the defined path and perform the same evasive maneuvers as the expert when it approaches the obstacles. However, GAIL was not able to make ePuck perform the evasive maneuver as soon as it made contact with the first object in the scenario. The model was able to move the agent along the trajectory at the beginning, but even with considerably low MSE and MAE values, the actions it sent to the robot's wheels did not make it perform an evasive maneuver when the sensors identified a nearby obstacle. Instead, the actions returned by GAIL led the robot to maintain forward motion until it came into friction with the first obstacle and remained stagnant as a consequence.

5 CONCLUSIONS

The navigational autonomy of a mobile robot inserted in a given environment remains one of the main research trends in robotics and automotive studies. Thus, this project proposes two navigation strategies for land mobile robots, inspired by Imitation Learning techniques. One of them is based on BC and the other on GAIL, the goal of building such strategies involves

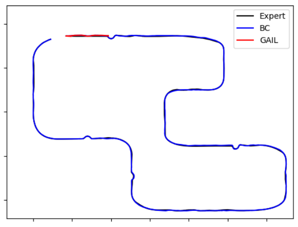

Figure 10: Path of Scenario 4 executed by the expert and the trained BC and GAIL models. The four curves present on the path represent the direction the agent took to avoid each obstacle.

not only learning locomotive behaviors specific to a mobile agent, but also analyzing and comparing the performance between them when driving this agent through different environments. Furthermore, it is desired to know the level of robustness in learning each strategy when different agents and navigation methods are used.

BC and GAIL implementations were imported from the Imitation API to aid in the development of this work, and the CoppeliaSim simulator was used to build four different virtual environments. Pioneer was defined as the agent for the first two scenarios that is guided through the paths by means of commands sent to the simulator. ePuck was chosen as the agent for the remaining scenarios that moves along the paths according to a Braitenberg algorithm.

Error metrics and the comparison of paths traveled between the expert agent and the learner agent were applied as methods for evaluating the results obtained in each strategy. The analysis of the achieved results made it possible to state that BC obtained lower MSE and MAE and a higher accuracy to drive the robot in all scenarios compared to GAIL. However, the use of the GAIL-based strategy with the parameters defined in Table 2 proved fully feasible for the first three scenarios. Considering all the obtained results, it can be stated that the BC and GAIL implementations coming from Imitation are adaptable for use in environments, agents and navigation forms external to the Gym toolkit.

In order to improve the achieved results, one future work aims to consider different amounts of hidden layers and neurons for the BC and GAIL architectures. Ascertaining the level of impact on performance when using other activation functions in the BC and GAIL MLPs is also an interesting topic for the future. To further evaluate the robustness of both strategies, it pretend to adapt and run the training of both strategies on learning other robot navigation techniques, such as SLAM or A Star. Scenarios that utilize a UAV agent in airspace environments or UUV in aquatic locations would also be considerable additions. Developing all of these additional ideas requires time and effort, but the results obtained could unravel the reason for GAIL's failure to learn obstacle avoidance in scenarios as S4, as well as enrich the knowledge of each strategy's capabilities and applicability.

ACKNOWLEDGEMENTS

The authors would like to thank the National Council for Scientific and Technological Development (CNPq) (421548/2022-3 and 88881.753469/2022-01) and Araucaria Foundation (17.633.124-0) for their financial support on this work. We would also like to thank the Manna Team for support, learning and collaboration of its members.

REFERENCES

Antonelo, E. A. and Schrauwen, B. (2010). Supervised learning of internal models for autonomous goal-oriented robot navigation using reservoir computing. In *2010 IEEE International Conference on Robotics and Automation*, pages 2959–2964. IEEE.

Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. MIT press.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Couto, G. C. K. and Antonelo, E. A. (2021). Generative adversarial imitation learning for end-to-end autonomous driving on urban environments. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE.

Fahad, M., Yang, G., and Guo, Y. (2020). Learning human navigation behavior using measured human trajectories in crowded spaces. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11154–11160. IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573.

Kuefler, A., Morton, J., Wheeler, T., and Kochenderfer, M. (2017). Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 204–211. IEEE.

Mao, Y., Gao, F., Zhang, Q., and Yang, Z. (2022). An auv target-tracking method combining imitation learning and deep reinforcement learning. *Journal of Marine Science and Engineering*, 10(3):383.

Pham, D.-T., Tran, T.-N., Alam, S., and Duong, V. N. (2021). A generative adversarial imitation learning approach for realistic aircraft taxi-speed modeling. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2509–2522.

Pore, A., Tagliabue, E., Piccinelli, M., Dall'Alba, D., Casals, A., and Fiorini, P. (2021). Learning from demonstrations for autonomous soft-tissue retraction. In *2021 International Symposium on Medical Robotics (ISMR)*, pages 1–7. IEEE.

Rohmer, E., Singh, S. P. N., and Freese, M. (2013). Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sheh, R., Hengst, B., and Sammut, C. (2011). Behavioural cloning for driving robots over rough terrain. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 732–737. IEEE.

Wang, C., Pérez-D'Arpino, C., Xu, D., Fei-Fei, L., Liu, K., and Savarese, S. (2022). Co-gail: Learning diverse strategies for human-robot collaboration. In *Conference on Robot Learning*, pages 1279–1290. PMLR.

Wang, S., Toyer, S., Gleave, A., and Emmons, S. (2020). The `imitation` library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation.

Xiao, X., Liu, B., Warnell, G., and Stone, P. (2022). Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, pages 1–29.

Zhou, Y., Fu, R., Wang, C., and Zhang, R. (2020). Modeling car-following behaviors and driving styles with generative adversarial imitation learning. *Sensors*, 20(18):5034.

Zolna, K., Reed, S., Novikov, A., Colmenarejo, S. G., Budden, D., Cabi, S., Denil, M., de Freitas, N., and Wang, Z. (2019). Task-relevant adversarial imitation learning. *arXiv preprint arXiv:1910.01077*.

Zuo, G., Chen, K., Lu, J., and Huang, X. (2020). Deterministic generative adversarial imitation learning. *Neurocomputing*, 388:60–69.