Generative Adversarial Imitation Learning for Steering an Unmanned Surface Vehicle

Alexandra Vedeler*[1] and Narada Warakagoda[2]

[1]Norwegian University of Science and Technology (NTNU)
[2]Norwegian Defence Research Establishment (FFI)

1 Introduction

We consider the design of an autonomous system for steering an unmanned surface vehicle (USV) using an end-to-end approach, where the system directly generates action parameters based on the sensory data. Compared to the traditional module-by-module design paradigm, this approach has the potential of making the overall system more compact, efficient and accurate.

Artificial Neural Networks (ANN) that offer a high level of expressive power is a suitable candidate for implementing such a system. They are able to handle highly nonlinear relationships between their input and output [12, 4], an ability which is necessary in order to perform end-to-end steering in USVs. Systems that couple deep ANNs with Reinforcement Learning (RL), have also proved to be able to learn complex tasks [14, 11, 10, 8, 3, 7, 15]. However, RL requires a known reward function to guide its training. We argue that such a function can be highly difficult to craft manually for USV steering as certain maneuverings do not express a clearly weighted cost structure [1]. Alternatively, one can combine RL with Inverse Reinforcement Learning (IRL), where the goal is to learn a reward function from a set of demonstrations typically performed by a human expert. This combination of RL and IRL forms an Imitation Learning (IL) system that can learn from demonstrations.

In this paper, we consider a USV that is equipped with a Radar sensor and study the problem of generating a single action parameter, *heading*. We apply an imitation learning algorithm with IRL-adjacent approach, known as generative adversarial imitation learning (GAIL) to develop an end-to-end steering model for a scenario where avoidance of an obstacle is the goal. The performance of the system is studied for different design choices and compared to that of a system that is based on pure RL.

The organization of the paper is as follows. In Section 2, we outline some of the important work related to our study. Section 3 provides a short description of the theoretical background of the study. Details of the method are presented in Section 4. Experiments and their results are in Section 5. Finally, the concluding remarks are given in Section 6.

2 Related Work

A well-known example of deep learning in steering is PilotNet [2], a deep ANN which maps the raw visual input to steering parameters of a self-driving car. However, this sort of behaviour cloning treats steering more like a deep ANN classification problem, a type of supervised learning. ANNs demand large datasets in order to be trained well [4] and in a supervised learning approach, the data must be labeled before training. PilotNet trained on 6 hours' worth of video and sensor data from a human driving a car. Such demands make this approach problematic in the case of USV steering, where collection of large amounts of labelled data is relatively more costly. In addition, this approach can lead to compounding errors. This means that a small mistake on the part of the policy can place the system into states that lie outside of the distribution in the training data, and hence supervised learning will in general not gener-

*Corresponding Author: alexandra.s.vedeler@gmail.com

https://doi.org/10.7557/18.5147

ate a policy with good long-horizon performance [8]. Another problem of behavior cloning is that it heavily relies upon features such as road edges in learning the policy. For the case of a USV, such clues are not available.

[10] used a policy search RL approach, specifically the Deep Deterministic Policy Gradient method [9], to find the desired policy for straight-path following for an underactuated marine vessel exposed to unknown ocean currents. As described in the article, the approach is model-free, requiring no prior knowledge of the system it is assigned to control. Another example of RL in USV steering is [3] who propose a deep RL approach for obstacle avoidance. As RL approaches, these require a pre-made reward function.

There are also some examples of the use of Inverse Reinforcement Learning (IRL) in steering. One such example is [16], who used a Maximum Entropy-based [17], non-linear IRL framework with Fully Convolutional ANNs to represent the cost model underlying expert driving behavior. However, we found no references for use of IRL in the task of USV steering.

3 Background

3.1 Reinforcement Learning

In RL, the process of the agent interacting with its environment and the resulting reward is formulated as a Markov Decision Process (MDP), a tuple $< \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R} >$. At each time step $t = 0, 1, 2, 3...$, the agent experiences the state of the environment, $\mathbf{x} \in \mathcal{X}$, and must decide on some action, $\mathbf{u} \in \mathcal{U}(\mathbf{x})$. A policy π maps the state $\mathbf{x}$ to the action $\mathbf{u}$. The choice of action results in the environment transitioning into a new state, $\mathbf{x}'$, and the agent receiving a scalar reward, $r \in \mathcal{R}$, as a consequence of this transition. The transition itself is modeled by the function $\mathcal{P}$ [15].

RL attempts to find the optimal policy, π^*, for action selection at each time step. Thus we want to maximize the expected discounted reward of the policy π:

$$\eta(\pi) = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{u}_0, \ldots}\Big[\sum_{t=0}^{\infty} \gamma^k r_t\Big] \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor.

The policy to be optimized can be parameterized and trained through the use of an ANN, a process dubbed Deep RL (DRL). Policy Gradient Methods are DRL methods which optimize a performance objective, $J(\pi_{\boldsymbol{\theta}})$, by finding a good policy, $\pi_{\boldsymbol{\theta}}$, using variants of stochastic gradient ascent with respect to the policy parameters $\boldsymbol{\theta}$. In this paper, we implement one such method called Trust Region Policy Optimization (TRPO) [13].

The TRPO algorithm establishes a trust-region through KL-divergence where the expected improvement of a new policy can be approximated locally as $L_{\boldsymbol{\theta}_{old}}(\boldsymbol{\theta})$, resulting in the following update rule:

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, L_{\boldsymbol{\theta}_{old}}(\boldsymbol{\theta}) \\ &\text{subject to } \bar{D}_{KL}(\boldsymbol{\theta}_{old}, \boldsymbol{\theta}) \leq \delta \end{aligned} \tag{2}$$

where $\bar{D}_{KL}(\boldsymbol{\theta}_{old}, \boldsymbol{\theta})$ is the average KL-divergence, $\boldsymbol{\theta}_{old}$ represent the parameters that make up the policy, and $\boldsymbol{\theta}$ the policy parameters which are to be improved upon.

3.2 Imitation Learning

RL methods are limited by their need for a reward function that captures the essence of the task at hand. IRL can help overcome this limitation. In IRL, the goal is to optimize a reward function based on observations of the states and the corresponding actions performed by a human expert. Combining IRL with RL, thus creates a form of Imitation Learning (IL) in which the goal is to optimize a policy through observations of expert performances and without the need for a pre-constructed reward function.

In this paper, we utilize the Generative Adversarial Imitation Learning (GAIL) algorithm [6] which share the spirit of IRL. The main idea of GAIL is that it does not learn a proper reward function, but utilizes a reward signal in the RL loop. In GAIL, a discriminator (classifier) provides this signal and the overall IL algorithm resembles a Generative Adversarial Network (GAN) [5]. Thus an IL system based on GAIL consists of two networks: a policy network and a discriminator. The policy network is trained

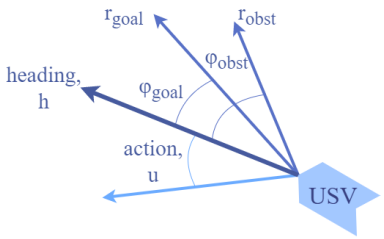


Figure 1: Illustration of positional vectors as defined in our system. All observational and action angles are given relative to the current heading.

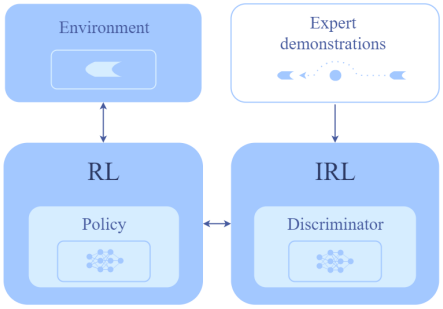


Figure 2: Simplified illustration of the IL system combining RL and IRL.

using feedback from a discriminator network which indicates whether the agent is acting as the expert would do in the given situation. The discriminator network is simultaneously trained to differentiate between the observation-action pairs of the agent and those of the expert. By playing off of each other, both networks improve their execution of their respective tasks and thus the agent will eventually learn to behave like the expert.

4 Method

Our goal is to train an end-to-end policy for obstacle avoidance with a USV moving at constant speed. The policy outputs the heading (i.e. the required angle of the USV's direction of motion) based on Radar sensor inputs.

We wish for the agent to learn to maneuver the USV around an obstacle that obstructs its path. In this scenario, we assume that the USV is moving along a straight line towards a goal position and that an obstacle, in our case a stationary pole, lies somewhere on this straight line. Thus the agent must maneuver around this object and back towards its original goal. In order to perform the task, the agent must observe its state, $\mathbf{x}$, and choose an action $\mathbf{u}$. In our system, we define $\mathbf{x} = [I, r_{goal}, \phi_{goal}]$. Here I represent the 2D array of a simplified radar image, either generated at runtime or provided from the expert demonstrations, while the vectors r_{goal} and ϕ_{goal} denote the distance vector from the USV to the goal position. r_{goal} and ϕ_{goal} are given in polar coordinates and are expressed relative to the current heading of the USV. This first-person view of the world makes the agent invariant to rotation, meaning the policy does not need to treat an obstacle it is approaching from the north differently than one it is approaching from the east. Further, this makes training easier and the resulting policy more general. Because of this same reasoning, we also express the action, $\mathbf{u}$, as a degree of adjustment to the current heading. This type of observation and action is illustrated in Figure 1.

In addition to this we performed additional experiments with the use of the positional vector for the obstacle. In this state-formulation, the agent was provided the position of the obstacle directly without needing to extract the information from an image, resulting in a simpler task.

4.1 Imitation learning

We trained our system (and hence the policy) using an imitation learning approach based on the GAIL algorithm. A simplified illustration of our system is depicted in Figure 2. As seen in Figure 2, there are two trainable networks: the policy network used in the RL loop and the discriminator network used in the IRL procedure.

The policy network is comprised of a convolutional layer, which allows for the extraction of information from the radar image, followed by two fully connected layers. Because the state vector consists of both a matrix part and a vector part, we input the second part of the state late, leaving only the image to be

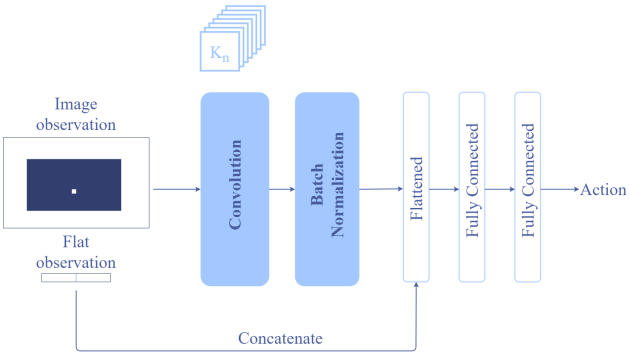


Figure 3: Simplified illustration of the policy network. An example of a generated radar-like image is depicted as input.

convolved before merging the flattened result with the vector input. The convolution layer consists of 10 kernels of size 7×7, performing convolution with a stride of 4 and *relu* activation, while the fully connected layers consist of 400 neurons for the first layer and 300 for the second. Based on the output of the last layer a scalar action is generated. We considered two types of actions: A continuous value drawn from a Gaussian and a discrete value drawn from a 40-way categorical distribution. All actions are scaled into the region $(-30°, +30°)$. A simplified illustration of the policy network is depicted in Figure 3. The discriminator network is similar to the policy network except that it also takes the action as input. For the experiments using positional vectors for the obstacle position instead of radar images, the convolution layer is dropped while the fully connected layers remain the same.

The system using IL through GAIL relies on observations of the task being performed by an expert. We collected a set of trajectories of a manually driven USV[1] performing obstacle avoidance. This resulted in 35 trajectories with states and actions comprising of recorded sensory input from the USV during the performance of the expert. These recordings provided us with radar data and positional data.

While the expert observations were collected using a physical USV, the training was performed by simulation, simulating the USV acting in the environment[2] and generating images to serve as radar images.

4.2 Pure reinforcement learning

Even though we have selected the collision avoidance task for studying IL approach, we can also manually design a reward function for this task because of its simplicity. Therefore, we also trained the system using pure RL, results of which can serve as a reference. For this, we constructed a handmade reward function based on Gaussian functions, $\mathcal{G}(x, \sigma, \mu) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}((x-\mu)/\sigma)^2}$, as these are smooth, symmetrical and differentiable functions whose output lies between 0 and 1. Using this, we define our reward function $r(p_U, p_O, p_G)$ for this problem as:

$$r_O(p_U, p_O) = -2\mathcal{G}(p_U - p_O, \sigma_O, 0) \tag{3}$$

$$r_G(p_U, p_G) = -1 + \mathcal{G}(p_U - p_G, \sigma_G, 0) \tag{4}$$

$$r(p_U, p_O, p_G) = r_O(p_U, p_O) + r_G(p_U, p_G) \tag{5}$$

where p_U, p_O, and p_G are the positions of the USV, obstacle and goal respectively. We deem avoiding the obstacle as more important than quickly reaching the goal and thus weight this penalty double. We selected the standard deviations $\sigma_O = 5m$ and $\sigma_G = 100m$ considering the relative importance.

5 Experiments and Results

Once the training is completed using IL and pure RL, validation was performed by executing the policy and measuring the results. 15 episodes were performed, each with a different, but fixed, starting position. The USV was placed between 50 to 100 meters away from the obstacle. The USV was placed at an angle spanning from 0 to 360 degrees from the obstacle, facing it. The goal was placed 220 meters from the USV, at the opposite side of the obstacle. We deemed an episode a success if the agent was able to

[1]The USV used for data collection was provided by FFI.

[2]A simulation of the dynamics of the USV was provided by FFI

Setup	S_{IL}	S_{RL}
Gaussian-positionalObs	86.6%	100%
Categorical-positionalObs	100%	100%
Gaussian-RadarObs	53.3%	100%
Categorical-RadarObs	86.6%	100%

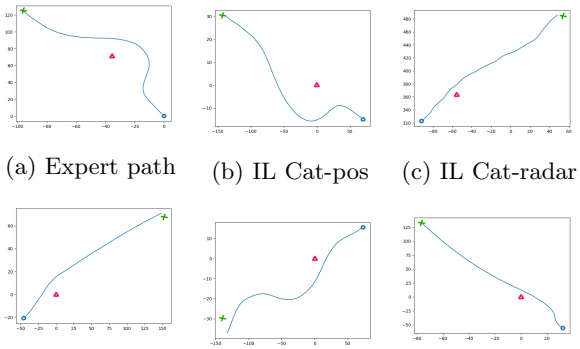
Table 1: Success rates S_{IL} and S_{RL} of the different system setups for IL and pure RL respectively.

get within 10 meters from the goal while never being closer to the pole than 10 meters at any point during the episode.

We considered four setups, depending on the observation type (positional or Radar) and the action variable type (Gaussian or Categorical). The performances of all 4 set-ups for both IL and pure RL are summarized in Table 1.

Overall, the results of our experiments have been mostly positive. Even though some unsuccessful episodes were recorded, the lowest success rate being 53.3%, all set-ups showed a clear grasp of the task, even when their performances were not accurate enough to be marked successful. However, some setups showed somewhat oscillatory behavior, most notable in those using the categorical representation of the policy network.

Both the RL and IL systems generate paths that aim for the goal position and turn to avoid the obstacle (see Figure 4). However, the IL system occasionally misses the goal position by more than 10 meters or passes the obstacle with too close proximity, its closest proximity being 5 meters at the worst.



(a) Expert path (b) IL Cat-pos (c) IL Cat-radar

(d) RL Gauss-radar (e) IL Gauss-pos (f) IL Gauss-radar

Figure 4: Example paths taken by the USV: (a) Example of a path taken by an expert (b) Successful episode of the IL setup with positional observations with categorical policy (c) A successful episode of the IL setup with radar observations with categorical policy (d) A successful episode of the RL setup with radar observations with gaussian policy (e) Unsuccessful episode of the IL setup with positional observations with Gaussian policy (f) Unsuccessful episode of the IL setup with radar observations with Gaussian policy. The blue circle marks the starting position, the green X marks the goal position and the red triangle marks the obstacle.

6 Conclusion

We have presented a system which learns an end-to-end steering model, through GAIL based Imitation Learning (IL) and the use of a set of expert demonstrations. This system uses Deep Learning techniques to learn a policy that maps input observations to steering actions. For the purpose of comparison, we have also presented a similar system trained through pure RL together with a manually crafted reward function. We have tested our systems on two types of observations, radar-like images and obstacle positions. Both systems show a clear understanding of the task at hand and are able to steer towards a target position while avoiding collision with an obstacle.

In comparison, the RL system performed at the highest accuracy overall, scoring 100% on our predetermined success measure. While the problem of learning from demonstrated behavior seems to be the more difficult task, resulting in lower accuracy, the IL system produces results that indicate it is able to grasp the concept of the task and that in many ways are on par with the RL system. We deem this to be promising for future use in tasks that are not as easily described by a reward function. While avoidance of stationary obstacles using radar observations seems to be a task which can be described by a manually crafted reward function without much difficulty,

other USV tasks such as dynamic obstacle avoidance and docking may not be as simple to capture. These tasks may benefit more from the IL approach. But further work is necessary to verify that claim.

References

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pages 1–8, 2004. `https://ai.stanford.edu/~ang/papers/icml04-apprentice.pdf`.

[2] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. D. Jackel, and U. Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *CoRR*, abs/1704.07911, 2017.

[3] Y. Cheng and W. Zhang. Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. *Neurocomputing*, 272:63–73, 2018.

[4] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27:2672–2680, 2014. `https://arxiv.org/abs/1406.2661`.

[6] J. Ho and S. Ermon. Generative adversarial imitation learning. *CoRR*, 2016.

[7] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[8] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17, 2016.

[9] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[10] A. B. Martinsen and A. M. Lekkas. Straight-path following for underactuated marine vessels using deep reinforcement learning. *IFAC-PapersOnLine*, 51(29):329–334, 2018.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[12] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003.

[13] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, 2015.

[14] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[15] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction, 2nd edition*. MIT Press Ltd, 2017.

[16] M. Wulfmeier, D. Z. Wang, and I. Posner. Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2089–2095, 2016.

[17] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the National Conference on Artificial Intelligence*, volume 3, pages 1433–1438, 2008.