# Autonomous Car Parking
## CS7IS2 Project (2021/2022)

Mayuresh Shelke, Ming Jun Lim, Lalu Prasad Lenka and Samridh James

shelkem@tcd.ie, limm5@tcd.ie, lplenka@tcd.ie, jamessa@tcd.ie

**Abstract.** Motivation behind the work, high level description of the problem, how it was solved by the proposed algorithms.

**Keywords:** soft-actor-critic, behaviour cloning, evolutionary algorithm, parking

## 1 Introduction

The Autonomous Vehicle (AV) industry is one of the most active and promising area in the recent years with many ongoing research to make it fully autonomous [23]. The International Society of Automotive Engineers (SAE) proposed a six-degree autonomy scale with level 0 as no automation and level 5 as full automation without any human interventions [21]. The current level of autonomy scale achieved today is between level 2 and 3 requiring some assistance from the driver and limited to ideal conditions.

Car parking is challenging and disliked by most drivers due to time required searching for spaces, risk of scratching vehicle, safety of pedestrian, etc [2][8]. It requires the driver to estimate the space required for the car and manoeuvre it into available space by controlling the steering angle and accelerator. While in fact the traffic report statistics survery results found in 12 million traffic accidents, there are about 10,000 traffic accidents occurring in the parking lots with many more number of accidents not reported [25].

Most vehicles are parked 95% throughout the lifetime [4]. Parking is required by every driver and applying AV will greatly improve the quality of life and ease of drivers. It increases driving safety by utilising various sensors to understand the environment surrounding vehicle and park at the designated space avoid collisions due to lack of human experiences [25]. Other potential application for autonomous car parking is saving spaces in car parking lots by parking in an optimal grid resulting in a much efficient parking lot capable of more car [4].

The highway-env github repository[1] contains a collection of autonomous driving and tactical decision-making tasks environment [12]. The parking environment[2] is selected for the project which is a goal-conditioned continuous control task in a given space with a vehicle aiming to reach the destination point. The

[1] https://github.com/eleurent/highway-env
[2] id of "parking-v0"

parking environment uses a grid like layout similar to actual car parks. the vehicle requires input of the steering angle and accelerator. Policy based algorithm is commonly used for autonomous vehicles instead of value based algorithm due its large action decision requiring huge memory [22]. The selected algorithms to be investigated in the project are Soft-Actor-Critic (SAC) from the reinforcement learning family, behaviour cloning from the imitation family and evolutionary algorithm from the genetic population-based family.

## 2 Related Work

The autonomous vehicle control system consists of mission trajectory planning system generating path for vehicle to reach target, navigation system detecting environment and either moving or fixed target and trajectory tracking controller to navigate the vehicle accordingly [13]. Path planning problem is the main subject of most autonomous vehicle studies. Prior to path planning, the algorithm needs to know the state of the vehicle such as its position and direction. GPS is commonly used to deduce the state by using signals Line Of Sight (LOS) from positioning satellite but not possible in the context of indoor parking lots [6] and high-rise buildings [18]. Instead other data such as vision-based sensors and LiDAR sensors are used to monitor the state of vehicle [3]. The works related to autonomous car parking uses different inputs and types algorithm families, this section mainly focuses on selected algorithms mentioned in Section 1.

Deep Learning (DL) combined with Reinforcement Learning (RL) enables us to build machine learning models which does not require labelled data to learn about a given task. Reinforcement Learning enables an artificial agent to learn optimal strategy from the environment through trial and error [27]. RL formalizes the premise that rewarding or penalizing an agent for its action increases the likelihood that the agent will repeat or avoid respective behaviour in the future.

Deep Reinforcement Learning (DRL) is useful when the agent needs to make a decision based on more than one contributing element or high-dimensional input data [1]. Autonomous driving systems constitute of multiple tasks where classical supervised learning methods are no more applicable due its complexity and absence of labelled data. Use of RL or Deep RL applied to autonomous driving or the self driving domain is an emergent field [17].

The two types of deep reinforcement learning algorithm widely used these days is Policy Optimization and Deep Q Learning. Deep Q learning algorithms try to learn the optimal Q function Q(s,a) with a function approximator Q_theta(s,a). It generally uses an objective function based on Bellman equation. To the contrary, Policy optimization algorithms try to directly learn the policy pi_theta(a—s) by optimizing parameters theta by using gradient ascent on performance objective J(pi_theta).

The primary advantage of policy optimization approaches is that they are principled, in that we explicitly optimize for what we desire. This makes them more stable and dependable. Q-learning approaches, on the other hand, only in-

directly improve agent performance by training Q theta to meet a self-consistency equation. Since there are so many failure scenarios in this type of learning, it is less stable. When they do work, however, Q-learning approaches have the benefit of being far more sample efficient than policy optimization techniques since they can reuse data more efficiently [26].

There are a number of algorithms that exist on this spectrum and are capable of carefully balancing the strengths and limitations of each side. Soft actor critic (SAC) [20] is a variation that stabilizes learning by using stochastic policies, entropy regularization, and a few more methods. After reaching the steady-state phase, the maximum entropy framework in SAC may diminish the efficiency of learning outcomes [19]. Highsight experience replay (HER) is a sample-efficient replay approach for off-policy DRL algorithms that allows the agent to learn from both failures and successes, comparable to humans. HER gives the agent a supplemental reward based on the idea of aim, which increases the efficiency of the learning outcomes even if the objective is not met [19].

Behaviour Cloning (BC) is from the Imitation learning family which uses observations from the expert to learn [24]. In paper [18] trains a Recurrent Convolutional Network for autonomous driving in lane changing context by using BC to learn from human experts driving vehicles. The paper noted the ease of changing architecture when using BC technique and found that spatio-temporal properties worked well in dynamic environments. The huge advantage of BC algorithm is its end-to-end trainable architecture which uses supervised learning to learn policies from an expert by inferring expert's actions [5][7][14][18]. However the BC approach suffers from several limitations such as over-fitting and generalisation issue [5]. BC algorithm is a simple yet effective approach in the project and it can be used to learn the policies created from other algorithms selected in the project.

Evolutionary algorithms are algorithms with ability to evolve in order to learn a task. EA's are based on the concept of survival of the fittest principal of the modern genetic [27]. Genetic algorithm is a population based metaheuristic algorithm belonging to the family of evolutionary algorithms [27]. Genetic algorithm is inspired by Darwin's theory of evolution. Genetic algorithm is used to solve complex real world problems, one of which is autonomous cars modelling. In the paper [1] implements the genetic algorithm with the combination of artificial neural network which is also referred as Neuro-evolution to model vehicle dynamics and train autonomous cars. The paper states that evolutionary algorithms evolve via trial and error method over the iterations and hence provide advantages over the data dependent legacy approaches. It is also observed that the use of neural network with genetic algorithm can reduces the initial phase of generalising features and works without any prerequisite data for model training [1]. One of the notable advantages of genetic algorithm is parallelism. The genomes in the population acts as independent agents, the algorithm is able to explore the search space in all the direction at a time [11]. Although, both genetic algorithm and neural networks largely depend on the choice of parameters

such as population size, cross-over and mutation rate, number of network layers. Hence wrong choices can lead to a situation where algorithm may not converge.

## 3 Problem Definition and Algorithm

Problem definition and algorithm

### 3.1 Soft-Actor-Critic Algorithm

Deep Reinforcement Learning's goal is to learn a new environment in a short amount of time and then generalize to other situations. In non-simulation contexts, the conditions may go unnoticed throughout training. Sample efficiency is a problem for some of the most effective RL algorithms in recent years, such as trust region policy optimization (TRPO), proximal policy optimization (PPO), and asynchronous actor-critic agents (A3C) [20].

This is because these strategies allow for on-policy learning and necessitate new samples following each policy modification. Using experience replay buffers, Q-learning based off-policy algorithms like DDPG learn efficiently from past samples. However, such approaches are hyper-parameter sensitive and require a great deal of adjustment to converge. SAC is an alternate strategy for accelerating convergence.

HER is a sample-efficient replay method that improves off-policy DRL algorithm performance by allowing the agent to learn from both successes and failures. HER is applied to SAC and propose to help SAC learn more effectively.

### 3.2 Behaviour Cloning Algorithm

Behaviour Cloning (BC) generates trajectory by learning policy from direct mapping of states to actions without recovering reward function of the expert [15].

$$\tau^d = \pi(s) \tag{1}$$

$$\upsilon_t = \pi(x_t) \tag{2}$$

Equation (1) is the general aim of BC algorithm to generate trajectory of agent by learning policy for a given state. The goal of BC for the project car parking in an action-state space can be formulated as equation (2) to generate control input $\upsilon_t$ using policy $\pi$ given the current state $x_t$. The driving task for BC can be treated as a supervised learning problem [14][15][18].

The abstract of BC algorithm described in [15] requires a dataset of trajectories from the expert $\mathcal{D}$, policy representation $\pi_\theta$ and objective function $\mathcal{L}$. The result is an optimised policy parameters $\theta$ trained using the dataset $\mathcal{D}$ by optimising the objective function $\mathcal{L}$.

$$\mathcal{D} = \{(\upsilon_i^*, s_i^*)\}_{i=1}^M = d^{\pi^*} \tag{3}$$

Equation 3 describes the dataset represented by a set of control input ($\upsilon_i$) and state ($s_i$) pair from the trajectories of the expert. The policy of the expert is used as the ground truth reward and assumed to be near the optimal policy $\pi^*$ [10].

$$\ell(x^L, x^{demo}) = (x^L - x^{demo})^T (x^L - x^{demo}) \tag{4}$$

The quadratic loss function also known as $\ell_2$-loss or least squares (LS) in equation (4) is commonly used as an objective function and most optimiser libraries include additional regularisation term. The $x$ terms are vectors and loss calculates the action from agent's policy $x^L$ against expert's action $x^{demo}$. Other loss functions can be used as the object function such as $\ell_1$-loss, cross entropy loss, hinge loss, etc.

**Algorithm 1** Behaviour Cloning algorithm

1: expert_trajectory $\mathcal{D} \leftarrow$ sample from expert's policy from environment
2: agent_policy $\pi_\theta \leftarrow$ initialise a random policy parameters $\theta$
3: train agent_policy $\theta$ dataset expert_trajectory $\mathcal{D}$
4: save optimised agent_policy $\theta$

The algorithm (1) is a high-level pseudo-code of the behaviour cloning approach. Line *3* uses supervised learning algorithms with hyper-parameters such as objective function $\mathcal{L}$, optimiser, network architecture, etc.

### 3.3 Evolutionary Algorithm

EAs are a family of algorithms which are inspired by biological process of natural selection and Darwin's evolution theory. Neuro-evolutionary algorithms are the combination of neural network and genetic algorithm [1]. Each genome in this algorithm is a neural network. Weights and biases of the neural network are initialized randomly. Each takes the observation from environment as an input and return required actions. Based on these actions fitness score is calculated and returned by the network. The goal of neural network is to optimize the fitness score. Genetic algorithm works based on five stages: Population, Fitness evaluation, Selection, Crossover, Mutation [16][9].

**Population**: Population is set of solution candidates to the problem we are dealing with. These individual candidates are also referred as genomes or chromosomes. Here each genome is a neural network whose weights and biases are randomly initialized.

**Fitness evaluation**: Fitness function calculates a fitness score for each individual or genome which is the measure of difference between desired and current state. Each neural network returns a fitness score which is used in selection to generate future individuals.

**Selection**: Selection is the process of selecting neural networks based on their fitness score as parents for future generations. Selection process can be done in

various ways. One of the most common approaches is ranked selection where individuals are order in descending order of their fitness score and those with higher fitness score are selected as parents.

**Crossover**: Crossover is the process where weights and biases of two neural networks are exchanged with each other. This process is also referred to biological mating where some genes from two individuals are pooled together to obtain new generation. Percentage of genes to be exchanged depends on crossover rate. New individuals which are referred as offspring are created until the crossover point is reached. These new individuals are further added to the population.

**Mutation**: Mutation is the process where some of the weights and biases of the neural network are flipped based on low probability. Mutation ensures that the population has diversity in population and further increases the search space for the algorithm.

**Algorithm 2** Genetic Algorithm pseudocode [28]

```
Random population initialization
Fitness evaluation of the population individuals
Repeat
    Select neural networks with high fitness score as parents
    Generate new individuals by crossover process
    Perform mutation on the newly created individuals
    Fitness evaluation of the newly created individuals
    Replace old worst individuals with new best individuals
Stop until termination criteria
```

## 4 Experimental Results

Experimental result

## 5 Conclusions

Conclusion

## References

1. Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.
2. E Baburaj, BR Tapas Bapu, M Tamilselvi, and Bhasker Dappuri. Smart autonomous car parking for the modern vehicles. In *Journal of Physics: Conference Series*, volume 1964, page 042070. IOP Publishing, 2021.

3. Teck Kai Chan and Cheng Siong Chin. Review of autonomous intelligent vehicles for urban driving and parking. *Electronics*, 10(9):1021, 2021.
4. Ch Q Choi. How self-driving cars might transform city parking. *IEEE Spectrum. Febrero*, 20, 2019.
5. Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019.
6. Alejandro Correa, Guillem Boquet, Antoni Morell, and Jose Lopez Vicario. Autonomous car parking system through a cooperative vehicular positioning network. *Sensors*, 17(4):848, 2017.
7. Wael Farag and Zakaria Saleh. Behavior cloning for autonomous driving using convolutional neural networks. In *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–7. IEEE, 2018.
8. Bosch Global. Autonomous parking in parking garages, Mar 2022.
9. Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.
10. Sham Kakade and Wensun. Lecture notes in cs6789 foundations of reinforcement learning, December 2020.
11. B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.
12. Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.
13. Letian Lin and J Jim Zhu. Path planning for autonomous car parking. In *Dynamic Systems and Control Conference*, volume 51913, page V003T32A017. American Society of Mechanical Engineers, 2018.
14. Abdoulaye O Ly and Moulay Akhloufi. Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, 6(2):195–209, 2020.
15. Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.
16. Stuart J Russell and Peter Norvig. Artificial intelligence: a modern approach. malaysia, 2016.
17. G Sainath, S Vignesh, S Siddarth, and G Suganya. Application of neuroevolution in autonomous cars. In *International Virtual Conference on Industry 4.0*, pages 301–311. Springer, 2021.
18. Saumya Kumaar Saksena, B Navaneethkrishnan, Sinchana Hegde, Pragadeesh Raja, and Ravi M Vishwanath. Towards behavioural cloning for autonomous driving. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 560–567. IEEE, 2019.
19. Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.
20. Andrew N Sloss and Steven Gustafson. 2019 evolutionary algorithms review. *Genetic programming theory and practice XVII*, pages 307–344, 2020.
21. Monika Stoma, Agnieszka Dudziak, Jacek Caban, and Paweł Droździel. The future of autonomous vehicles in the opinion of automotive market users. *Energies*, 14(16):4777, 2021.

22. Rikuya Takehara and Tad Gonsalves. Autonomous car parking system using deep reinforcement learning. In *2021 2nd International Conference on Innovative and Creative Information Technology (ICITech)*, pages 85–89. IEEE, 2021.
23. Edgar Talavera, Alberto Díaz-Álvarez, José Eugenio Naranjo, and Cristina Olaverri-Monreal. Autonomous vehicles technological trends, 2021.
24. Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.
25. Wenbo Wang, Y Song, Juan Zhang, and H Deng. Automatic parking of vehicles: A review of literatures. *International Journal of Automotive Technology*, 15(6):967–978, 2014.
26. Xin-She Yang. *Nature-inspired optimization algorithms*. Academic Press, 2020.
27. Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.
28. Meriem Zouita, Sadok Bouamama, and Kamel Barkaoui. Improving genetic algorithm using arc consistency technic. *Procedia Computer Science*, 159:1387–1396, 2019.