Reinforcement Learning Applied to Autonomous Vehicles

5A Polytech Nancy project report

GUYOT Antonin

2022-2023

Contents

A - [Acknowledgements 3](#_Toc127912631)

B - [Introduction 4](#_Toc127912632)

[I - Context 4](#_Toc127912633)

[II - The project 4](#_Toc127912634)

C - [Reinforcement Learning 5](#_Toc127912635)

[I - Introduction 5](#_Toc127912636)

# Acknowledgements

I would thank Polytech Nancy for giving me the opportunity to make this project that made me learn a lot of very interesting concepts and for providing the really advanced tools I got to use during this project.

A special thank goes to Mayank Shekhar JHA, that supervised me during all this project and was there to answer to my questions.

I also thank Floriane COLLIN and Hugues GARNIER, who gave me this opportunity and assisted to my presentation, showing a lot of interest and curiosity for these aspects.

# Introduction

## Context

Reinforcement Learning (RL) approaches have recently emerged as the most efficient control solutions for complex systems. Indeed, these model-free methods provide a way of learning control laws based on real-life data. These approaches are also robust to dynamic changes and to uncertainty.

The goal of this project is to apply RL to autonomous driving vehicles. These systems are indeed pretty simple to control, and can provide a lot of information on their environment, thanks to numerous sensors.

## The project

The purpose of this project is to apply RL algorithm to a miniaturized version of a self-driving car.

The algorithms are provided by the Reinforcement Learning Toolbox[[1]](#footnote-1) of MATLAB/Simulink. The miniaturized car is the QCAR[[2]](#footnote-2), made by Quanser and provided with the QUARC Real-Time Control Software[[3]](#footnote-3), used to interface the QCAR in MATLAB/Simulink.

The goal of this project is to implement RL algorithms provided by MATLAB for the control of the QCAR, to achieve simple goals (obstacle avoidance, path planning).

# Reinforcement Learning Introduction

## Concept

### RL agent purpose

Reinforcement learning is an approach to machine learning in which an agent learns to make decisions by interacting with an environment and receiving rewards or penalties for its actions. The agent's goal is to maximize by trial and error the overall reward it receives, by learning to take actions that lead to beneficial states.

The environment is everything which is not included by the controller, such as the plant, the reference signal, or the measurements. The goal of the agent is to learn the best policy with respect to this environment to maximize its reward. The policy is represented by the following function:

Equation 1: Policy representation

With the policy function, the current state perceived by the agent (determined by the observations of the agent), and the action to take.

### RL compared to traditional control

In general, a control system aims to identify the appropriate actions to control a system and produce the desired behaviour. Feedback control systems utilize measured output to enhance performance and correct any unexpected disruptions or errors. Engineers use this feedback and a model of the system and its surroundings to develop a controller that meets the system's requirements.

Figure 1 shows the difference between traditional controls and RL. We can see that these two controls work almost in the same way, but an RL agent tends to constantly improve its policy by considering the feedback of the environment and update its policy accordingly.

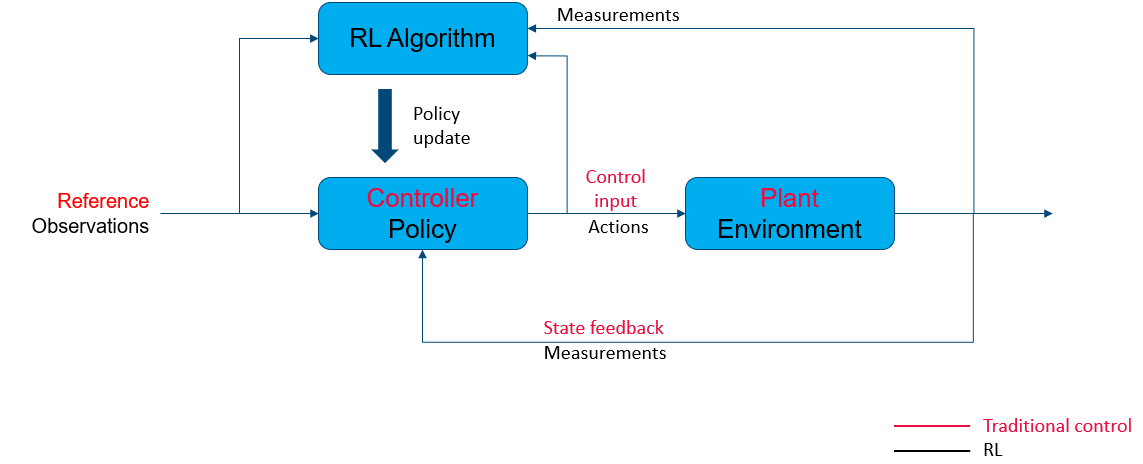


Figure 1 : RL vs Traditional control

## DDPG algorithm

### Motivation

RL can work with discrete or continuous states and time. The goal of the project is to make an agent work in an HIL configuration, and sensors used such as LIDAR return continuous values. These requirements have led to the choice of Deep Deterministic Policy Gradient (DDPG) algorithm, which is a RL algorithm inspired by Deep Learning and working with two neural networks.

### Architecture

DDPG works with two neural networks in the form of an actor/critic architecture.

The actor network applies the policy learnt by the agent, depending to the observations. The critic network tunes the policy by looking at the associated reward and trying to maximise it. The critic generates a Q-value, determined with the reward of the iteration and the Q-value of the next iteration (Equation 2). This value serves to make sure that the action taken at a moment will have a positive aspect at long term.

Equation 2 : Q-value calculation

is a discount value. It determines how much we take in consideration the long-term benefits of an action.

Une image contenant texte, signe

Description générée automatiquement

Figure 2 : Actor-critic architecture

### Training

A training episode is composed of a list of replay buffers created for each iteration. A replay buffer is the tuple displayed in equation 3.

Equation 3 : Replay buffer

With :

* the observation made at the beginning of the iteration
* the action taken accordingly to
* the reward associated to the action
* the observation made after the action is taken

For each replay buffer, the networks are trained one after the other.

Actor is trained by maximising Q with the network shown in figure 3:

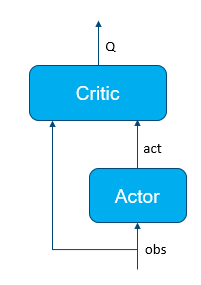


Figure 3 : Actor training

Then, critic is train by minimizing the cost function based on the double network shown in figure 4:

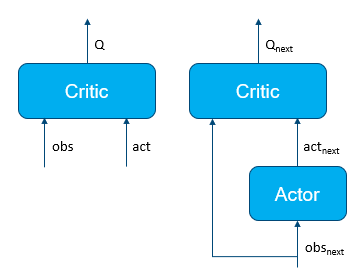


Figure 4 : Critic training

# simulation with MATLAB/Simulink

## Bicycle model

## Sensor modelling

## RL implementation

## Results

1. RL Toolbox documentation : https://fr.mathworks.com/help/reinforcement-learning/ [↑](#footnote-ref-1)
2. QCAR presentation : https://www.quanser.com/products/qcar/ [↑](#footnote-ref-2)
3. QUARC software : https://www.quanser.com/products/quarc-real-time-control-software/ [↑](#footnote-ref-3)