Reinforcement learning for aerial robotic

Antoine Mahé
CentraleSupelec
antoine-robin.mahe@centralesupelec.com

Introduction

Developping aerial robotic in the Greater Region[1] is the focus of the GRoNe project (FEDER INTERREG VA).

In machine learning, the optimal control problem is generally addressed throught reinforcement learning [6]. In robotics, policy search approaches are more especially considered, generally in conjunction with dynamic motor primitives, inspired by automation [5].

Recently deep learning approaches have shown promising results for direct control from visual information [2]. Studying the complementarity between theses different methods and how to combine them [8] is my first research perspective.

Experimental settings

In order to demonstrate control algorithm on real plateform, Bebop 2 drone from PARROT is used (shown in figure 1). The existing Robotic Operating System (ROS) driver for this plateform allow for an efficient integration of the plateform. Directly testing on real robots still is hasardous and can have dire consequence in cost. In order to minimize risks and improve the time efficiency simulation is of great help. Gazebo is a simulation environement well use in the robotic community and has good compatibility with ROS.

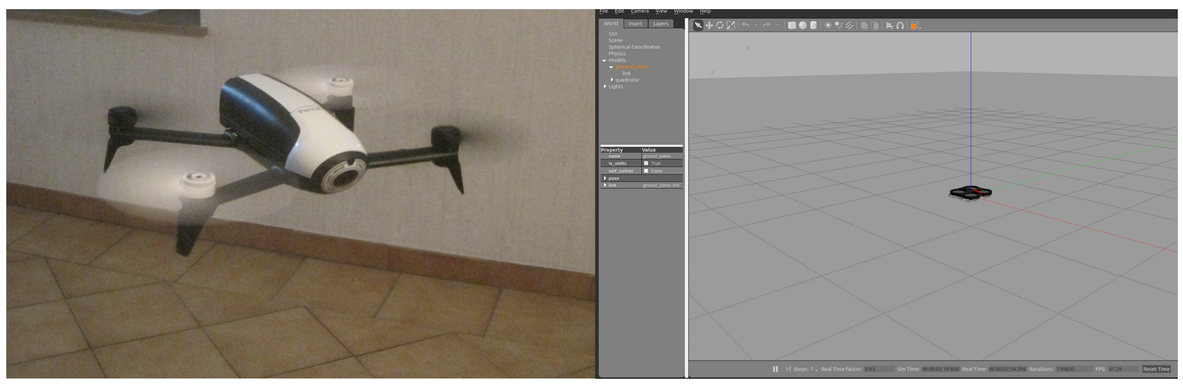

Figure 1: Parrot Drone and simulation environment

One of the research perpective being to study the potency of neural network to help solving the control problem, good framework for developing neural network architecture is important. The Tensor Flow implementation [1] is one of the most used framework. In order to improve the speed of the design high level python API focus on enabling fast experimentation such as Keras is very usefull.

Control algorithm

In order to tackle the control problem, a first approach consist into implementing a well known algorithm and then used Reinforcement Learning framework to improve from them. One of such standart algorithm is the Model Predictive Controle (MPC). In [7] whichs serve as basis for this reflexion used a special formulation of MPC called Model Predictive Path Integral (MPPI) which used reinforcement learning framework.

MPC

MPC programs use a model of the controled system to predict its behaviour over a certains period of time. Then it optimise the control strategy according to a cost function. For complex system modelisation may be a difficulty. But the cost function design allow some flexibility.

MPPI

In [7] the used of a more flexible MPC methode, MPPI a sampling-based algorithm. For example it will compute several trajectories for a mobile robots by generating set of commands and calculating the behaviour of the robot according to those. Then it tries to find an optimal trajectory, i.e. to find a set of optimal command, using a cost function.

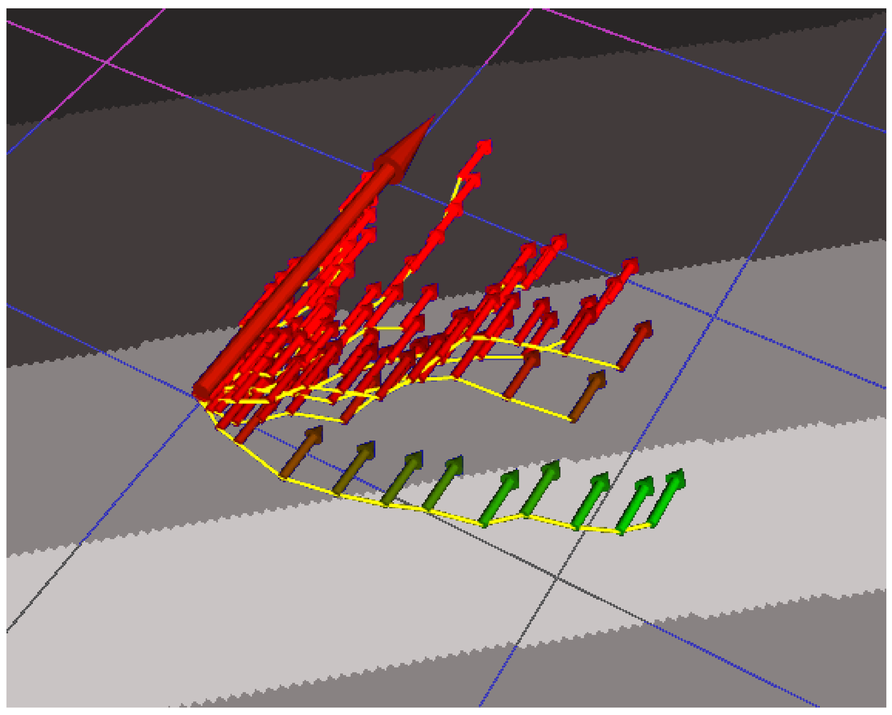

Figure 2: Sampling of trajectories for the droen

System modelisation with Neural Network

MPC and MPPI need a model of the system. One way to make such a model is to use a neural network to simulate the dynamic of the system.

[1]The Greater Region is an european regional cooperation https://en.wikipedia.org/wiki/Greater_Region

Simple Neural Network approach

A first approach to this modelisation is to use a simple architecture with dense layer that will be train using data that is generating by the system control by a human ooperator or a basic controller such as a Proportional Integral Derivative (PID) controller. The network learns to predict the next state of the system $X(t+dt)$ knowing the last state and the control applied to it $X(t+dt) = F(X(t), U(t))$.

Stochastic approach with Generative Adversial Network

An other possibility is to used a neural network as a stochastic model instead of a deterministic one. Using network architecture such as Generative Adversial Network (GAN) makes it possible. Such network have been used to do image generation [3]. A GAN is made of generator and a dicriminator. The discriminator is tasked with telling if a sample comes from a select data collection and discard those comming from the generator. While the generator do is best not to get discard. It is shown that those systems converge toward a state where the generator create sample that are impossible to differentiate for the data collection used to train the discriminator. Conditionnal GAN is specialised networks which generates his result not only from noise but also from input condition. Using such architecture could allow for model able to take stochastic behaviour in account (such as the wind in the case of flying drones).

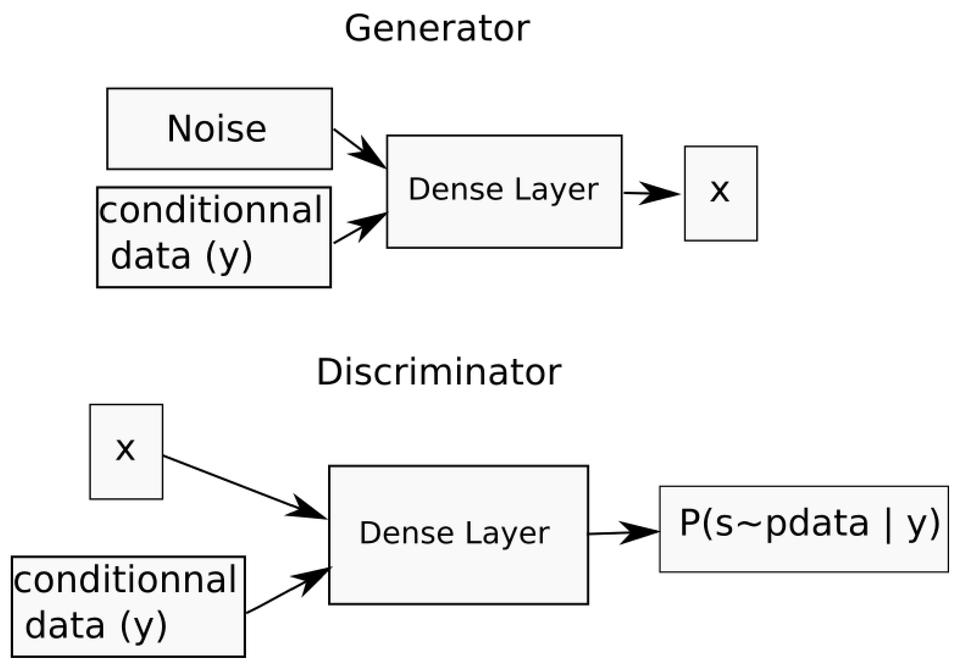

Figure 3: Example of Conditionnal Generative Adversial Network architecture

Forthcoming Research

The experimental validation of the use of GAN in a control setting is my current focus. The exposed solution here presents a way to first learn a model and then use it to control the system. To realy be in a reinforcement setting the knowledge should come from the agent interaction, thus learning to control a system in real time is another step ahead. Moreover it might be possible to use some of the network architecture in other way. For example it might be interesting to use conditional GAN in order to find better value following as it has been done with convolutionnal network in [4].

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep Spatial Autoencoders for Visuomotor Learning. *ArXiv e-prints*, September 2015.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[5] Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of tetris. *J. Mach. Learn. Res.*, 16(1):1629–1676, January 2015.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, 1998.

[7] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning.

[8] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search. *ArXiv e-prints*, September 2015.

Acknowledgements

This work is done under the Grande Région rObotique aérienNE (GRoNe) project, funded by a European Union Grant throught the FEDER INTERREG VA initiative.