ENSTA Bretagne
2, rue François Verny
29806 Brest cedex
France
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

report

CI2020
second year

July 26, 2019

2019 Summer Internship Dalhousie University, Halifax, Nova Scotia, Canada

Robust Deep Learning Tracking Robot

written by Arnaud Klipfel, Robotics specialization at ENSTA-Bretagne
supervised by Dr. Thomas Trappenberg, Dalhousie University

Contents

Ackowledgement

Resume

Introduction

Keywords

Part 1

Project Contextualization

1.1 Project Specification

1.1.1 Motivations

The main goal of this project was to build a *Deep Learning Tracking Robot*. Many tracking robots have been implemented in the previous years, few have used the adaptive, versatile, and now standard robotic Middleware *ROS*[1]. In addition, even fewer have ventured combining *ROS* and *Deep Learning* techniques.

Why *ROS* precisely? As defined by Anis Kouba in [Kou16], *ROS* is an « open-source *middleware* » that provides a robust and reliable framework to build robots. *ROS* is a voguish tool to create robots with advanced functionalities so much so that it has become the standard to develop robots. *ROS2* is already in part operational. In this project though, it has been decided that the first version of *ROS* will be used since *ROS2* is still in heavy development and *ROS* is better documented than its counterpart.

Why is *Deep Learning* suitable for robotic applications? In general, *AI* [2] is really adapted for critical algorithms, such as embedded codes on robots, since the performances are much higher than with traditional implementations. By traditional implementations, one could for instance consider iterative algorithms where for a particular task all processes are hand-implemented. The conventional solution that comes to mind regarding tracking is an *RGB* tracking, that is to say an algorithm that tracks the color in a frame[3] by applying color filters. *AI* enables much finer and more robust implementations since the *model*, which can be regarded as the applied processes, can evolve by itself through diverse methods such as training.

1.1.2 Requirements

In order to realize the tracking robot, several requirements had to be met in this project. First of all, the robot had to reuse an old *Race Car* platform comprising the chassis, the wheels . . . all the mechanics. The brain of the robot, that is to say the embedded computer had to be the *Jetson Nano* designed by *Nvidia* [MSV19]. By and large, to build the tracker the material of the laboratory had to be used as much as possible, and the equipment bought had to remain affordable.

[1]Which stands for *Robot Operating System*.
[2]Artificial Intelligence, which comprises *Deep Learning*.
[3]image.

1.2 Strategy

1.2.1 Different periods

Developing a robot is a critical task, and has to be conducted thoroughly. In this regard, *ROS* builds a framework for developing robots in a more organized way. For each task to implement, the work-flow has always been the following: research, simulation, isolated tests, integration.

Before even implementing something and integrating it, one should research all the possibilities that are offered to them. Then one of the easier or maybe most sustainable options could be chosen. Yet, even the tests, simulation, and integration has to be thought and foreseen beforehand.

Simulating robots, without depending on the hardware is a needed step. A specific hardware has always some specificities that could hide some issues and unpredictable behaviors. As a matter of fact, simulating algorithms that will be integrated in the robot afterward in a controlled environment prevents any hardware based issue to shadow our understanding of the basic implementation of the *software*.

Once the simulation is in place, it is then recommended to test the simulation as much as possible in order to unveil some detrimental singularities. In the case of the hardware, each component has to be tested independently before integration [4].

The final step is to integrate the work in the robot, and to test it to see if the robot behaves as expected or if there is any regression.

A robot is not just a piece of software, it is a system where the interaction of the software and the hardware is by definition the future behavior of the robot.

1.2.2 Different aspects

The development of a robot encapsulates different areas or type of work. In the construction of the tracker, principally two aspects have been tackled, the hardware, and the software.

In the first place, the hardware has to be well-chosen beforehand. The motor had to replaced, the *wifi* access had to be added, batteries had to be chosen . . . to fit the requirements of the tracking robot.

In the second place, the sofware has to be implemented. Precisely, the tracking algorithm had to be chosen, and the *ROS* architecture had to be designed.

For the entire project, the philosophy was to try to have a working first prototype as soon as possible. In a nutshell, having a hardware and software which are compatible, and then refining the existing platform. Indeed, *ROS* provides here another crucial boon, for it enables us to have an evolutive architecture where each part can be replaced easily without undermining the overhaul process. For instance, once an architecture is working, the tracking technique could be easily replaced.

Throughout this report, which presents the results of the project, three questions will be answered:

- How can the hardware of the robot be selected? You can find the answers to this question in the part 3 on page 10.
- How can the tracking algorithm be built? Which belongs more to the software. You can find the answers to this question in the section 4.1 on page 11.
- How can the *ROS* architecture be designed? Which belongs more to the software. You can find the answers to this question in the section 4.2 on page 11.

[4] Especially, some driver issues are sometimes really hard to pinpoint.

Part 2

State of the art

In order to build the tracker, the first step was to choose the hardware of the robot. Secondly, the *ROS* architecture had to be decided and thirdly the tracking algorithm had to be implemented. This part presents a brief overview of different existing projects and solutions that could have been selected throughout the project. All the projects presented and listed here will then be compared and the choices made will be underpinned in the parts 3 on page 10 and 4 on page 11.

2.1 Building an electric car

Several projects have aimed to realize an electric car robot, sometimes autonomous. Having in mind these projects can give some piece of advice on how it is common to design such systems.

Some projects have used the on-board computer *Jetson Nano*. One can name the following robots : *Jetbot* [nvia], *Kaya* [nvib]. These robots are both designed by *Nvidia*, the designers of the *Jetson* board series[1]. It is notably interesting to take as example their hardware selection since they use the same embedded computer the tracker use.

Other *RC-cars*[2] use other types of embedded computers. However, the hardware architecture with regard to *RC-cars* is almost always the same. With either the *DeepRacer* of *Amazon* [Ama], or the *Pi-Car raspberry* powered car [Sun], or *Ghost* car [Dan], or the *Roscar* [nai], or the *F1tenth* car [F1T], or the *Donkey* car [Diy], the architecture follows some fundamental principles.

The selection of the hardware is discussed further in the part 3 on page 10.

2.2 Tracking

The mission of the tracker is to follow a specific object in a frame [3], to be able to learn the object in that process, and even to be able to recover the target after any occlusion. Looking at the deep learning models and algorithms which have been developed in the recent years, some could be more labeled as detection algorithms, do not really learn the specificities of the target, and even track multiple targets. For instance, the network *Yolo* [Bje16] is rather designed for multi-tracking and not to follow a specific target.

Considering deep learning models that are able to track a specific target the database *GOT-10k* has produced an enumeration and ranking of the current most effective techniques [GOTB19].

For sure, in order to automate the robot, a detection algorithm could precede the tracking.

[1] Jetson Nano, Xavier, TX1, TX2 . . .

[2] *Remote Control Car.*

[3] Given by a camera for instance.

2.3 Using ROS

Developing with *ROS* gives also an incredible advantage to any roboticist. In fact, some *ROS packages* which are totally open-source already provide cutting-edge algorithms such as sensor fusion by Kalman filtering, *SLAM*. Unfortunately, applied to tracking few *ROS* packages were developed, and the existing ones do not use deep learning techniques at all.

Apart from that,

Part 3

Hardware

Part 4

Software

4.1 Tracking

4.2 ROS

Conclusion

Bibliography

[Ama] Amazon. *Amazon Deep Racer.* https://aws.amazon.com/deepracer/ (cit. on p. 8).

[Bje16] Marko Bjelonic. *YOLO ROS: Real-Time Object Detection for ROS.* https://github.com/leggedrobotics/darknet_ros. 2016–2018 (cit. on p. 8).

[Dan] Steven Daniluk. *Ghost II : controlling an RC car with a computer.* https://medium.com/hackernoon/ghost-ii-controlling-an-rc-car-with-a-computer-b1d1849d9e43 (cit. on p. 8).

[Diy] DiyRoboCars. *Donkey Car.* https://docs.donkeycar.com/ (cit. on p. 8).

[F1T] F1Tenth. *F1Tenth race car.* http://f1tenth.org/build.html (cit. on p. 8).

[GOTB19] GOT 10k : Generic Object Tracking Benchmark. *Leaderboard : Up-to-date generic object tracking performance.* http://got-10k.aitestunion.com/leaderboard. 2019 (cit. on p. 8).

[Kou16] Anis Koubaa. *Robot Operating System : the complete reference (volume 1 and 2).* Springer, 2016 (cit. on p. 6).

[MSV19] Janakiram MSV. *NVIDIA Brings Affordable GPU to the Edge with Jetson Nano.* https://thenewstack.io/nvidia-brings-affordable-gpu-to-the-edge-with-jetson-nano/. 2019 (cit. on p. 6).

[nai] Github repository by the user naisy. *ROSCAR.* https://github.com/naisy/roscar (cit. on p. 8).

[nvia] nvidia. *Jetbot : an educational robot.* https://github.com/NVIDIA-AI-IOT/jetbot/wiki (cit. on p. 8).

[nvib] nvidia. *Kaya.* https://docs.nvidia.com/isaac/isaac/apps/tutorials/doc/assemble_kaya.html (cit. on p. 8).

[Sun] SunFounder. *Raspberry Pi Smart Robot.* https://www.sunfounder.com/picar-s-kit.html (cit. on p. 8).