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 25, 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

Mots clés

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].

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. *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 will be the *Jetson Nano* designed by *Nvidia*. 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.

1.2 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

[1] Which stands for *Robot Operating System*.

[2] Artificial Intelligence, which comprises *Deep Learning*.

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 [3].

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.

[3] 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 the facing page and 4 on page 10.

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.

2.2 Using ROS

2.3 Tracking

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

Part 3

Hardware

Part 4

Software

Conclusion

Bibliography

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

[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).