Autonomous F1/10-car using Artificial Intelligence

Jens de Hoog*, Thomas Huybrechts*§, Peter Hellinckx*§
jens.dehoog@student.uantwerpen.be, thomas.huybrechts@uantwerpen.be, peter.hellinckx@uantwerpen.be

*Department of Applied Engineering, Electronics-ICT, University of Antwerp, Belgium
§IDLab, Department of Applied Engineering, University of Antwerp - iMinds, Belgium

***Abstract*—Insert abstract of the paper.**

I. Introduction

Despite the fact that artificial intelligence (AI) is not a new concept, it has become increasingly popular these days. Its first goal was to let machines replicate the intelligence of a human being, but that goal appeared to be more difficult than researches thought it would be. Slow progress was made for the next years, but technology has become more powerful and the interest in AI increased again, especially in marketable aspects [**?**]. At present, AI is quite ubiquitous and is implemented in many topics, such as self-driving cars.

Moet nog uitgebreid worden

II. Prerequisites

The racecar was built using the online tutorials provided by the organisation, f1tenth.org. Although these tutorials are made for using the car on a race circuit, certain prerequisites have to be made.

A. Slow Accurate Driving

First, the car cannot drive slowly with the current configuration, which is a challenge when debugging and testing software on the NVIDIA Jetson. Therefore, a solution has to be found.

1) Background: The Teensy microcontroller provides the signals for the steering servo and the Electronic Speed Control or ESC. These signals are independent square waves with a frequency of 100 Hz and a particular duty cycle, which ranges from 10% to 20% with 15% as neutral value. When applying the neutral signals on both ESC and steering servo, the car does not move. The duty cycle of 10% corresponds to steering to the left or driving backwards at full speed, whereas a value of 20% corresponds to steering right or driving forward at maximum speed.

The numeric values used in the code for generating the PWM-signals are represented by a 16-bit integer. Thus, a duty cycle of 10% accords to ± 6554 and 20% corresponds with ± 13108. One can find that the matching value of 15% is ± 9831.

2) Problem: A problem arises when the ESC-signal has a duty cycle which is a fraction higher than the neutral value. The motor does not drive instantly. Only at a particular numeric value of 10120 or at a duty cycle of 15.44189%, the car starts driving at walking speed. Though, this speed is too fast when developing and debugging software. Hence, a certain deadzone exists between duty cycles 15% and 15.44189% which cannot be filled in by the hardware components. A software approach is needed in order to drive more slowly.

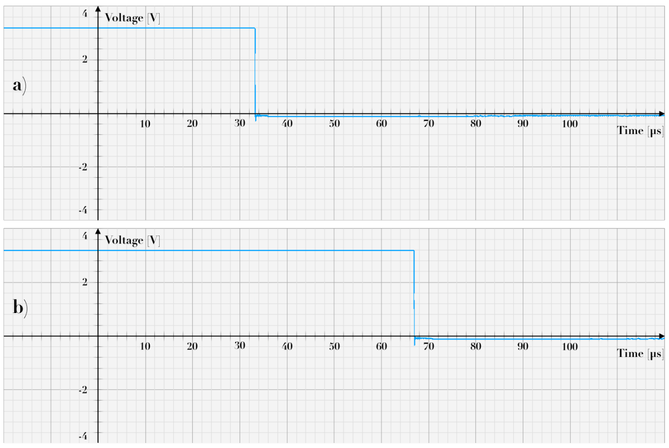

Fig. 1. a) Shows the motorsignal in its neutral state. The duty cycle is 15%. b) Shows the motorsignal at the moment when the car begins to drive. The corresponding duty cycle is 15.44189%. Note that these figures are zoomed in order to display the difference between the two signals. Therefore, the complete period of the signal is not shown.

Figures 1a and 1b show the existing deadzone between the neutral state (shown by 1a) and the state at which the car starts to drive (shown by 1b). The figures are zoomed to make sure this difference is visible. As a result, the

3) Solutions: The general approach is to modulate the ESC-signal. More specifically, by switching the ESC on and off at a particular frequency, the speed can be regulated. The switching signal can be compared with a gate which allows the motorsignal to pass through. To generate such a signal, built-in timer mechanisms are needed. Luckily, the Teensy development board has three timers in its processor which can be accessed with low-level registers. Paul Stoffregen built a library which carries out the timer functions. This library is a fork of the original library for Arduino and implements multiple optimisations, along with support for more hardware configurations.

The first concept dealt with a PWM-signal which would modulate the ESC-controlsignal. The motor controller was turned on and off using the generated square wave. By varying the duty cycle of that wave, the active time of the ESC varied with it.

A square wave signal was generated by using the aforementioned timers. When a square wave with a fixed duty cycle of 50% is generated, an Interrupt Service Routine (ISR) can be connected which will be executed every time the pulse is high. When a signal with a variable duty cycle is produced, no ISR can be attached. The solution for this problem was to bound the signal to a GPIO-pin on which an ISR is connected. Thus, every time the pin goes high due to the PWM-signal, the ISR is executed. Inside this ISR, the ESC is alternately turned on and off.

Although this solution should work in theory, it appeared to be more difficult in practice because of limitations of the hardware components. First, the speed controller does not accept the signal at a high switching frequency with a low duty cycle. Besides that, the Teensy board cannot generate signals at a low switching rate with a high duty cycle.

The second concept takes a different approach. Instead of generating a switching signal with a fixed frequency and a variable duty cycle, a signal is generated with a fixed duty cycle of 50% and a variable frequency. When the car needs to speed up, the frequency of that signal is decreased. Thus, the ESC has more time to increase the motor speed and the car will drive faster. By increasing the switching rate, the motor has less time to accelerate by which the car will slow down.

This approach is implemented by using the same timers of the previous approach. Instead of directly changing the timer frequency, the timer frequency is fixed at 100 Hz and generates interrupts at that rate. The ISR increments a counter which resets when a certain limit is reached. This limit value is set when the ESC-signal is located inside the deadzone. Depending on the specific location within that deadzone, the value of the limit is calculated. When the counter inside the ISR resets, the ESC is either turned on or off by sending the appropriate controller signal. In other words, by increasing or decreasing the limit value of the counter, the period by which the ESC is turned on or off is varying in the same way.

B. *Driving at constant speed*

Het is een handigheid om op een hoger abstractieniveau ”Ik wil 5km/u rijden” te kunnen zeggen. Dit komt vooral van pas bij de artificile intelligentie. Daarom moet er een systeem gebouwd worden dat het rijden tegen constante snelheid voorziet. Dat systeem wordt in dit deel uitgelegd. De lezer ontdekt welke technieken ik heb geprobeerd en welke uiteindelijk gekozen is.

C. *Calculations*

Hierin vertel ik welke berekeningen gemaakt moeten worden om proper te kunnen rijden. Mooie voorbeelden zijn het throttlen en sturen voor het rijden op de ideale racelijn.

III. Artificial Intelligence

A. *Introduction and definitions*

The subject ’Artificial Intelligence’ has many definitions, but Davis preferred this one: ”There are a number of cognitive tasks that people do easily—often, indeed, with no consciousthought at all—but that it is extremely hard to program on computers. Artificial intelligence, as I define it, is the study of getting computers to carry out these tasks.” [**?**] Copeland stated another definition: ”Artificial Intelligence is usually defined as the science of making computers do things that require intelligence when done by humans.” [**?**]

According to the research of Tkáč et al. [**?**], artificial neural networks are structures which were built to match the aggregation of knowledge in a human nerve system. These artificial structures can solve non-linear problems which are extremely hard to solve by a conventional computational structure. Because of the flexibility, sturdiness and efficiency, these artificial neural networks are gaining popularity in different sorts of applications such as pattern recognition systems and financial utilisations. [**?**], [**?**]

Kan nog uitgebreid worden

B. *Applied on F1/10-car*

In dit deel praat ik over hoe AI in mijn systeem is geïntegreerd en hoe het concept in z’n werking gaat. Hoe de sensoren, ROS en de AI samenwerken, wordt hier uitgelegd met behulp van blokschema’s en flowcharts.

C. *Implementation*

Dit onderdeel handelt over de exacte implementatie, zonder stukken code te laten zien. Hier vertel ik over hoe het gekozen framework (bijvoorbeeld TensorFlow) in elkaar gestoken is. Het verschil met de vorige paragraaf is dat daar de AI gezien wordt als een zwarte doos die verbonden wordt met andere onderdelen. In dit deel wordt die zwarte doos opengetrokken en ga ik kijken naar de binnenkant van de AI. Zoals ik reeds zei komen hierin geen stukken code, maar eerder de concepten binnen de AI.

Ook kunnen hier designkeuzes naar voren komen, zoals welk framework ik heb gekozen en waarom.

IV. Results

In dit deel komen de resultaten naar boven van mijn masterproef. Hierin vertel ik hoe het autonoom rijden zijn werk doet en wat ik ermee behaald heb. Zo kan ik vertellen over de parcoursen die de auto heeft afgelegd en hoe de wagen dit tot een einde heeft gebracht.

V. Further Research

Dit deel handelt over het onderzoek dat nog verricht moet worden om een volledig autonome auto te verkrijgen. Dit kan gaan van objectdetectie met de LiDAR-sensor tot interactie met andere wagens met oog op racen.

VI. Conclusion

De conclusie van dit thesis wordt hierin verwerkt. Wat er exact in zal komen, weet ik nog niet precies. Ten eerste kan er iets gezegd worden over de voorbereidingen die getroffen moesten worden vooraleer de AI zijn werk kon doen. Als tweede kan er verteld worden over de AI zelf en hoe deze opgelost is, en of dit een goeie oplossing bleek te zijn. Als laatste kan er iets in komen over het verdere onderzoek dat nog moet gebeuren.