KTH Mechatronics Advanced Course

MF2063, HT 2018

FINAL REPORT

ESS-NW/ESS-CAR

Jonas Ekman
Yini Gao
Jacob Kimblad
Leon Fernandez
Fredrik Hyyrynen
Yifan Ruan

December 8, 2018

Abstract

Abstract starts here, what should be included:
The problem issue subject being addressed
How the problem is tackled
Overview of the results, and indication as to what level they solve the problem.
Implications of the results

Contents

List of Figures

List of Tables

1 Introduction

This report presents the process and results of two projects "Embedded Service for Self-adaptive Network" (ESS-NW) and "Embedded Service for Self-adaptive Car" (ESS-CAR). This chapter will start by describing the background of the two projects. The next thing to be described is formulation, goals and motivation of the two projects. Following this will be a short discussion on the delimitations for our team. The last part of this chapter will present an explicit report disposition which helps readers to get a sense of the overall report.

1.1 Background

1.1.1 Background subsection blabla

1.2 Project Description

1.2.1 Project Description sub blabla

1.3 Delimitations

1.4 Report disposition

2 Literature Review and State of the Art

3 Methodology

3.1 Engineering approaches ?

3.2 Tool-chains ?

3.3 Project management

Scrum project management is used during the process of our projects.

4 Implementation

4.1 System overview

maybe put communication diagram here

4.2 Implementing SDN network

4.3 shared memory things

4.4 Communication between Beaglebone and Arduino ?

As the Beaglebone only supports to be the master in a SPI connection, so we have two beaglebones with SPI configurations and each of them has two Arduino slaves. One Beaglebone has two spi devices: SPIDEV0 and SPIDEV1, and the SPIDEV1 device is disabled initially but is used for HDMI interface.

In order to have one SPI device on Beaglebone connecting to multiple Arduino slaves, we use a third party C++ library called SimpleGPIO which enables us to use other general purpose pins as slave select of the SPI connection.

4.5 Sensors

Three categories of sensors are implemented in the prototype vehicle to monitor its surrounding environments. Data from distance sensors and speed sensor will be sent to an Arduino initially, then sent to corresponding Beaglebone. Data from Pi Camera will be sent to the Raspberry Pi which is directly connected to the main network.

4.5.1 Ultrasonic sensor

To get data from HC-SR04, a short 10µs pulse should be supplied to the trig pin of ultrasonic sensor, then the sensor will send out an 8 cycle burst of ultrasound at 40 kHz and raise echo. The echo signal we get is a distance object which is pulse width and the range of the signal is in proportion.

An Arduino Micro handles both the generating the trig pulse and interpreting the echo signal. The Arduino will set the output pin to low, wait for 5ms, set the pin to high, wait for 10ms, set the pin back to low. This is the process of generating the trig pulse. After the Arduino sends out the trig pulse, it waits for 2 ms, then reads the value from the pin connected to sensor's echo pin. The last step is convert the received value to distance in unit of centimeter.

4.5.2 Reflective object sensor

4.5.3 Pi Camera

In computer vision part, we tried to compare the performance of two different methods, which are object detection based on neural network and color detection using OpenCV.

The technique of transfer learning is used to apply MobileNet on RPI, which is an efficient model for mobile and embedded vision applications. The speed remains slow although hardware acceleration and multiprocessing have been carried out to improve.

By using OpenCV's DNN module, we are able to pass input images through the deep neural network and get a output image with bounding box of specific object and label.

Since we are working on source constrained device Raspberry Pi, we need to make the network simple and decrease the computational cost, so we combined MobileNet with Single Shot Detector.

In order to optimize the Raspberry efficiently and use sufficiently limited resource and memory on it when running neural network, we firstly apply hardware optimization which is to install optimized OpenCV complie. So, ARM NEON as an optimization architecture extension for ARM processors

is used. It is used for faster video stream processing, image analysis and speech recognition that is exactly what we are look for in our application. This architecture can execute multiple processing in the pipeline by a single instruction. Besides, VFPV3 as a floating point optimization is also used. After the first stage improvement, we get a 30 percent speed increase since we have make full use of the 4× ARM Cortex-A53,

When it comes to second stage of optimization, multiprocessing is used to increase the speed of processing video stream. The I/O tasks, dislike CPU bound operations, always take lots of time and delay the process. So moving the reading of frames to a separate thread from frame processing can obtain higher speed. Otherwise, every time I/O port access Pi camera, the main thread of script is blocked until the frame is captured and return to script. Multiprocessing can decrease the influence of I/O on CPU heavy application like video stream processing, especially in our real-time case. Now we can obtain a detection result within 1 second.

Additionally, there is a trade off between accuracy and output speed. In our application, we set a threshold for the output, in more details, only when the confidence score of detection result is high enough (above the specified threshold), the result can be output as a signal to steer the car. Otherwise, it will grab another frame and do the object detection for the other iteration. As a result, if the threshold confidence score of output increase, the output speed will decrease.

Since the object detection method outputs result slowly, we move on to the other method which is color detection using OpenCV to see whether the speed is fast enough to be applied on a car prototype.

We define the upper and lower limits for pixel values to classify three colors. Then specify which pixels fall into specified upper and lower range by masking. The speed of color detection is fast and also accurate enough for real-time application.

So we finally apply the color detection on our car, we use the result of detection to steer the car.

4.6 Controlling actuators

4.6.1 Steering servo

4.6.2 Motor ESC

4.7 Assemble the car, power supply, etc

5 Verification and Validation

6 Results

7 Discussion and Conclusion

8 Future Work