Summary

As traffic on highways and in cities becomes more and more chaotic, the number of vehicles increases, therefore a need for a system capable to solve these problems and at the same time to provide a safe and energy–efficient way to travel. This project proposes a prototype construction level of an autonomous vehicle illustrating the principle of vehicle platooning. This principle makes it possible for vehicles to travel together closely yet safely, reducing the amount of space used by a number of vehicles traveling on a highway. Thus, more vehicles can use the same road without traffic congestions, while providing lower fuel consumption. As a comparison to the existing solutions, the autonomous vehicle will be controlled by video processing the number plate aspect ratio of the platoon leader. The project aims to combine control and video processing algorithms into a real-time system that can illustrate the principle stated above. The outcome is set to be a solution for future means of transport that besides the facts stated will bring comfort and relaxation for commuters.

The author's contributions to the project were as follows:

- *Development of a new algorithm for vehicle platooning that uses number plate localization of the leader vehicle (pages 28-45) and optimizing it for a real-time application (pages 46-50)*
- *Development of the control subsystem and the communication protocol, in collaboration with Denys Berkovskyy (pages 10-14)*
- *Design of the SMPS power supply with 6 individual controlled outputs and current monitoring (pages 16-21)*
- *Design of hall effect speed feedback sensor (pages 15-16)*
- *Design of steering feedback sensor (pages 14-15)*

Nevertheless the most important contribution was system integration and development of an efficient and reliable system.

Content

List of Figures

List of Tables

Chapter 1

Introduction

Though ten years ago talking about an autonomous vehicle was more of a subject for a scion fiction movie, nowadays we hear about them being trailed at different test facilities around the world. “The idea is simple: a computer drives the car for you based on input from the surrounding environment” [1]. Although it may sound scary tomorrow cars will be able to do this and the future of autonomous cars is in sight. The benefits of driverless cars are multiple and should be taken into account even by their opponents. A recent study shows that the typical American driver spends in average 100 hour per year driving, all this in an era where time should be managed more efficiently. But what about goods that can be sent to destination by only loading them into a presented driverless car where you do not have to worry about the driver speeding and damaging the fragile load. If this is not enough to convince you that there is a need for them imagine renting a car that can take you to your destination while you are enjoying the view.

Even if the concept of designing a robust and efficient system seems to be simple, it is more than a stone throw away. This is mainly because the system has to respond to every unusual situation in the best way possible. Therefore only by intensive testing these situations can be simulated and incorporated in the system’s design. Even if the driverless cars were up for sale starting today there are still legal issues that have to be overcome. As Michael Mandel describes them in [1], this seems to be the pebbles that block the stream of evolution in this field. It should be mentioned that the first state where autonomous vehicles are allowed to drive on public roads is Nevada, USA and that this was approved in 2011 due to intensive pressure from Google.

Although extensive research has been conducted in this field predominantly from 2000 onwards, there is still no autonomous vehicle that is able to guarantee your

safety. We should mention at this stage the most relevant projects conducted in the last couple of years.

The ARGO Project [2], which started at the University of Parma and the platform of which was based on a Lancia Thema was concluded in June 1998 with a journey around Italy and is one of the first successful tested projects. At the same time, the European Commission was sponsoring one of the biggest budget project EUREKA [3], a collaboration between universities and cars manufacturers. In 1995, as part of the EUREKA Project, an S-class Mercedes Benz drove from Munich, Bavaria to Copenhagen, Denmark a 1,000 mile trip with autonomous control for over than 100 miles. Although the processing platforms were based on personal computers and therefore not feasible for embedded integration and considering the fact that the vehicle needed a human supervisor, the ARGO and EUREKA projects are of reference in this filed.

We should also mention DARPA[4] Grand Challenge, an international competition for driverless cars held in the USA and sponsored by the Department of Defence which challenges universities and individuals to develop sustainable algorithms and hardware for different driverless situations.

Nevertheless, probably the most well-known autonomous project of our days is the one conducted by Google[5]. The project has developed under the supervision of Sebastian Thrun from a Stanford University project which took part in DARPA's 2005 competition. The platform is based either on Toyota Prius or Audi TT and so far Google confirms that the cars have driven for more than 1,000 miles without human interaction and more than 140,000 miles with occasional human interaction.

Although it seems that the best performances were reached in the last couple of years, all the project mentioned above owe their research to that conducted by Ernst Dickmanns, a Mercedes-Benz engineer [6] who developed in the 1980s, a vision-guided Mercedes-Benz van that was following a leader vehicle at a constant interspacing, in a platoon type of formation.

1.1 Vehicle Platooning and its Benefits

A research path for autonomous vehicles is vehicle platooning. In the following we will try to familiarize the reader with its benefits and the previous work conducted in this field.

While the number of commuters increases, so does the number of vehicles utilizing the existing infrastructure. Traffic density problems arise all over the world. One common concern is to increase road capacity by the efficient utilization of the existing infrastructure. Capacity [7] is calculated by the following formula:

$$C = \frac{V}{Xr+L} \tag{1.1}$$

where C is the capacity, measured in the number of vehicles crossing a fixed point every time unit, V is the vehicle speed of flow, X_r is the vehicle inter spacing and L is the vehicle length. For simplicity we will consider that each vehicle using the infrastructure travels at the same speed (V) has the same length (L) and maintains the same distance to the vehicle in front (X_r). Since we cannot work on the vehicle length (L), it is easily seen that the only way of increasing flow rate is either by increasing the speed (V) or by decreasing the vehicle inter-spacing. The first option cannot be taken into account since vehicle speeds especially on motorways have reached a level where they already raise concerns on road safety. Therefore a way of increasing the roads capacity is by decreasing the vehicle interspacing. Since the drivers reaction time ranges on the average from 0.2 seconds to 1.2 seconds, decreasing the spacing without the use of an automated control system can cause safety issues. Even if the reaction time should not be of concern, this would be merely impossible because of the individual driving style of each individual driving at a distance that he considers safe to the vehicle in front.

One potential method of increasing road capacity by using a constant minimum vehicle spacing is grouping vehicles into platoons. Vehicle platooning is a convoy of vehicles where a lead car drives a fleet of other vehicles. Each car measures speed, direction and distance and adjusts itself to follow the car in front, thus forming a road train driven by the locomotive (lead car) attached to multiple carriages (following cars). This is achieved by inter vehicle communication (IVC) and autonomous driving systems that control the vehicle path according to the information received from the leading vehicle as well as from the monitoring systems of the car itself.

The California Program on Advanced Technology for the Highway (PATH) [8] was the pioneer in automated vehicle control and vehicle platooning. The program started in 1986 as a cooperation between the Institute of Transportation Studies, University of California and California Department of Transportation. Its aim was to provide fully automated control of vehicles both on lateral and longitudinal axes as well as a mean of increasing road capacity.

In August 1997 the first fleet of platooned vehicles hit the road at the National Automated Highway Systems, San Diego USA [9]. The platoon was formed of 8 “Buick Le Sabres” with a fixed vehicle to vehicle distance of 6.5 meters (21 feet) and a speed of up to 65mph. The test showed that platoons with the characteristics described above and an inter platoon spacing of 60 meters (200 ft.) would increase the road capacity from 2,000 vehicles per hour to 5,700 vehicles per hour. Though the figures are impressive and provide an increased road capacity of more than 200%, the system is not

feasible, needing besides vehicle to vehicle communication a vehicle to infrastructure communication, mostly for the lateral control, as well as for non linear distortions. Therefore, the system cannot function on the existing infrastructure and thus has an important drawback.

In 2009 the European Commission founded Safe Road Trains for the Environment (SARTRE) [10], a project to investigate and trial technologies for the safe platooning of road vehicles. The project includes a leading vehicle, usually a truck driven by a professional driver and multiple vehicle followers. Compared to PATH the system elaborated in the SARTRE project is much more feasible because it does not rely on vehicle to infrastructure communication. The first successful trial took place in January 2011 on Volvo's test track in Sweden, when a car was slaved behind a truck.

1.1.1 Platooning Benefits

The environmental benefits of platooning have been investigated in the PATH program. An average benefit of around 20% is estimated for highway driving [9]. These benefits vary with the aerodynamic, spacing and number of vehicles. In general small spacing between vehicles means great energy consumption, due to the low air drag (follower vehicles) and to a much lower extent due to the reduced rear turbulence (leader vehicle). However, keeping a small spacing puts more pressure on the control systems, so a balance needs to be established.

Safety benefits should not be forgotten either. A Department of Transport report stated in [10] shows that 95% of road accidents are caused by human contribution. In an automated system with a professional driver leading, this will be minimized.

Moreover, this also brings extended comfort for the commuters, just imagine driving to work while checking your e-mail. However, traffic flow benefits are the main contributions of this concept.

1.2 Aims and Objectives

This project proposes a prototype construction level of an autonomous vehicle illustrating the principle of vehicle platooning. The aim was to develop a self-controlled

vehicle capable of navigating itself without human intervention. The autonomous vehicle had to follow a so-called platoon leader vehicle at a close yet safe distance. The platooning method proposed has to rely on video processing of the movements of the leading vehicle and not on vehicle to vehicle communication, thus providing a system which may be adapted on the existing infrastructure and with no need for special equipment fitted on the lead vehicle.

As discussed above, most of the past projects relied on full sized vehicles and on expensive equipment. Most of them are packed with different type of sensors so that they can assess as well a given situation. We should also take into account the hardware mass and power consumption. All this does not comply with the budget or with the time allocated for a student project. With the latest progress in technology, we can find reliable and low cost navigation and processing systems proper for system integration within the budget and time slot allocated. As a vehicle chassis, we can opt for a remote controlled electrical car which has easy to integrate control input signals. The serious constraints on the system description are related to the fact that the system has to fit the budget and the time allocated for integrating all the modules. Another fact that should be taken into account at this stage is that our system should be as power efficient as possible due to the limited capacity of on-board batteries.

An interactive process of defining the systems design criteria, by comparing all the possible solutions was used. The major constraints were, as discussed above, the temporal and financial resources allocated as well as ease of integration and development on the final system. The final version presented here has been defined by the author and approved by his project tutor, Dr Naim Dahnoun.

Key Design Criteria:

- The autonomous vehicle has to have the ability to follow the leading car at a close distance.
- The system has to be independent from vehicle to infrastructure communication and from intervehicle communication.
- Independence from Global Positioning System or any other type of relative positioning that is not feasible in environments such as buildings or cities.
- Ability to follow the leader vehicle up to a distance of 1.5 meters.
- No active radars or sonars should be used.

Systems to be Integrated:

- The car chassis and safety applications should be controlled by an independent system allowing further development on the platform.
- Image processing subsystem implementing the concept of vehicle platooning.

- Internal data exchange using a standardised communication protocol between the image processing subsystem and the chassis control one.

Other Design Criteria:

- Prototype mass has to be limited to a reasonable weight
- Maximum speed should not exceed 30 km/hour
- Power consumption constraints have to be imposed so as the system should have the ability to function for at least 30 minutes without the need of recharging or an external power supply

Final System Design

In order to solve the above stated constraints and to comply with the project aim and the design criteria, the following systems have been selected from the ones made available by previous projects and approved suppliers. The vehicle chassis will be based on Traxxas E-MAXX [11][12], a 1:10 scale remote controlled electrical vehicle produced by Traxxas, USA. The built-in radio controlled transceiver converts the radio input commands into a Pulse Width Modulated (PWM) signal that is further on supplied to the chassis motor controller, EVX-2, and to the chassis servomotors. This will be replaced by a control system built on a microcontroller platform that will provide the versatility and the system integration needed. The remote controlled car meets the demands imposed and offers the appropriate size, permissible load, high-capacity batteries and it is capable to accommodate the following equipment:

- Logitech Webcam Pro 9000 (2MP, up to 1600 X 1200) for video input
- Stellaris LM3S8962 Evaluation Board for vehicle control
- PandaBoard with Texas Instruments OMAP4430 for the image processing subsystem
- Netgear Gigabit Ethernet Switch
- Switch Modulated Power Supply (SMPS) with 6 independent outputs.

1.3 Thesis Outline

In the following sections of this thesis we will discuss firstly about systems integration (Chapter 2). Here we will emphasise the overall system description, the tasks performed by every subsystem as well as the integration between them. Next we will talk about a new and innovative method for vehicle platooning, one which uses neither intervehicle communication nor vehicle to infrastructure communication. We

will discuss the algorithm proposed for number plate detection at some level of detail, using explicit examples (Chapter 3). In Chapter 4 we will discuss the optimisations performed on our algorithm, starting with optimisations at algorithm level and continuing with parallel processing. The last chapter is reserved for project conclusions and further work.

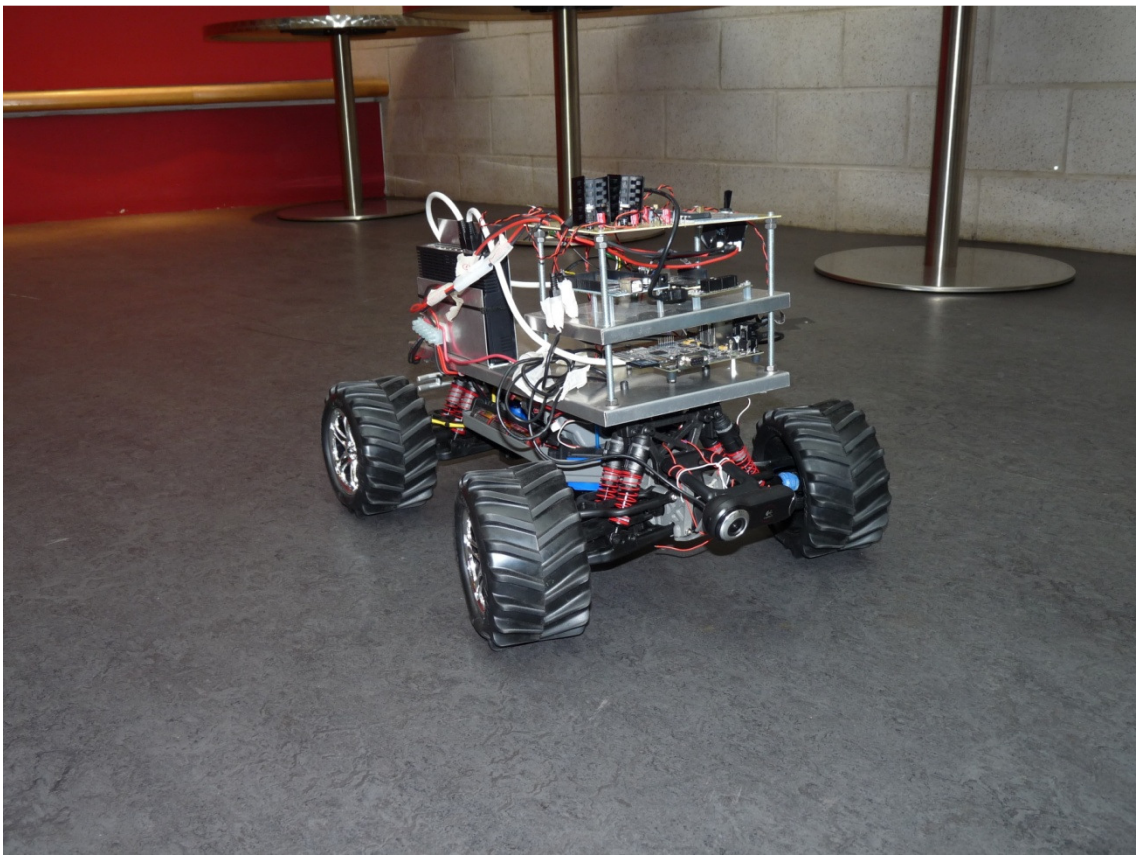


Figure 1.1: The Autonomous Vehicle

Chapter 2

System Integration

In this chapter we will discuss the overall design of our system. A block diagram of the components can be seen in figure 2.1. As stated in the aims and objectives of our project, we want to develop a system feasible for further development. Our project is meant to develop over time in a platform capable of testing and integration of future vehicle related paradigms.

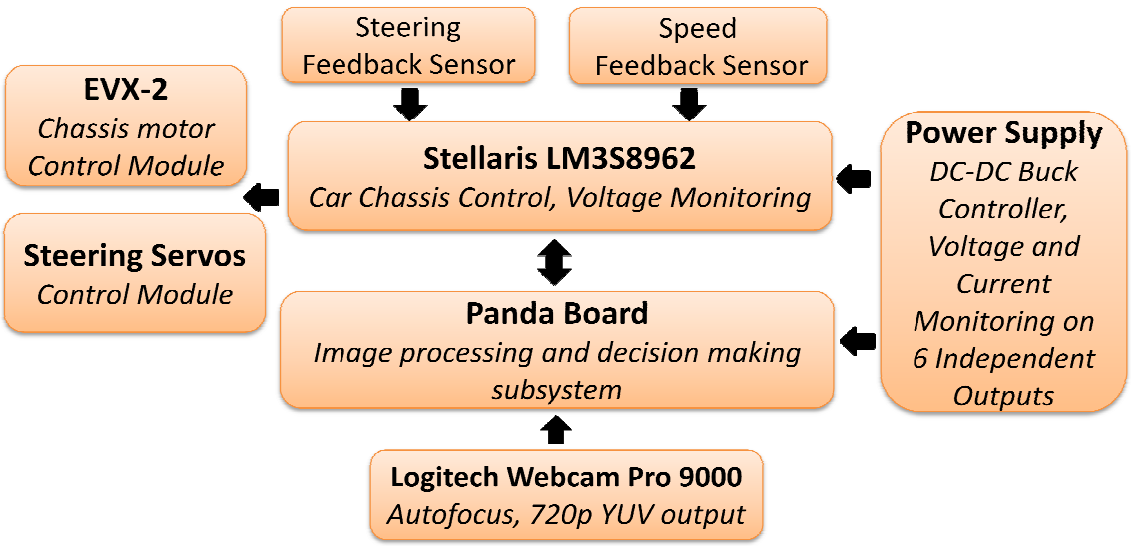


Figure 2.1: Top Level Diagram of the Prototype

The key component of our system is the vehicle chassis. This has to be a loyal abstraction of a real car chassis. Nevertheless the processing platforms used throughout the project have to provide both the processing power needed for complex control and image processing algorithms as well as a power efficiency suitable for a battery powered device. In the following, we will discuss shortly about the vehicle chassis, processing platforms as well as the sensors and power supply.

2.1 The Vehicle Chassis

The vehicle chassis has to have the ability to mirror the forces and the versatility of a real car. Our vehicle is based on the chassis of Traxxas E-MAXX [11][12]. The vehicle is a 1:10 scale electrical remote controlled car with a 4WD motor system capable of speeds of up to 30 + mph. It provides all the features that you would expect from a real car. From these we will enumerate the four points independent suspension system and the limited slip differential. All these, in turns, make the electrical vehicle a close resemblance of a real car, thus complying with our goals.

Figure 2.2: Vehicle Chassis – Traxxas E-MAXX

The chassis will be powered by two 8.4V, NiMH batteries with a capacity of 3,000 mAh. The 2.4GHz radio transceiver converts the radio frequency signals received from the remote control into input signals for the two servos and the motor controller EVX-2. It is worth mentioning that the controller and the two servos receive a PWM signal, but we will discuss this in more detail in the next subchapter. For now we should note that EVX-2 controls two 16.8 V motors, that trust the vehicle forward. Other key features of Traxxas E-MAXX can be looked at in the table 2.1.

Length:	518 mm
Front / Rear Track	417/419 mm
Ground Clearance	101 mm
Mass	4.19 kg
Height	248 mm
Tire Size	5.75” x 3.8”
Speed Control Type	EVX-2 Forward/Reverse/Brake
Motor	Electric
Drive System	Shaft-Driven 4WD
Radio System	2.4 GHz
Top Speed	45+ km/h
Batteries	Two 8.4 NiMH, 7-Cell

Table 2.1: Specification of Traxxas E-MAXX

2.2 Chassis Control

As we discussed in this chapter we want our system to be modular, so that other student projects can be developed on the platform supplied. The motor controller EVX–2 and the two chassis servos are controlled by PWM signals which have a frequency of 100Hz. The idle position for both the controller as well as the servos corresponds to a duty cycle of 15%. This further means that when PWM signals with a frequency of 100Hz and a duty cycle of 15% are applied, the car will be moving straight and the instant acceleration will be 0. The upper and lower limit correspond to control signals with a duty cycle of 10%, 20% respectively. Thus at the limits of our interval the vehicle will either steer left, right and fully accelerate or fully break. A waveform of the control signals can be seen in figure 2.3.

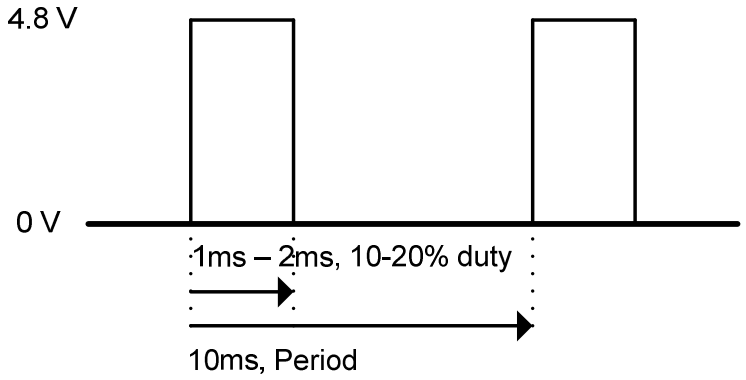


Figure 2.3: PWM Signal Waveform used for Vehicle Chassis Control

To further clarify what will the control system do and how it will interact with the vehicle chassis, please take a look at figure 2.4. We can see that the radio transceiver receives the commands from the remote control and converts them into control signals, with the properties described above, for the servos and the motor control. Therefore the task of our control platform is to receive speed and direction commands from the video processing unit and convert them into the appropriate PWM signal values.

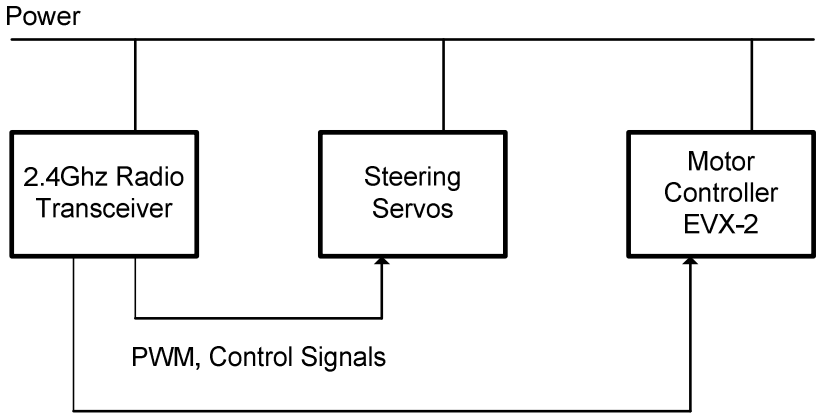


Figure 2.4: Original Control Logic, Traxxas E-MAXX

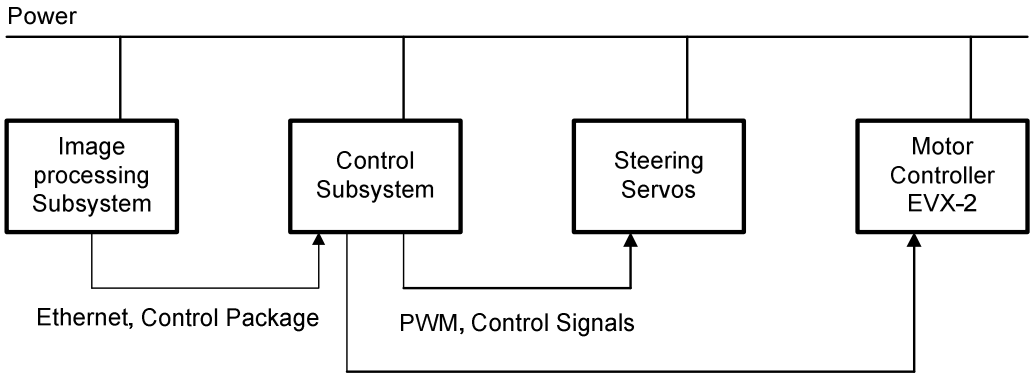


Figure 2.5: Autonomous Vehicle Control Logic

The video processing platform which will be discussed shortly is based on a development board running Linux as a multitasking operating system. Since we would like a modular system where each block has a presented task, an independent control subsystem which will communicate with the image processing unit is needed.

The control system is developed on the Stellaris LM3S8962 [13], an ARM Cortex-M3 microcontroller based architecture. In the following we will enumerate some of the microcontroller features.

Stellaris LM3S8962 Microcontroller:

- 32-bit RISC, ARM Cortex M3 @ 50Mhz
- 256 Kb Flash Memory
- 64 Kb SRAM Memory
- 2 PWM integrated generators
- four 10 bit ADC Channels
- Controller Area Network (CAN) module

Meanwhile the development board used is based on the above mentioned microcontroller benefits of the following features.

Stellaris LM3S8962 Development Board:

- Stellaris LM3S8962 microcontroller
- 10/100 embedded Ethernet controller
- OLED graphics display 128x96 pixels
- Magnetic speaker
- MicroSD card slot

As we can see from the specifications, the board is perfectly suited for our purpose, while leaving room for further development with features such as a CAN controller, a communication bus intensively used in the modern vehicles.

The algorithm developed converts the UDP package received over the Ethernet connection from the image processing platform to PWM control signals. The UDP package consists of 13 bytes of data and has the structure described in table 2.2.

The current implementation only uses 4 bytes of the UDP package. The package size has been chosen so that further work can be done and other values can be sent in the transmitted package alongside with speed and direction. Currently 2 of the bytes are used for speed control while other 2 for steering. Thus the PWM duty cycle will be changed in accordance with the values received as shown in the table 2.3

Name	*Size, bytes*	*Description*
Control flags	4	32 control flags
Reset	1	If set the control subsystem is restarted
PWM control	4	Values for 2 PWM output control, controls the duty cycle for speed and direction
Not used	4	-

Table 2.2: Structure of the UDP Package received by the control subsystem

Speed/Direction (2 bytes)	*PWM Duty Cycle*
0 …	10% …
32768	15%
… 65535	… 20%

Table 2.3: Correspondence between values received and PWM Signal

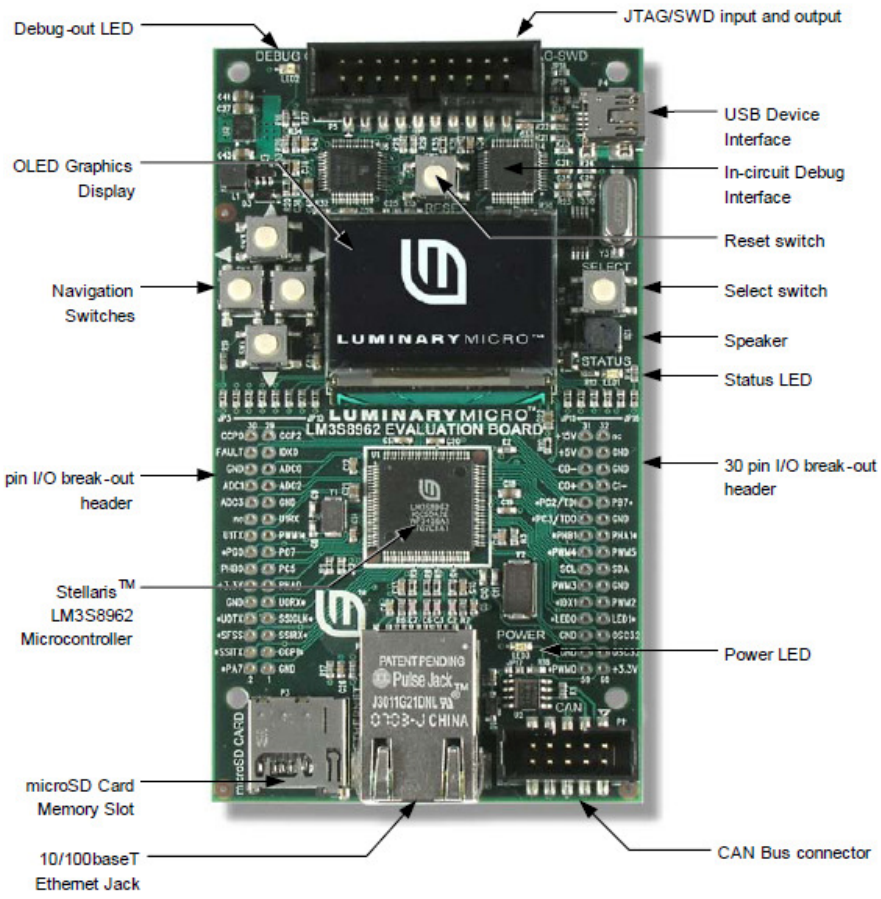


Figure 2.6: Stellaris LM3S8962 Development Board Layout

The board also interacts with the steering and speed sensors developed. Although this information is not used as a feedback in our system it allows room for further development and enhancement of the autonomous vehicle features. Therefore the control algorithm and the communication protocol have been developed in collaboration with Denys Berkovskyy.

2.3 Steering Sensor

In the initial testing process when we established the specifications of our system, we noticed that the steering system needed a form of feedback. This is because the torque associated with the steering servos was not enough to beat the friction force, when the vehicle is standing still. This is true if the vehicle speed is 0 mph. and we try to steer fully left or right the wheels will not always get to the limit position, depending on the road surface friction force. Nevertheless, this minor issue would not affect our platooning algorithm, since in our case the vehicle will be travelling at a speed higher than 0 mph.

A simple and ingenious steering feedback sensor was developed. To understand how our sensor works, let us take a moment and inspect how a servo motor works. Take a look at figure 2.7 [14].

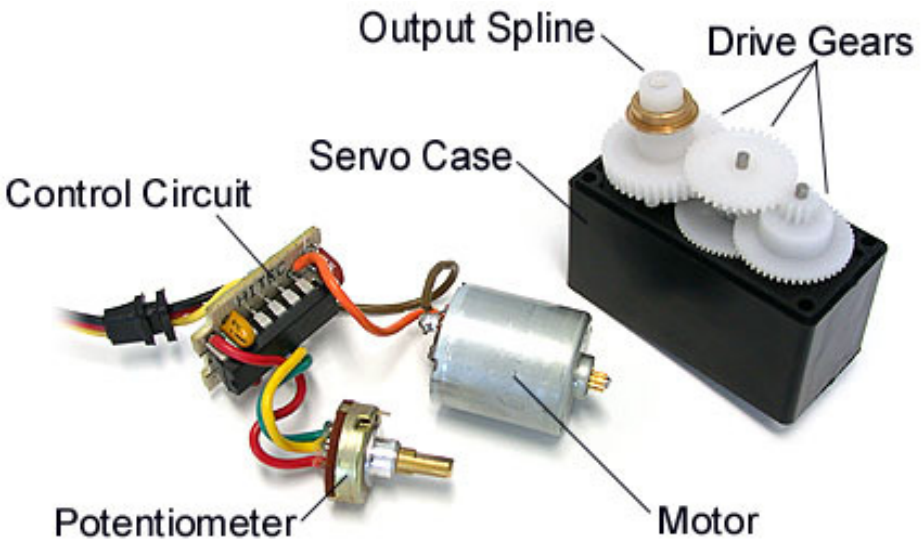


Figure 2.7: Servo Motor [14]

So, a servo motor works by receiving a PWM signal, in our case of 100Hz frequency and of varying duty cycle. The duty cycle determines the rotation of the output spline. To know the exact position in which the spline is at a certain moment of time, the servo control circuit uses a potentiometer which is directly connected to the rotation axis. This is exactly what we are looking for, however the rotation angle is not transmitted externally, being used as a form of internal feedback.

Since all the circuitry on our platform has the same common ground we will use the potentiometer reading as a feedback for our vehicle steering. The output voltage collected from the variable resistor varies between 1.1V and 1.7V with a value of 1.4V for a straight heading (90° servo rotation). This is further connected to one of the ADC ports of our control circuit.

2.4 Speed Sensor

The same can be said about the vehicle speed. For a robust system we need not to rely solely on the value of the duty cycle of the PWM control signal sent to the EVX-2 controller. We need a form of feedback sensoring as in the case of steering.

Our feedback circuit is based on a hall sensor mounted between one of the rear wheels and the transition shaft. A hall sensor is a transducer that changes its output voltage in relation to the magnetic field. This is valid when a strong magnetic field *(B)* of corresponding polarity is applied and the voltage will rise sharply while the voltage will drop sharply when the magnetic field *(B)* is decreased.

After numerous tests with different sensitivity as well as different operational types of hall sensors, we decided to use a unipolar hall sensor of medium sensitivity. Thus, when a South magnetic pole greater than B_{op} (the threshold value of the magnetic field applied B) faces Melexis US5781 [15], the sensor output will turn low, while when the value of the magnetic field applied decreases below the value of $B_{op,}$ the sensors output will turn high. A schematic of the speed sensoring system is shown in the figure 2.8.

The sensor output pin is connected to one of the input pins of Stellaris LM3S8962. Every switching, that corresponds to a wheel rotation, will generate an interrupt, thus measuring the time from one interrupt to another, while knowing the length covered by a complete wheel rotation will be equivalent to the instantaneous speed according to:

$$V = \frac{ds}{dt} \tag{2.1}$$

The initial speed monitoring system used one hall sensor and one permanent magnet mounted on the wheel with its south pole facing the sensor. Due to the fact that in one wheel rotation the car covers almost 50 cm, this seems not good enough for an accurate reading. Therefore the final version has one hall sensor and five permanent magnets, thus a speed reading is made every 10 cm or so.

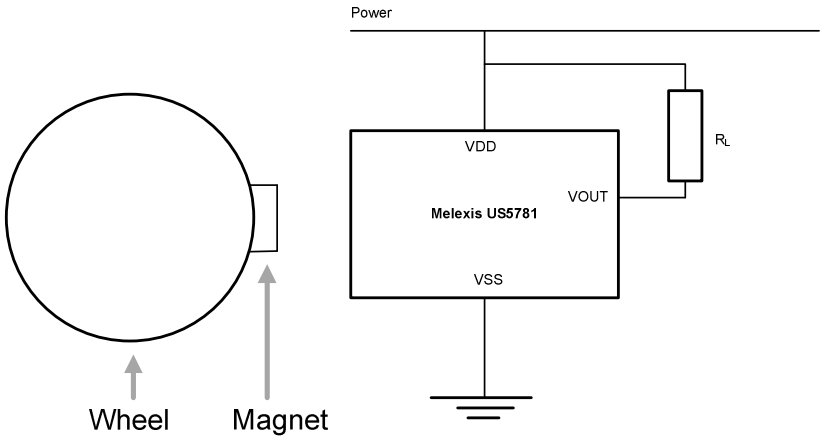


Figure 2.8: Speed Sensor

2.5 Power Supply

Since our system is battery powered we have designed a system as power efficient as possible. The vehicle chassis is powered by two 8.4V, 3,000 mAh, NiMH batteries. The same type of batteries is used to power the processing platforms. Therefore we will have two batteries for the processing platforms and sensoring system and two batteries for the chassis.

Since the input voltage is at a level of 8.4V and the boards need a 5V input we have to design a power supply that can convert between these two supply rails.

The easiest way to do this is by the use of a linear voltage regulator. To understand the drawbacks of this type of voltage converter, please take a look at the figure below.

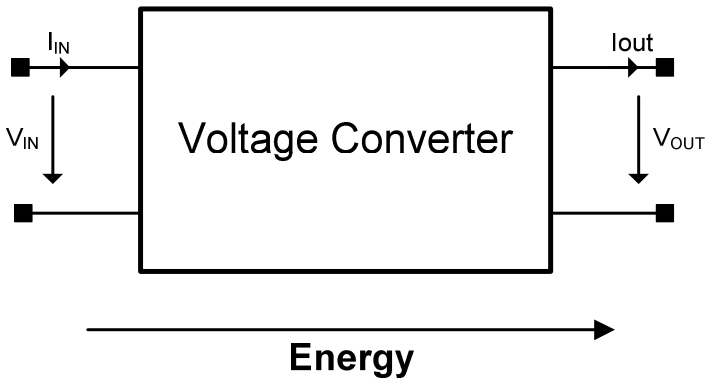


Figure 2.9: Liner Voltage Converter

We can note that the current I at the input port is the same with the current *I* at the output port. Thus, if our boards consume a total current of $I = 20A$, the power consumption at the output node would be:

$$P = U \cdot I = 100\ W \tag{2.2}$$

At the same time the power consumption at the input node would be:

$$P = U \cdot I = 168\ W \tag{2 .3}$$

Therefore between the input and the output node we would have a power loss of 68W. Nevertheless the linear voltage regulator would need a heat sink capable of dissipating the power loss. Thus besides the power efficiency issue, our power supply would have a considerable mass as well.

We need a power supply more efficient than this. The most efficient way is to use a switch modulated power supply (SMPS). If we treat it as a black box it is an energy transfer device that converts between input and output voltages by the use of a magnetic field.

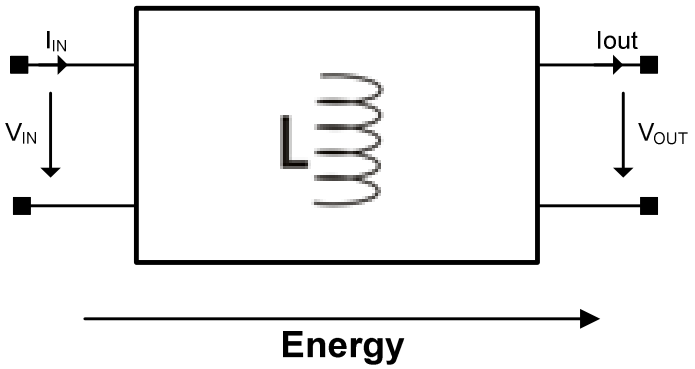


Figure 2.10: SMPS Voltage Converter

There are several constructive topologies, in the following we will introduce the reader to the Buck topology used in our power supply.

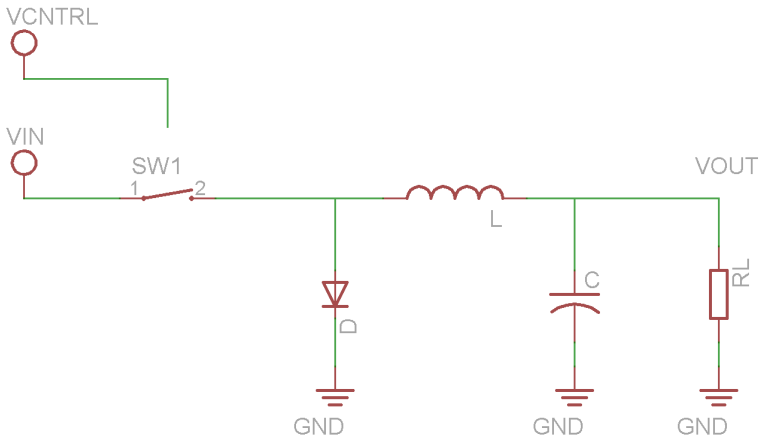


Figure 2.11: Buck Topology SMPS

In practice, the switch SW1 is replaced by a VMOS commutation transistor, while VCNTRL is replaced by a controller overseeing the process. Let us take a look at the way in which the power conversion is performed. Let us consider that at $t = 0$, the switch SW1 is closed. The equivalent circuit is drawn below.

From this equivalent circuit we can extract V_{IN} as

$$V_{IN} = V_L + V_{OUT} \tag{2.4}$$

Therefore the voltage drop on the inductance L is

$$V_L = V_{IN} - V_{OUT} \tag{2.5}$$

However we know that when applying a voltage to an inductor, the current will rise linearly at a rate of:

$$\frac{\Delta I}{T_{ON}} = \frac{V_{IN} - V_{OUT}}{L} \tag{2.6}$$

where T_{ON} is the time for which the switch is kept closed.

When the controller opens the switch the circuit will become as in figure 2.12. We can notice that the voltage drop on the inductor has changed its direction, due to the energy accumulated. Further on we now see the purpose of the diode D1 which assures a decrease path for our current. The circuit equation becomes:

$$\frac{\Delta I}{T_{OFF}} = \frac{V_{OUT} - V_D}{L} \tag{2.7}$$

where T_{OFF} is the time for which the switch is kept open.

In practice Schottky diodes are used with small conduction voltages compared to the output voltage V_{OUT}, therefore the equation 2.7 becomes

$$\frac{\Delta I}{T_{OFF}} = \frac{V_{OUT}}{L} \tag{2.8}$$

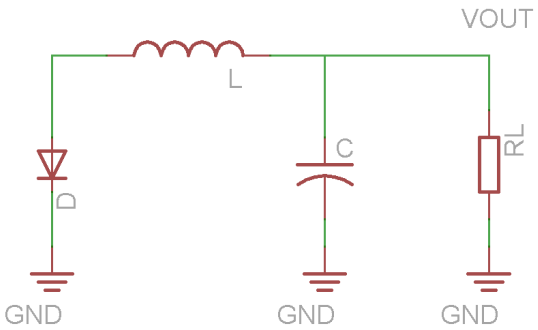


Figure 2.12: Buck Topology OFF Cycle

It has to be imposed that the value by which the current rises should be the same as the one by which the current drops. This is because if the inductor does not release the stored energy, from one cycle to another, in the form of a demagnetisation current, the current will rise, saturating the inductor (the inductor would act as an wire).

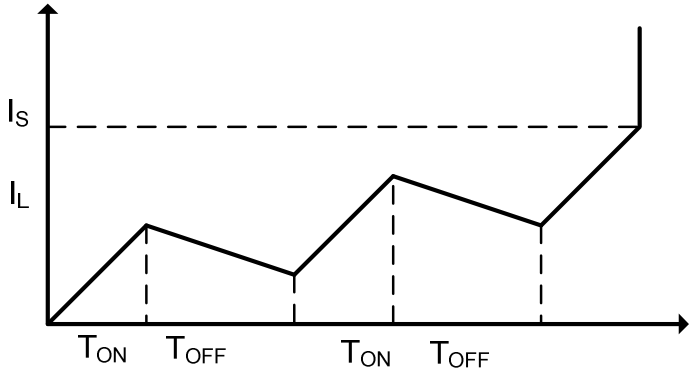


Figure 2.13: Inductor Saturation Current (I_S)

where I_S is the inductors saturation current. Therefore the current variation in the ON cycle has to be the same with that in the OFF cycle. Thus the transfer function is

$$\frac{V_{OUT}}{V_{IN}} = \frac{T_{ON}}{T_{OFF} + T_{ON}} = D \tag{2.9}$$

where *D* is the associated duty factor. Hence, we obtained an output voltage proportional with a control parameter *D*. We can also notice that we managed to overcome the power loss associated with the linear voltage regulator.

The topology used in our power supply is named Half Bridge, a Buck topology where the switch is replaced by two VMOS transistors. The controller (TI 40140) is capable of working with input voltages of up to 15V, while maintaining a fixed output voltage in the range 0.7 – 5.8V and a switching frequency of up to 1 MHz. However, although the high switching frequency translates to a smaller inductor we have to design our system to operate at a frequency that the transistors are able to cope with.

For the switching transistors we have chosen RJK 0305, due to their ability to withstand a maximum drain to source current of 30A and current peaks of up to 120A. The worst switching time, according to the datasheet, is that of the OFF transition and has a value of 35ns. Therefore we impose a switching period 50 times bigger.

$$T = 35ns * 50 = 1750\ ns \tag{2.10}$$

The inductor should be able to cope with the worst case scenario. Therefore from the inductance equation we determine the appropriate value as follows ($U_{IN\ MAX}$ *= 10V, T=2us,* dI_{MAX} *= 20A*)

$$L = U\left(\frac{dI}{dt}\right) = 0.66\ uH \tag{2.11}$$

However there still is head dissipation, thus power loss, associated with the switching transistors. For this we shall consider the two extreme scenarios:

Minimum input voltage of 6V.

$$\frac{V_{OUT}}{V_{IN}} = 0.83 \tag{2.12}$$

Maximum input voltage of 10V.

$$\frac{V_{OUT}}{V_{IN}} = 0.5 \tag{2.13}$$

Thus the ON transistor works more than 50% of the switching cycle. Even so the highest heat dissipation occurs while switching. We would like in an ideal scenario the transistor to work for ¼ of a period. Therefore we used one of the attributes of out controller, which can work with 2 sources connected in parallel, working sequentially. Thus when one is working the other is not. Therefore, our final design has 4 switching transistors and 2 inductors.

We have designed a power supply with an input voltage between 6 -10V and an output voltage of 5V with a maximum current of 20A. The power supply has 6 independent outputs. Each of them can be individually turned on or off. The maximum

current allowed on each output is limited by a 5A fuse. Also each output current is measured by the use of a linear hall sensor which converts the magnetic field (*B*) associated with the current to a voltage. This is done by the use of Allegro ACS712, a low resistance current detector able to withstand currents of up to 5A.

The power supply was designed to be controlled and monitored by the Stellaris LM3S8962. Therefore all the monitor output voltages have been rescaled to a maximum range of 3.3V with the use of a voltage divider and limiting Zener diodes. At the same time, the input that turns on of off any of the individual outputs is separated by optoisolators thus protecting the microcontroller from any unexpected voltage peaks. The two batteries are connected in parallel, providing the system with a 8.4V input and a 6,000 mAh capacity. Two Schottky diodes with a low conduction voltage have been mounted to prevent connecting the batteries in reverse polarity.

A complete schematic of the power supply can be found in APPENDIX I. Because the designed PCB was received later than expected the monitoring outputs are not integrated with the control platform.

We should conclude this subsection by mentioning that the autonomy of our vehicle has been improved from approx. 45 min (with the use of a linear voltage regulator power supply) to approx 2 hours.

2.6 Processing Platform

The basis of our system is constructed on the development board built around TI OMAP 4430 processor, dual core ARM Cortex A9 architecture, running at 1 GHz [16]. The board also benefits of a powerful graphics chip, supplied by Imagination Technologies and 1 GB of DDR2 RAM. As stated by the producers the board is meant to be used as a development platform for Smart phones or other mobile devices. Two USB ports and HDMI video and sound outputs are provided. Nevertheless, the most important interface provided, in our case, is the 10/100 Ethernet port. A built in boot loader loads, at start-up, the Linux Ubuntu 11.04 kernel from the 16GB SD Card.

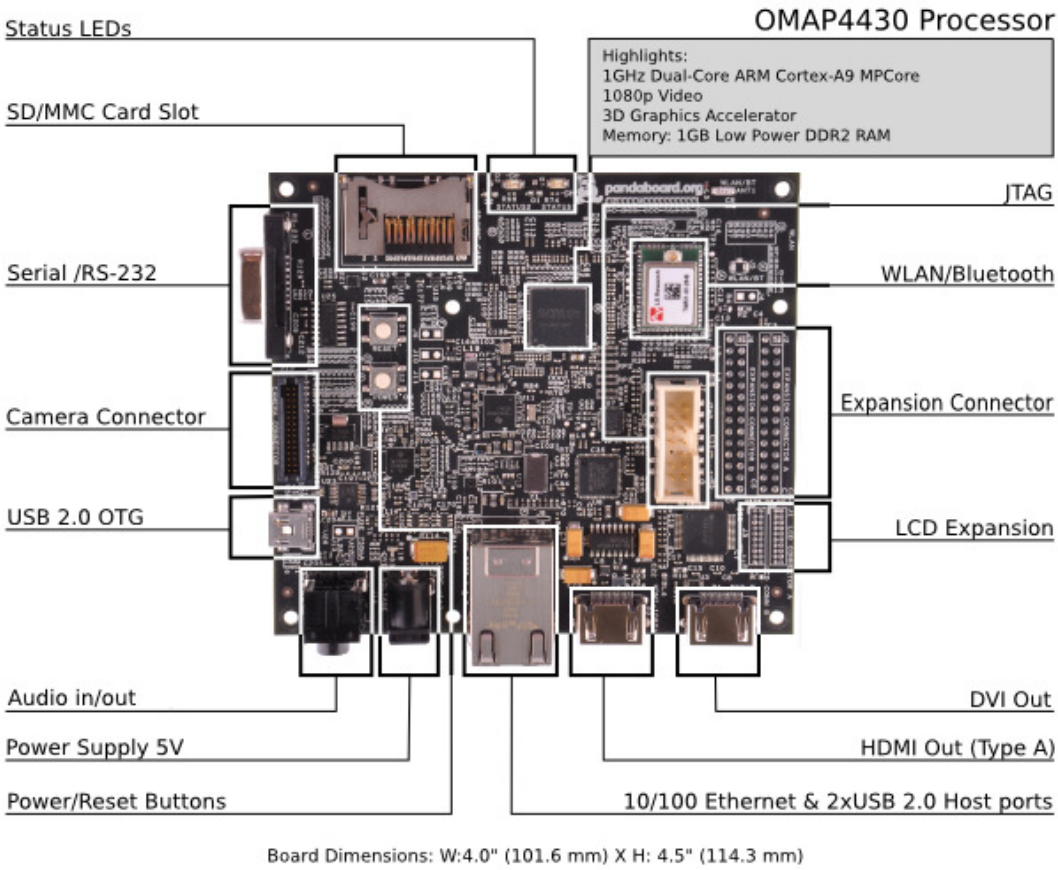


Figure 2.14: PandaBoard Layout

As specified above, the board is running Ubuntu 11.04 as a multitasking operating system. On this platform the image processing algorithm, discussed in detail in the next chapter has been developed. The board is interfaced with an USB webcam (Logitech 9000 Pro) which provides the video input for our algorithm. We can consider this platform as being our main decision centre, as an output of the processing, values for speed and direction are sent in an UDP format (discussed in subchapter 2.2) over Ethernet to the control platform which performs the appropriate conversions.

The processing power, size and power efficiency make the Panda Board the ideal candidate in regard to the specifications imposed. In the next chapter we will discuss in detail the algorithm developed on the above presented processing platform.

Chapter 3

Number Plate Detection for Vehicle Platooning

The main drawback of all the systems proposed so far is that they rely on intervehicle communication. Both PATH [8] and SARTE [10] systems rely on the information sent by the leading truck. The limitations of these systems are that with the extensive use of wireless communication they can become unreliable as well as the fact that the convoy can be led only by a vehicle that benefits of such equipment and communication protocol and can provide the information needed by the platooned vehicles.

The PATH project presents a method for lateral control that relies on magnetic sensors embedded in the road surface. In [31] the lateral movements are based on the tricycle model. Both approaches seem rather inappropriate, in the first case, that of the PATH system, there is a need for a changed infrastructure while in the second one presented in [31] the vehicle speed cannot exceed a certain limit.

A serious drawback for lateral control is the fact that at a certain moment a vehicle follower knows only the current position of the leading one, therefore it has to interpolate the trajectory in between. One way of overcoming this problem is the use of an absolute positioning system. As it is described in [32], a GPS system capable of accurate location, like RTK-GPS is both expensive and unreliable in urban applications where the satellite signals can be masked by tall buildings. Therefore most of the platooned vehicles rely on the robotic proposed methods.

The system proposed has to rely on video processing the movement of the leading vehicle in such a way that it can determinate both the direction as well as the distance to the leader. As presented above, the literature proposes a complex system that relies on complex stereovision systems for distance measurement and on road to vehicle

or vehicle to vehicle communication. Therefore the imposed design criteria for the image processing subsystem is to be able to provide direction and distance information at any given moment, in relation to the platoon leader.

As a common feature that can be used as a reference to all the vehicles within a certain country, that has a standardized dimension is the number plate, placed in the same position in terms of the car width. As imposed by the requirements for the image processing system, the platooned vehicles have to provide information about the distance and direction of the leader vehicle. Both of them can be provided by a real time number plate localization algorithm as follows:

Distance measurement – as the number plate has standardized dimensions, then by determining its width we can calculate the distance to the leading vehicle.

Direction – as the number plate is located in approximately the same position relative to the vehicle center by processing the location of the license plate relative to the center of the image we can determine direction.

3.1 Number Plate Detection

Automatic number plate recognition or ANPR systems have been around since 1976. The first number plate recognition system was developed by the Police Scientific Development Branch, in UK between 1976 and 1979 [17]. The first arrest was made in 1981 by detecting the number plate of a stolen car. Nowadays most of the countries rely on such systems for road rules enforcement.

Typical ANPR systems use a two stage algorithm composed of number plate localization and identification and optical character recognition (OCR). We can also classify the existing systems in two main categories:

- On site ANPR systems – which capture the image containing the number plate, process it and send the characters to a centralized database.
- Off site ANPR systems – which capture the image and without performing any additional computation send the raw image to a centralized processing plant.

Both types of systems described above have their pros and cons. On site ANPR usually preferred where the bandwidth of the communication line is limited. They perform OCR and send only a package containing the number plate characters and specific information like speed on weight of the vehicle. The drawback of this type of

system is that because of the limited processing power, a complete loop from capturing the image to sending the package can take up to 300ms.

Off site ANPR systems are preferred in places where a picture of the vehicle has to be saved with such information. They are dedicated cameras with a fast shutter speed. The image is captured and then the processing is performed by an off site processing plant. Thus the system is more feasible for highways where vehicle often exceed 100km/hour.

All ANPR systems success rate depends on the input image quality. Thus they function in accordance to garbage in, garbage out computing principle. The most important element of the system is the image capturing. For this dedicated cameras with fast shutter speed and a presented region of interest are used. Most of the countries use reflective number plates so that the number plate will be even easier to detect. Usually they are backed-up by an infrared illuminator. The perfect number plate detection scenario would be in the dark, using an infrared illuminator and capturing only the number plate region, since this will be the only reflective area of the vehicle. Obviously this is hard to accomplish, therefore the governments enforce the regulation related to number plates. Gaps are usually needed between lookalike characters such as P and D as well as a high contrast between the number plate background and the character itself. It is worth mentioning that ANPR systems are individually tailored to each country's number plate.

As part of the author's collaboration with TDC Systems, a UK based traffic monitoring company, we should state that ANPR systems are not used just in conjunction with speed detection systems. Probably the most extensive use of ANPR systems is in traffic classification, where each vehicle is classified by type, weight and registration number. Currently ANPR is used in car parks where the parked vehicle time is calculated as a difference between the entry arrival time and the exit one. Probably the most known use of ANPR systems in the UK is within the London congestion fine scheme.

3.2 Proposed ANPR Methods

Number plate detection methods are usually based on small regions of interest with a prior knowledge of the number plate location. In the following we will approach a short literature review of the current and proposed methods of detection. Since we are interested just in locating the number plate and not in character recognition we will focus on number plate location methods.

In [18] number plate localization is characterized as being the most important step in ANPR systems. This results because OCR is based on the region passed as a possible number plate. Therefore passing a false candidate results in a false detection of characters. The number plate location is found using texture information, similar to finding text in an image. Then a region of interest is characterized by a set of features extracted and the plate is tracked by a match of these features. Finding text in an image is based on spatial variance along the line. High variation regions are chosen as being possible candidates. The main drawback of this method is that the number plate region is a known prior, therefore not feasible for our number plate platooning method.

[19] is similar to the above method. It relies on the grayscale image spatial variance and on the fact that it will vary more over the number plate area than in other sections of the image. As a first step, an adaptive binary threshold is applied to the input image. The threshold level is found through an iterative process, starting from a predefined value and subtracting it as long as a number plate candidate is not found. Segmentation is done using some prior knowledge of the number plate size. Then an edge count is performed on the possible candidates. As a final step the number background is checked using the HSV image. HSV field is chosen since Hue and Saturation are enough to characterize the color information of the image. It is important to note that the author emphasizes the use of as much knowledge on the number plate as possible. Nevertheless since in our case the number plate size will not be fixed within a certain range, neither this method is feasible for our purpose.

Number plate regions are described by abundance of vertical edges. This is what is emphasized in [20]. As a first step dark characters on a light background are emphasized using the bothat operator. A vertical Sobel mask is used for detecting the edges, after removing the salt and pepper noise by the use of a median filter. Then the result is convoluted with a mask having an aspect ratio similar to the number plate. As a final step a number plate verification process is applied as follows:

- Plate regions are very small
- Plate regions are similar to a rectangle with a fixed width and height
- Plate regions average intensity has to be in a certain range
- Plate regions should be horizontal with a maximum angle of 35°

Again, due to the fact that the number plate region is assumed to be constant this method is not feasible to be integrated within our system.

The difference between neighboring pixels is the basis of [21]. For this a high quality image is needed. The line with the highest variations is selected as a candidate one. The algorithm searches then for the left and right borders. Binarization is used afterwards based on a threshold level computed from the histogram information. The upper and lower borders license plate are determined from the binary image. Although this method seems not to rely on any predefined window size, after testing this approach it seems inappropriate because of the high variation between neighboring pixels generated by the test road surface.

A vertical Sobel edge detector is used in [22] to cope with different illumination conditions. Next the image is convoluted with a licence plate sized window and the number of inside edge pixels is counted. The false candidates are eliminated by using two successive images and comparing the possible candidates in both of them.

Although other methods are proposed in the literature, like statistical distribution [23] or neuronal networks [24] we have presented above the methods that have influenced us in developing our license detection algorithm. We can notice that the common problems related to number plate detection are poor quality images, different illumination conditions, as well as abundance of grayscale variation in non number plate regions.

3.3 Algorithm Overview

As we have seen the proposed methods for number plate localization are not feasible for implementation in a video based platooning algorithm. This is mainly because they rely on fixed sized number plate as well as a known plate location within the input image. Though valuable information can be extracted from them, we will have to cope with most of the problems previous authors faced, besides the problems that we will face due to the fact that both of the vehicles will be moving. In the following section we will describe our proposed method and the problems faced.

Since we are interested in the actual size and location of the number plate we will focus on a contour finding method which can precisely determine the parameters needed. The method has to be effective in various illumination conditions as well as various locations of the number plate.

The algorithm has been developed with the use of OpenCV [25][26], an open source computer vision library. It runs under Linux, Windows, MacOS, and recently under Android.

OpenCV is designed in C and focuses on computational efficiency. Further optimizations can be achieved by using Intel's Integrated Performance Primitives (IPP), low-level optimized routines developed for Intel Core's architecture.

The focus of this open source project, established by Intel is to provide an easy to use computer vision infrastructure that helps programmers to build vision applications quickly and reliably. It is used in various domains such as security, robotics, medical imaging, etc and has soon become a reference in the computer vision world.

In the following section we will discuss to some level of detail the algorithm and its OpenCV implementation. A block diagram of the algorithm is presented in figure 3.1.

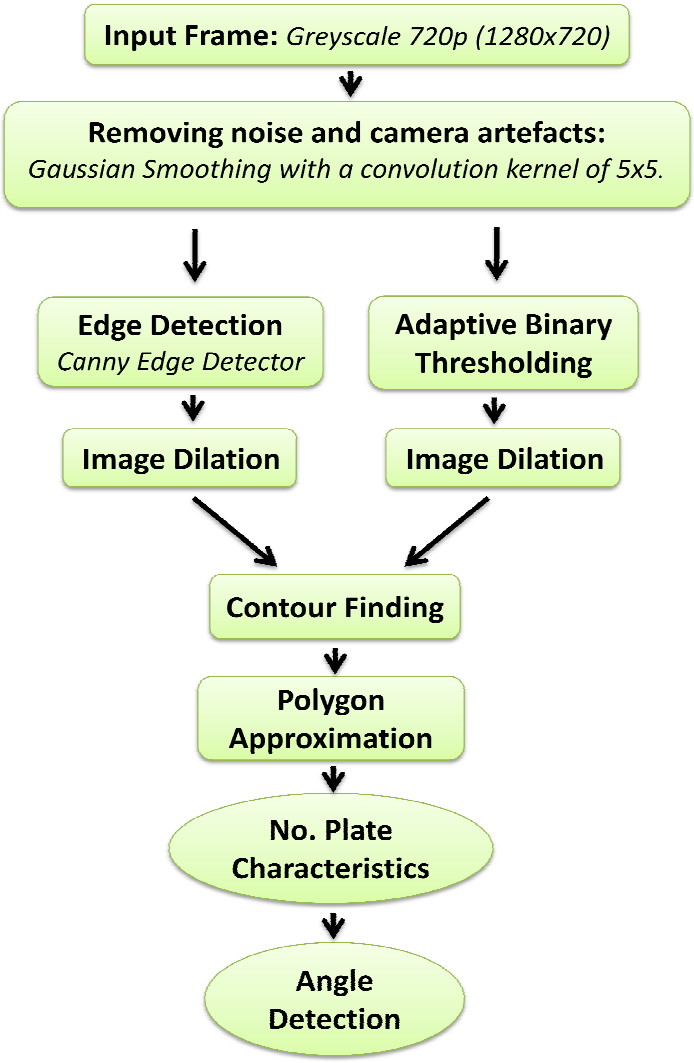


Figure 3.1: License Plate Detection - Algorithm Diagram

3.3.1 Frame Capture

The first thing we need to do is get the input frame from the camera. To do this we have to initialize a CvCapture structure with the appropriate values by the use of *cvCreateCameraCapture()*. Nevertheless we need some form of interaction at each loop cycle that can get the new frame and prepare it for further processing. This can be achieved by the use of *cvRetrieveFrame()* which will query the device initialized above and will retrieve a new frame.

As detailed in the block diagram, our algorithm expects the input image to be in a grayscale format. Therefore the currently captured frame will have to be transformed

from RGB to grayscale space. OpenCV facilitates color space changes by the implementation of *cvCVTcolor()*.

Figure 3.2: Sample Input Frame

Since we are only interested in the part of the frame where we can find a vehicle we will set a region of interest that covers the full width and ¾ of the height of the input frame. Based on our test environment boxes, barcodes, or books with the same aspect ratio as the number plate this will enhance the effectiveness of the algorithm as we are saving valuable processing time since the next functions will process less data.

3.3.2 Gaussian smoothing

As we have seen, preprocessing of the image is required so that we can eliminate noise and camera artifacts. Previous proposed methods use a median filter at this stage due to its ability to preserve edges better than other smoothing methods. Due to the fact that our leader car and number plate region have a high contrast difference, we are not required to do so and we can use Gaussian filtering. Also when the leader car has a considerable distance in relation to the platooned one, using a median filter can cause loosing number plate information such as small gaps between characters.

Gaussian blur or Gaussian smoothing [27] is used in computer vision algorithms to reduce noise or to reduce details as a preprocessing step. It is named after the famous scientist Carl Gauss. Gaussian filtering is performed by convoluting each point in the input array with a Gaussian kernel and then summing to find the result, also known as a two dimensional Weierstrass transform. It has the same effect as viewing the input image through a translucent screen.

Figure 3.3: Gaussian Filter Applied to Input Frame

For better understanding Gaussian smoothing and calculation of the associated kernel, let us take a look at the Gaussian function.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \tag{3.1}$$

And by multiplying the Gaussian distribution of the *x* scale and that of the *y* scale , the Gaussian distribution for a two dimensional array is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2-y^2}{2\sigma^2}} \tag{3.2}$$

where σ is the standard deviation of the Gaussian function and σ^2 is the variance and *x* and *y* are the distances from the original pixel on x and y scale, respectively.

The soothing functionality in OpenCV is provided by *cvSmooth()*. The function can perform soothing on an input image using several techniques such as Gaussian, median filtering or bilateral filtering. We will only focus on using the Gaussian smoothing although several other soothing techniques have been tested. The smoothtype

parameter has to be set to *CV_GAUSSIAN*. The transformation can be done in place, thus saving memory by replacing the source pixel with the destination one.

The kernel size which seems to produce the best result, both in terms of eliminating the noise as well as in their computation time is a kernel with a width and height of 5. As seen above, in order to calculate the coefficients of the Gaussian kernel, we still need the standard deviation. OpenCV accepts the value of σ as an input or, as in our case, it calculates the appropriate value at runtime according to:

$$\sigma_x = \left(\frac{n_x}{2} - 1\right) * 0.30 + 0.80, \quad n_x = param1 \tag{3.3}$$

$$\sigma_y = \left(\frac{n_y}{2} - 1\right) * 0.30 + 0.80, \quad n_y = param2 \tag{3.4}$$

where *param1* and *param2* are the width and height of the kernel. Replacing n_x and n_y above and calculating σ_x and σ_y we obtain $\sigma_x = \sigma_y = \sigma = 1.25$. A graphical interpretation of the above formula can be seen in figure 3.4.

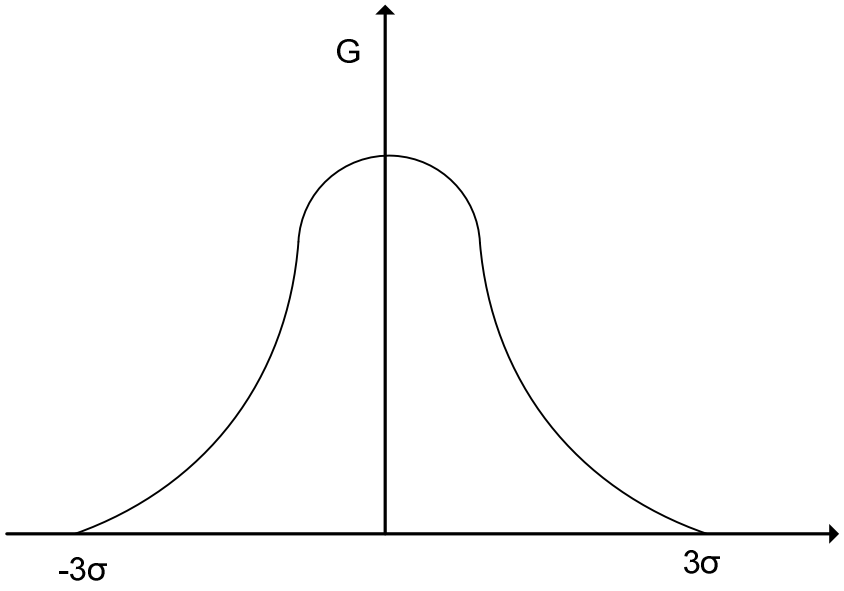


Figure 3.4: Bell Shape of Gaussian Distribution

We can notice that each pixel in the output image is a weighted average of the pixels in the input image where the original pixel value has the highest weight. Therefore we can conclude that the Gaussian filtering is an edge preserving smoothing technique. It reduces the high frequency component acting as a low pass filter. An interesting property of this smoothing technique that is worth mentioning is that applying successive filters with a kernel size of let us say 4 and 3 has the same effect as applying a single convolution kernel of size 5.

$$\sqrt{4^2 + 3^2} = 5 \tag{3.5}$$

This property becomes useful when using large convolution since it can save precious processing time. Nevertheless in our case the computation time was fairly larger when we applied a kernel of size 3 and then one of size 4. Usually in practice the Gaussian kernel is limited to the values of the kernel in the interval [-3σ, 3σ].

3.3.3 Canny Edge Detector

The purpose of edge detection is to preserve the structural information while reducing the amount of data and thus preparing the image for further processing. An edge is a series of points in an image where the brightness changes sharply. Generally an edge describes one of the following:

- Variations of illumination
- Discontinuities
- Changes of material

An ideal edge detector would detect any object boundaries on any given image.

Canny edge detection was developed by John F. Canny in 1986 [28]. Canny's goal was to develop a method of edge detection which is optimal in terms of the following criteria:

- Good detection – the algorithm has to be able to find as many edges as possible.
- Minimal response – image noise should not create false edges while an edge in the input image is detected only once.
- Good localization – a detected edge has to be as close as possible to its real position.

In order to better understand the edge detection method developed by Canny and its OpenCV implementation we should first consider the Sobel derivatives.

Sobel operator is the most widely used way to represent differentiation in computer vision. It operates for any order derivative and even for mixed partial ones. The Sobel derivatives are implemented in OpenCV by *cvSobel()*. It is worth noticing that they are approximations of derivatives in a discrete field, representing the derivative fit to a polynomial. One of the function parameters is the *aperture_size* and it represents the width and the height of the convolution kernel used for derivative approximation. For a simple understanding imagine the variation from a pixel value to the next pixel value and so on. By using a larger size kernel, a better approximation can be calculated, by using more reference pixels. Larger kernels are preferred for their

accuracy, while smaller kernels are preferred for the small computation time, but not for their sensibility to noise.

Now let us take a look at what happens in an edge region. Consider the first derivative of a function. This will be at a high value whenever the function changes rapidly. Therefore local maximum of first order derivative will be recorded in an edge like region. To detect this we will derivate the function once more. The 0 of the second order derivative correspond to points of local maximum, therefore edge regions. The only bad thing at this stage is that less meaningful edges will be detected.

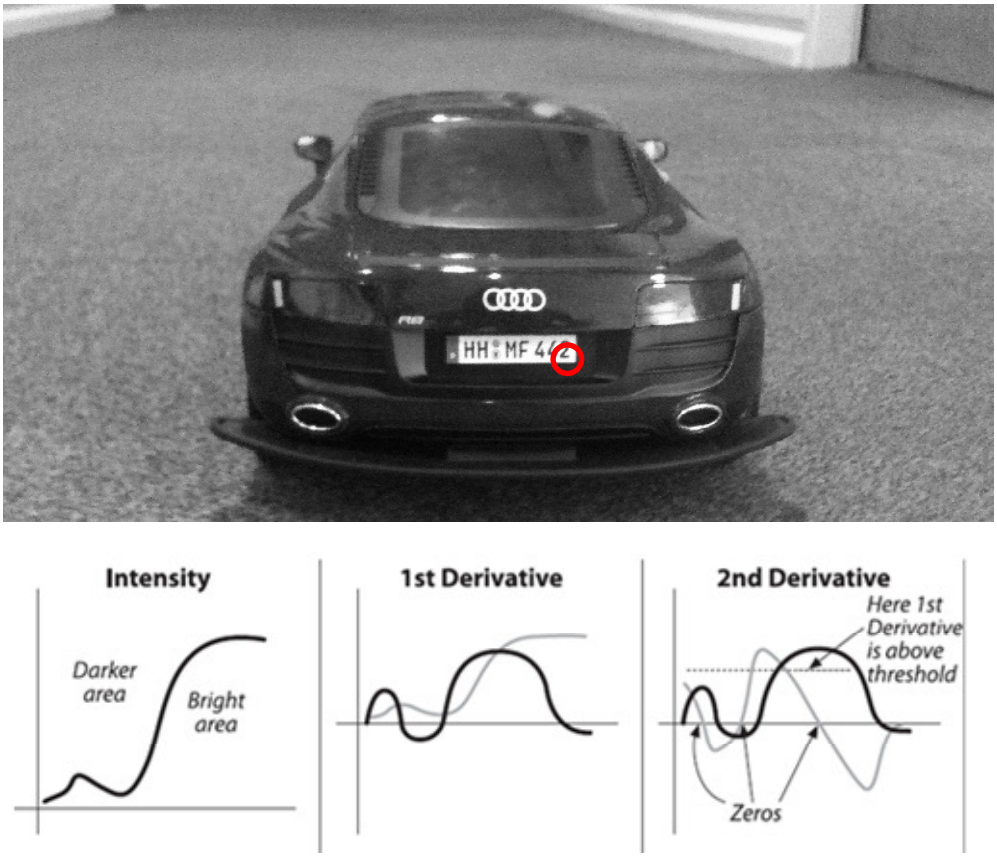


Figure 3.5: Canny Edge Detection

Canny algorithm first computes the derivatives in horizontal *(G_x)* and vertical *(G_y)* directions. Next the edge gradient and direction are determined.

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.6}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.7}$$

The edge direction is computed along one of the four permitted directions horizontal (0°), vertical (90°), and one of the two diagonals (135°, 45°). However, as we

noticed before, less meaningful edges will be detected. Therefore Canny imposes a so called hysteresis thresholding. This translates to using a threshold band pass as follows:

- if a pixel's gradient is higher than the upper threshold, it is accepted as an edge;
- if the gradient is between the threshold levels the pixel is accepted as an edge only if it is connected to an edge pixel;
- if the gradient is lower than the lower threshold the pixel is rejected from the edge assembly.

Figure 3.6: Canny Edge Detection on Input Frame

The result is a binary image which is prepared for contour finding. The threshold levels used are 0 and 50 since when the vehicle is at a considerable distance from the platooned one; we still want to be able to detect the number plate. The aperture size used is 5 since it seems as an ideal compromise between processing time and accuracy.

3.3.4 Binary Thresholding

After several tests with Canny edge detector as an image preprocessing step for contour finding it seemed that we would need, in some cases, a different preprocessing method to benefit of a fully reliable system. When the leading vehicle is at a

considerable distance compared to the platooned one edge detection, using Canny fails in finding the exact contour of the number plate. This subsequently affects the next steps. Further more, because all the action is dynamic and the vehicle can travel both in shady and sunny conditions the fixed thresholding levels of Canny seem not to produce the expected result.

Since our number plate background is white, in contrast with the black bumper of the vehicle, a logical and simple transformation that would convert the grayscale image into a binary one ready for contour detection would be some sort of thresholding.

As we have seen in the literature review, this method is popular among license detection systems. The basics of thresholding show that sometimes we need to process only pixels above a certain level and reject those that are below. OpenCV function *cvThreshold()* does exactly this. We are interested only in the thresholding method *CV_THRESH_BINARY*. Thus, every pixel in the input image will have a new value corresponding to its relation to the chosen threshold. Therefore, the pixel values above the presented threshold will have a new value transferred to the function as *max_value*. Those below will have a new value of 0. The result of this operation can be compared to that of Canny edge detector in the sense that the output image will be an array of binary values (0, 255).

Figure 3.7: Binary Thresholding for Different Thresholds

As seen in the above figure, the main drawback is that this method relies on a fixed threshold. Because of different illumination conditions that we have to face we would need a form of adaptive thresholding able to cope with these differences. One of the algorithms revised at the beginning of the chapter used a fixed thresholding level and subtracted it until a number plate was found. This will not suite our algorithm since we are using a high resolution image and at each loop with no result and we will lose valuable processing time.

Therefore we turned our attention to the OpenCV function *cvAdaptiveThreshold()*. This computes a pixel by pixel weighted average on a *blocksize* by *blocksize* area. From the value resulted from this weighted average a constant value passed as *param1* is subtracted. The pixel in the middle of the kernel is thresholded to the value resulted. It is worth mentioning that OpenCV implementation provides two distinct methods for calculation of the weighted coefficients. We will use *CV_ADAPTIVE_MEAN_C* where all the coefficients are equal. After a thorough testing, the best results were provided when using a block size of 7 and a value of *param1* of 2. A result of this step can be seen in figure 3.8.

Figure 3.8: Adaptive Binary Thresholding on Input Frame

3.3.5 Image Dilation

As we mentioned before, our number plate detection algorithm relies on the quality of the preprocessing of the image. Therefore we can consider that the contour finding and the processing steps involved afterwards comply with the computing principle garbage in, garbage out. In other words if we pass an image where the number plate region is not well distinguished, then we will not be able to find the plate.

With this in mind we inspected the results of edge detection and binary thresholding. In some of the cases, the edges were broken making it impossible for the

following processing steps to find a positive result. The same can be said about binary thresholding, when the followed vehicle is at a considerable distance and the thresholding process will output an image where the white background of the plate is barely visible.

Therefore it seems that in both of the cases a better result would be obtained if we can in some sort of way unite broken edges or enhance the white background. Removing noise, joining edges or isolating individual elements in a binary image is usually done by the use of a morphological operation. The basic morphological operations are erosion and dilation. In the following we will present the benefits and the interworks of dilation. Erosion can be considered the equivalent opposite of dilation.

Both dilation and erosion are products of the interaction between the input image and a structuring element. The structuring element has an anchor point and a distinctive shape. Although OpenCV offers the user the ability to design his own structuring shapes we did not do this, since the basic rectangle offered significant results.

To better understand the effects of dilation on our image let us consider *A* as the input array and *B* a structuring element. If $(\widehat{B_s})$ is the structuring element shifted by *s*, then dilation is the array composed by all the possible shifts that satisfy:

$$A \oplus B = \left\{ s \middle| \left(\widehat{B_s}\right) \cap A \neq \emptyset \right\} \tag{3.8}$$

In other words dilation computes the maximal value over the area covered by the structuring kernel and replaces the anchor pixel with this value. This subsequently causes bright regions to expand. This growth is the origin of the name “dilation”. A suggestive result of dilation can be seen in figure 3.9.

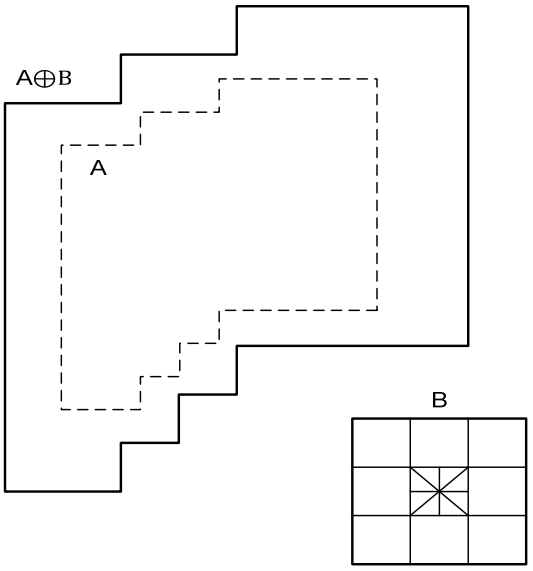


Figure 3.9: Dilation on Array A with Kernel B

OpenCV implements this transformation with the use of *cvDilate()*. This function takes two *IplImage* structures, one as an input and one as an output and performs the above described operation using the square structuring element with a base of 3 pixels.

The results obtained after dilating the image as a pre contour finding step were really good, the broken edges were united back while the white background of the number plate became more distinguishable.

3.3.6 Contour Finding

Although previous processing steps have separated the number plate in the image , while reducing the amount of data associated, they do not tell us anything about the edges as entities themselves. A simple and effective way to gain this type of information is to separate the edges into contours. In some way it can be compared to what we do when we look for a particular object with our own eyes. We try to classify the objects and to rule out those that do not match the searched characteristics.

To begin with we should define what a contour is. A contour is a series of points that describes a curve in the image. OpenCV provides contour finding with the help of *cvFindContours()* function. The input image accepted by this function has to be a binary image containing only the curves and the associated background. To better understand how contour finding works and to familiarize the reader with the functionality provided it is worth taking a look at the following example.

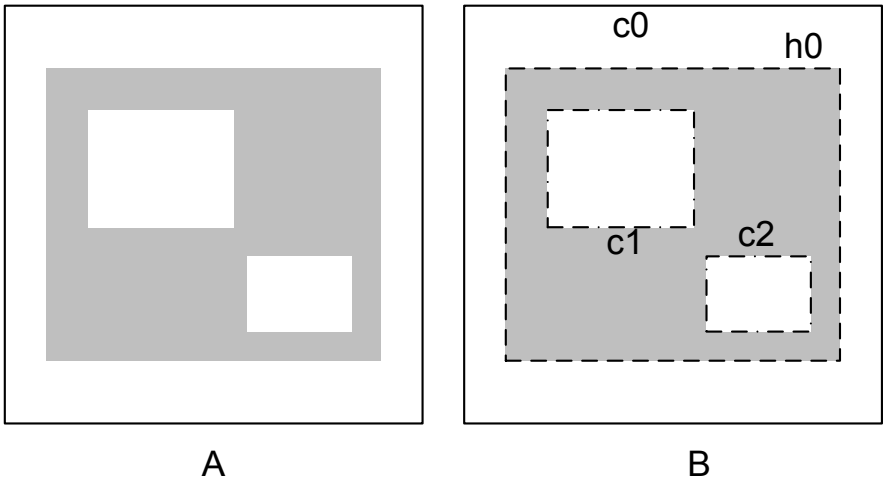


Figure 3.10: Contour Detection on Sample Image

Consider the image in figure 3.10a as an input image for contour finding. We can straightly away notice that the image is in a binary form containing only pixels of

two values. To simplify the concept behind contour finding, consider that a contour is found by following all the curves in the image from the start to the end and describing them by a sequence of points. As you can see from the result of contour finding in the lower part of figure 3.10b, OpenCV distinguishes between two types of contours. The single dashed line represents a transition between a black and a white area. This is considered to be a contour while a transition from white to black is considered to be a hole.

These two types of transitions are important to be distinguished since they play a key role in detecting objects. They are used as a classifying criterion for the returned structure which we will discuss shortly. If we think back at our input image and in particular at the black to white transition between the number plate and the rear bumper, it seems that we will always will be looking for a hole.

Let us take a moment and inspect what happens when we pass an edge image to *cvFindContours()*. Because the image only contains the background and a light curve, symbolizing an edge, OpenCV detects a hole for every contour found. This is because on every edge we will have a black to white to black transition. Nevertheless, the number plate will be fenced both by a contour and a hole. Therefore it will be enough to search for the holes.

With this in mind, let us inspect the way in which OpenCV returns contour information. OpenCV computes contours as sequences, thus every curve in the input image is stored internally as a list of points. This list describes a contour or a hole in the input image. By setting the return type mode, the list describing the contours found can be arranged as a leveled structure. This property is exploited by our algorithm since we will process only the holes. The return mode used is *CV_RETR_CCOMP*, a two level structure, where holes are located on the second level.

Nevertheless maybe the most important argument of *cvFindContours()* is the contour finding method . Since we are interested in a classification of the contours based on a shape analysis, we are interested in a sequence that contains only the ending points of a segment. For example, in the case of a curve describing a square we would like to have a sequence of four points describing the corners.

A result of this processing step can be seen in figure 3.11. We can notice that the curves are separated into entities. From now on we have to search through the results for the curve which has the highest resemblance to a number plate. Therefore in the following section we will elaborate some general searching criteria that number plates observe.

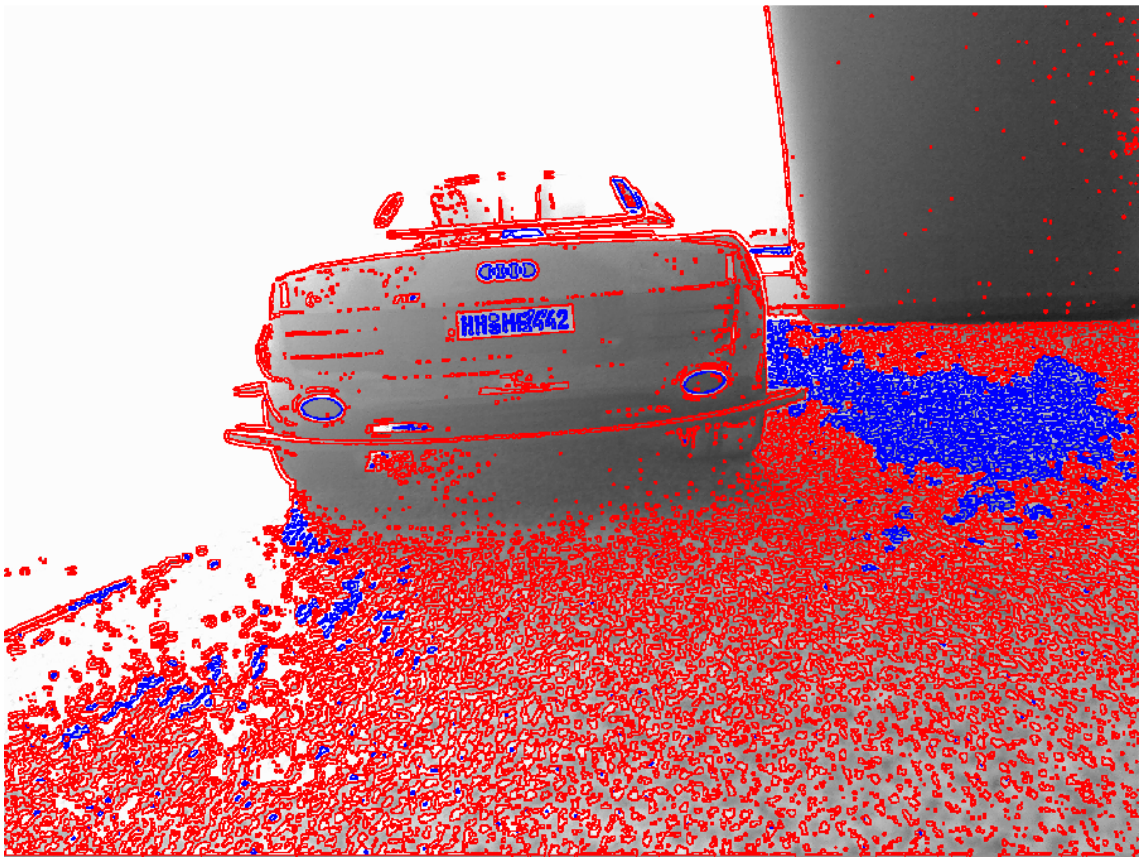


Figure 3.11: Contour Detection on Input Frame

3.3.7 Polygon Approximation

From now on we have to establish some rejection steps so that at every succession step we will have less and less number plate candidates, until we find a candidate with the expected properties.

As a first step, we will classify the number plate as a square, thus a four corner shape. As we described in the previous subchapter we will have a sequence of "important points", describing a contour, therefore a sequence of points where the curve changes its direction. In an ideal case a square will be only represented as a four points sequence. Due to the different illumination conditions as well as the image noise, this will not happen all the time. To better understand what happens, please take a look at figure 3.12. We can note that in this case the contour will not be represented as a 4 point sequence.

Figure 3.12: Number Plate with more than 4 Major Points

The purpose of this processing step is to find a way of describing a shape with a similar one with fewer vertices. An algorithm that performs this approximation is the Ramer-Douglas-Peucker [29]. To understand how the algorithm works and to familiarize the reader with its OpenCV implementation, let us take a look at the picture in the figure 3.13.

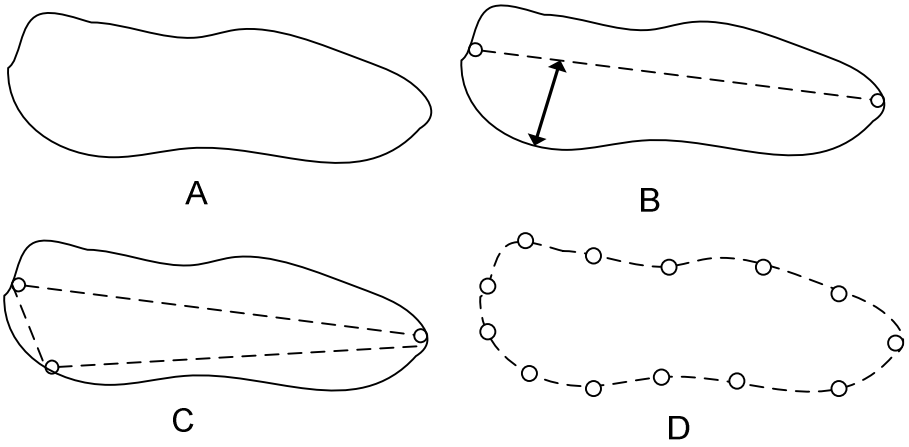


Figure 3.13: Polygon Approximation

As we can see, the algorithm starts by connecting with a line two external points of the contour (panel B). Next, the furthest point from the line just drawn is searched and added to the approximation (panel C). This process is iterated until the next chosen point is at a distance from the current approximation less than the precision parameter passed to OpenCV's function *cvApproxPoly()*.

3.3.8 Number Plate Characteristics

As mentioned previously we have to pass the found contours to several rejection steps until we find the candidate that matches all the criteria. Therefore our rejection steps have to rely on as many as possible characteristics of the number plate, thus eliminating the possibility of finding a false candidate.

A number plate will always be described by a square, a structure with four corners, represented after polygon approximation as a four points sequence. We can note that in figure 3.11 (contour finding), that most of the found contours represent small edges in the road surface. Therefore we need a sorting method that can easily distinguish between the type of "noisy contours" and possible number plate candidates. Thus, we will impose that further processing should be performed only on contours which have an area larger than a presented threshold. Nevertheless we want our contours to have a convex shape, not a concave one.

3.3.9 Angle Detection

At this stage we have a four corners polygon with an area larger than a presented threshold. We still have to refine our search. Therefore we turn our attention to the common features of number plates.

A number plate will always have an angle of 90° between its sides. Since the contour sequence describing a curve in the binary image is a sequence of four points describing the vertices where there is an angle change, we can compute this information to find the values of the angles.

Therefore we need to find the angle between the vertices. To do this we will use the trigonometric calculation of the cosines between two vectors. The formula is detailed below.

$$cos\alpha = \frac{\vec{u} * \vec{v}}{|\vec{u}| * |\vec{v}|} = \frac{x_1 * x_2 + y_1 * y_2}{\sqrt{x_1^2 + y_1^2} * \sqrt{x_2^2 + y_2^2}} \quad (3.9)$$

where x_1, y_1 are the coordinates of $\vec{u}$ and x_2, y_2 of $\vec{v}$. We will threshold all the possible candidates to an angle value between the vertices of 80° to 100°, so that we can compensate for the leader lateral movements.

3.3.10 Final Verification Criteria

As a final verification step, the candidates are measured and compared to the presented values. Therefore we impose that the number plate width should be in a certain interval. Nevertheless we want to see if the overall dimensions are the same as those of a real number plate. Due to the fact that the distance to the leader vehicle is dynamic, so are the values of width and height. Thus comparing either the width or height to a presented value is not a feasible option. The way to avoid this is to compare the aspect ratio of the detected number with an imposed interval. This is because the aspect ratio (width/ height) will be the same at any distance.

It can be argued that at this stage a rectangular box with the same aspect ratio as the number plate can be detected as being the plate that the vehicle has to observe. By limiting the input data to the region of interest and by passing the candidates through multiple verification stages we try to limit this as much as possible. In the test conducted this happened very rarely, therefore there was no need to compensate for it.

Nevertheless in real case scenarios, a further processing step can be added to the algorithm that can distinguish between boxes and real plates. An imposed edge count thresholding can be applied, with successful results. This is because the character region of the number plate will generate and abundance of vertical and horizontal edges, while a plane box will not. Due to the limited processing power as well as to the fact that it did not seem to be a crucial problem, this step has been neglected in the final implementation.

The position and dimensions of the detected number plate are computed and used as an input for vehicle control. The position of the number plate is used as an input for lateral control, while the width of the plate is used as an input for longitudinal control. Thus, at each successful search, new values for speed and direction are computed and sent to the vehicle control system, using an UDP package over an Ethernet connection. The structure of the package and the speed and direction values are discussed in the chapter related to vehicle control.

We should conclude this chapter by mentioning that the algorithm has been tested before using a camera with 50 different images containing a number plate, at different distances and with different illumination conditions. The algorithm was

successful in more than 95% of the cases. We should also mention that the aim, namely to be able to detect a plate at a distance of up to 1.5 meters was accomplished.

Figure 3.14: Number Plate Detected on Input Frame

Chapter 4

Optimisations

As described in the specifications and throughout this entire paper, the most important constraint of our system is real time processing. The algorithm developed for number plate platooning has to perform the necessary computation fast enough to be able to follow the leader vehicle. Nevertheless the control platform has to perform the conversion from the UDP format to the PWM control applied to the vehicle chassis fast enough so that no package is lost. However the computation time constraints are easily solved by the control platform.

Not the same can be said in the case of the image processing algorithm. To solve the real time constraints and to be able to process the images at a rate of more than 15frames/ second (hard imposed constraint by the limitations of the video acquisition system, Logitech 9000 Pro) the algorithm has to be optimized.

The software has been developed on the author's personal computer, an Intel Core 2 Duo @ 2GHz architecture with 2GB DDR2 RAM, running the same distribution of Linux. The initial processing time on this platform was close to 15 frame/second. Therefore further optimizations had to be done to cope with the Panda Board processing. We have tried to use division as less as possible since neither the author's personal computer or the PandaBoard benefit of a hardware division block. Since we perform computation on blocks of data (images) that are loaded partially onto the chip cache memory, we have used extensively in place transformations that use the same image as input and output. At this level it is worth taking a look back at our algorithm and reviewing the contour finding step. We first find all the available contours and then perform the number plate verification criteria. However a better solution would be to find the contours one by one, since our initial checks are very restrictive. OpenCV facilitates this by the use of the function *cvFindContours().* At the end of this optimization step the computation time on the development platform was satisfying with a total loop time of less than 60ms.

However, when tested on the Panda Board the computation time was almost twice as long. Therefore further optimization had to be performed.

4.1 Image Capturing

For the next processing step we started by profiling the code. We noticed that capturing a new frame from the camera took as much as the detection algorithm. Therefore we needed to improve the image capturing process. Unfortunately there was no other way for us to improve this by the use of any OpenCV function. To better understand what we had to improve let us take a look at all the preprocessing process.

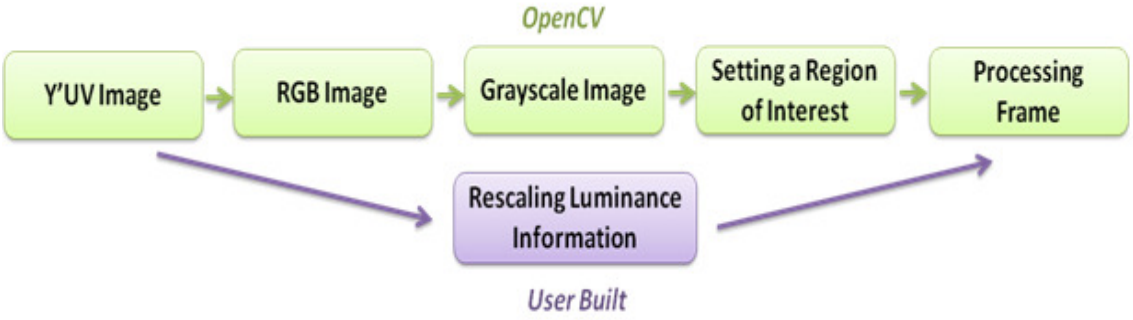


Figure 4.1: Frame Acquisition Process

The output image sent by the Logitech webcam is Y'UV 4:2:2. From this format, OpenCV has to convert the image to a RGB format as it is the standard output for the *cvQueryImage()* function. Next since our algorithm uses a grayscale image as an input an RGB to grayscale conversion is done by the use of *cvCVTColor()*. As a final step before processing, we set a region of interest (ROI) so that we ease the work of our algorithm. All these steps take up to 70ms on each new frame, thus if our algorithm does only this, we would achieve a rate of 15 frames/second.

Therefore we need to develop an interface between the USB device and OpenCV's functions which perform the number plate search. For this a fast and simple capturing function has been developed with the use of V4L library [30]. It is worth mentioning that almost all the interfaces between video devices, such as a webcam and user applications under Linux are built using the above mentioned library. Even OpenCV's capture functionality is implemented with its use.

As we said before the webcam outputs a frame in a Y'UV 4:2:2 format. This contains luminance and chrominance information for each element. However we are just interested in the luminance (*Y*) since our algorithm expects a grayscale image as an

input. In Y'UV 4:2:2 the luminance information is scaled in the interval [16, 235], thus we need to rescale it to [0, 255]. The ROI is set by taking from the input buffer only the values of interest. The results of the optimizations are significant, now the preprocessing step taking around 10ms/frame.

4.2 Parallel Processing

Although numerous optimizations have been done at the algorithm level we have not yet exploited the maximum potential of our architecture. This is because so far we have been processing the image as a single thread program, thus using only one of the CPU's cores at a time. To better understand the optimizations achieved at this level let us take a look at thread programming.

We should start by saying that each program run at user level in Linux or any other operating system is treated as a process. Thus a process is a program that is running, an address space with one or more threads executing within the address space. A thread is a lightweight process, a sequence of control within a process, that shares the same global memory within that process.

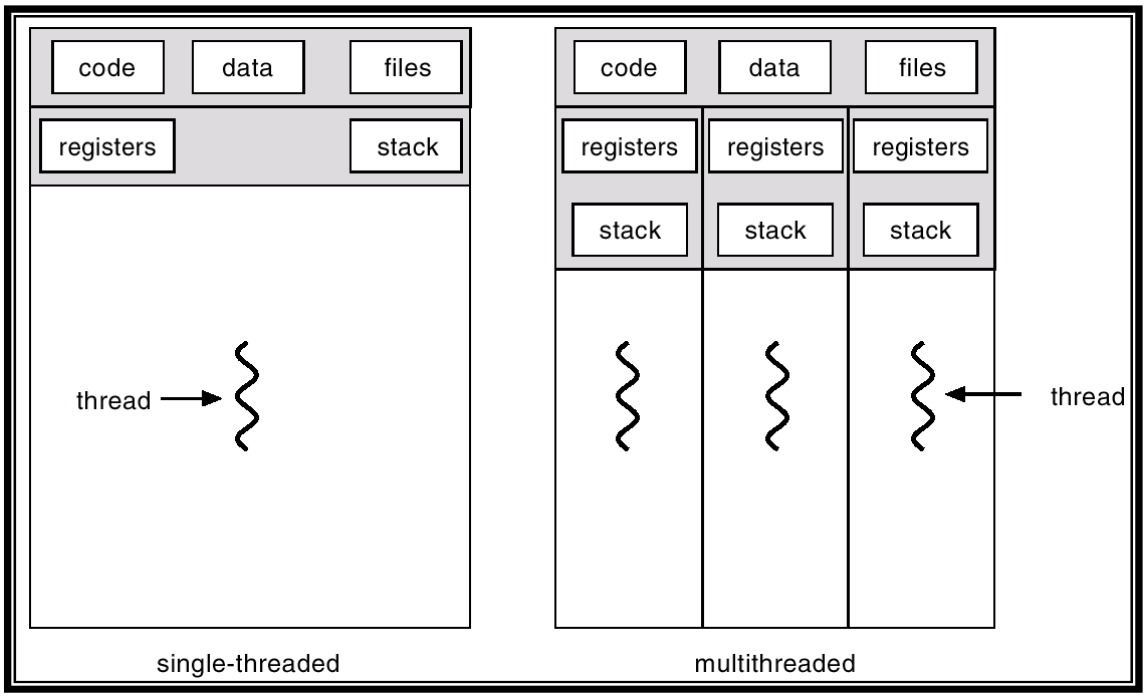


Figure 4.2: Single-threaded vs. Multithread Process [33]

Thus as seen in the figure above our aim is to have several execution threads competing for the system's resources. The main advantages of threads, over processes

are that the overhead for creating and switching between threads is less than that of a process. Although all this writing multithread programs requires careful thought.

Thread programming is made available under Linux by the POSIX [33][34] standard IEEE 10031.c. The <pthreads.h> library handles management, and creation of threads as well as that of shared resources. Thus the aim of this stage is to be able to separate our program into individual threads that compute several bits of the code. The best arrangement is to have 3 threads, one for frame capture and preprocessing and other 2 searching for the number plate in the two proposed methods, by using binary thresholding or edge detection.

However due to the fact that we are using an external library we have to make sure that the functions that we are using are thread safe. These are in turn functions that do not return a pointer to a *static*, or *global* variable. This is because we want every computation done on such variables to be atomic (one at a time). After carefully testing our functions correlated with the little information on parallel programming related to OpenCV we have concluded that all the functions that are not included in the highgui library are thread safe, thus we can perform the optimization wanted. We have also tried to minimize the use of same global variables or resources throughout our code. The only mutual exclusion used, was enforced over the function that sends the UDP package over Ethernet. We have also changed the scheduling priority of our program and threads, now having the highest priority and running as real-time processes. Nevertheless the niceness level has also been enhanced (the time slot allocated per process by a Round-Robin scheduler) and changed the scheduling type to computation expensive threads to FIFO.

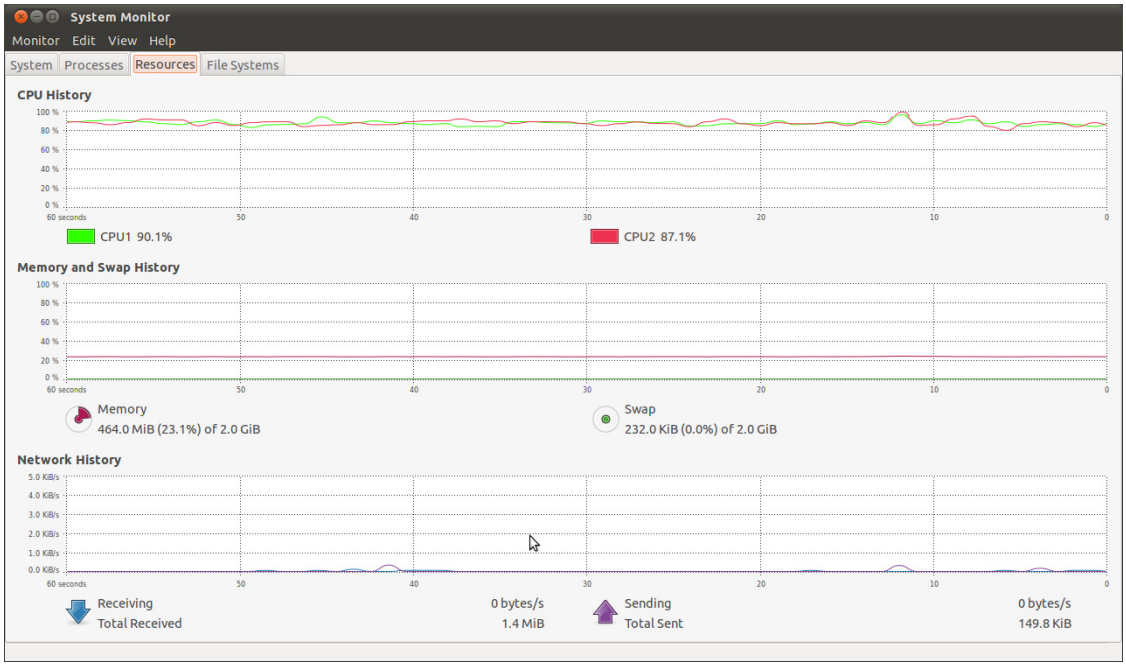


Figure 4.3: Algorithm's CPU Usage

As a result of this stage we have reduced the computation time from 90-80 ms/ frame to approximately 50-40 ms/ frame. The processor usage shown in the figure below is now constant on both cores and over 80% for most of the time.

4.3 Other Optimisations

Now that we have achieved a total processing time per frame lower than the hard constraint of our system, which is that imposed by the image acquisition camera, the only feasible way to continue the optimisation path is to change the camera with one that can query a new frame at a rate higher than 15 frames/ second. For this we tried to integrate to our system the Leopardboard 365 [35], a development board with a 5MP built in camera. The only way to integrate it to our system was to use it as a USB webcam. Although the Leopardboard can be used as a USB webcam there are no available drivers for interfacing it with Linux. For this we had to modify the UVC driver.

The first problem we faced at this stage was that the UVC driver was continuously writing warning reports to the system log file. In a 5 minutes period the log file reached a size of over 10GB. Therefore the author removed all the UVC related files and established a direct connection with the camera using V4L library. However even at this stage we faced several problems. Firstly we should mention that when establishing a connection with a camera using V4L a standard query routine takes place. Although the camera did not respond to any of the querying messages. Thus we had to use a trial and error process to find the communication parameters that the camera expected. Even after establishing the connection because of the fact that the camera did not allocate any buffers on the device the acquisition speed did not exceed that of the Logitech 9000 Pro Webcam. Nevertheless the image quality was not feasible for integration with our algorithm since the webcam does not benefit of an autofocus property. It is worth mentioning at this stage that the author is in the process of writing an application note in concerning Leopardboard integration as an USB camera.

Other optimisations had been made on the operating system's run level. Since we do not need a graphical interface we are using a user modified run level where the graphical interface is not loaded at system start-up.

Chapter 5

Conclusions and Further Work

The aims of the project were successfully achieved. We have demonstrated that platooning methods with no inter vehicle communication can be developed, using the number plate as a tracking feature. The system was designed so that further projects can be developed on the platform provided. We have throughout our work tried to leave room for further enhancements and add-ons. The autonomous vehicle is able to track the leading vehicle up to a distance of 1.5 meters. On the other hand our algorithm, although using external libraries, manages to do this in real time. The vehicle autonomy has far exceeded that of previous projects developed using the same chassis, by the use of an SMPS power supply, capable of powering the car for almost 2 hours.

However we should also mention the personal knowledge accumulated due to this project. Although before starting the project the author did not have any previous knowledge about the image processing subfield, he quickly managed to become fond of it. Thus, as a result, he and two other colleagues are planning to attend a computer vision contest, next year in Minnesota, USA. Nevertheless the project was very challenging and involved lot of extra work which was not included here but helped us to achieve our goal.

Further work can always be done on a student project. However we should point out the areas where we consider that further time can be invested in. One of them is the algorithm itself which does not have a 100% success rate, here other classification methods can be added to enhance the ones existing. Further optimisations can be done also by replacing the OpenCV functions with user built one. Here we emphasise on using ARM's specific SIMD (single instruction multiple data) instruction set, NEON, capable of operating on 128bit at a time. Nevertheless the control system has to be refined so that the leader vehicle is followed smoothly. However the bottleneck of our system is the USB camera which can only work at a rate of 15 frames/ second.

References

[1] Tyler Cowen, *"Can I See Your License, Registration and C.P.U.?",* In: Economic View Section, The New York Times, 28th May, 2011

[2] Broggi, A. Bertozzi, M. Fascioli, A, *"ARGO and the MilleMiglia in Automatico Tour"* In: Intelligent Systems and their Applications, IEEE, Issue: February 1999

[3] *"Driverless Cars"*, Online, Cited: 15th September 2011 http://www.ultraprt.net/cms/PRTdesignArticle.pdf

[4] Darpa Grand Challenge, Online, Cited: 15th September 2011, http://www.darpagrandchallenge.com/

[5] Tom Scott, *"Are you feeling lucky? Why Google's driverless cars show its technology heft",* In: The Guardian, 9th October 2010

[6] Gehring, O. and Fritz, H. *"Practical results of a longitudinal control concept for truck platooning with vehicle to vehicle communication"* In: Intelligent Transportation System, IEEE Conference, November 1997

[7] Raza, H. and Ioannou, P. "*Vehicle following control design for automated highway systems"* In: Control Systems, IEEE, Issue: 6, June 2005

[8] Shladover, S.E. Desoer, C.A. Hedrick, "*Automated vehicle control developments in the PATH program"* In: Vehicular Technology, IEEE Transactions, Issue: 1, February 1991

[9] University of California, Berkeley, 1997 California PATH Annual Report. In: 1997

[10] Arturo Dávila and Mario Nombela, *"SARTRE: Safe Road Trains for the Environment"* In: Conference on Personal Rapid Transit PRT, London Heathrow, September 2010

[11] Traxxas, *"The Ultimate Electric 4X4 Maxx Monster!",* Online, Cited: July 2011, http://traxxas.com/products/models/electric/3903emaxx/

[12] Wikipedia, *"Traxxas",* Online, Cited: July 2011 http://en.wikipedia.org/wiki/Traxxas/

[13] T.I., "Stellaris LM3S8962 Evaluation Board – User's Manual", Revision B, 9th February 2010

[14] ServoCity, *"How do servos work?",* Online, Cited: 10th September 2011, http://www.servocity.com/html/how_do_servos_work_.html/

[15] Melexis, *"US5781 Unipolar Hall Switch – Medium Sensitivity",* Data Sheet, Revision 013, November 2006

[16] PandaBoard, Online, Cited: September 2011, http://pandaboard.org/

[17] Wikipedia, *"Automatic number plate recognition",* Online, Cited: September 2011, http://en.wikipedia.org/wiki/Automatic_number_plate_recognition/

[18] Yuntao Cui, Qian Huang, *"Automatic license extraction from moving vehicles",* In: Image Processing, 1997. Proceedings, IEEE, October 1997

[19] Guangzhi Cao, Jianqian Chen, Jingping Jiang, *"An adaptive approach to vehicle license plate localization",* In: Industrial Electronics Society, 2003, IEEE, November 2003

[20] Mahini H., Kasaei S., Dorri F., *"An Efficient Features - Based License Plate Localization Method"*, In: Pattern Recognition, 2006. ICPR 2006, IEEE, 2006

[21] Xiaobo Lu, Guanghua Zhang, Bin Liu*, "Localization of Vehicle License Plate Based on Gray Level Variation"*, In: ITS Telecommunications Proceedings, 2006, IEEE, June 2006

[22] Rattanathammawat Preemon, Chalidabhongse Thanarat H., *"A Car Plate Detector using Edge Information"*, Communications and Information Technologies, 2006, IEEE, September 2006

[23] Wenjing Jia, Huaifeng Zhang, Xiangjian He, *"Mean shift for accurate license plate localization"*, Intelligent Transportation Systems, 2005, IEEE, September 2005

[24] Park S.H., Kim K.I., Jung K, *"Locating car license plates using neural networks",* Electronics Letters, August 1999

[25] Gary Bradski, Adrian Kaehler, *"Learning OpenCV",* First Edition, O'Reilly, September 2008

[26] OpenCVWiki, *"OpenCV"*, Online, Cited: June – September 2011, http://opencv.willowgarage.com/wiki/

[27] Moo K. Chung, *"The Gaussian Kernel",* University of Wisconsin – Madison, September 2007

[28] Svetlana Lazebnik, *"Edge Detection",* University of North Carolina, Chapel Hill, 2009 Lectures

[29] Keith Clarke*, "Generalization and Structure-to-Structure Transformations",* Analytical and Computer Cartography, University of California, Santa Barbara

[30] Michael H. Schimek, *"Video for Linux Two API Specifications",* Revision 0.24, March 2008

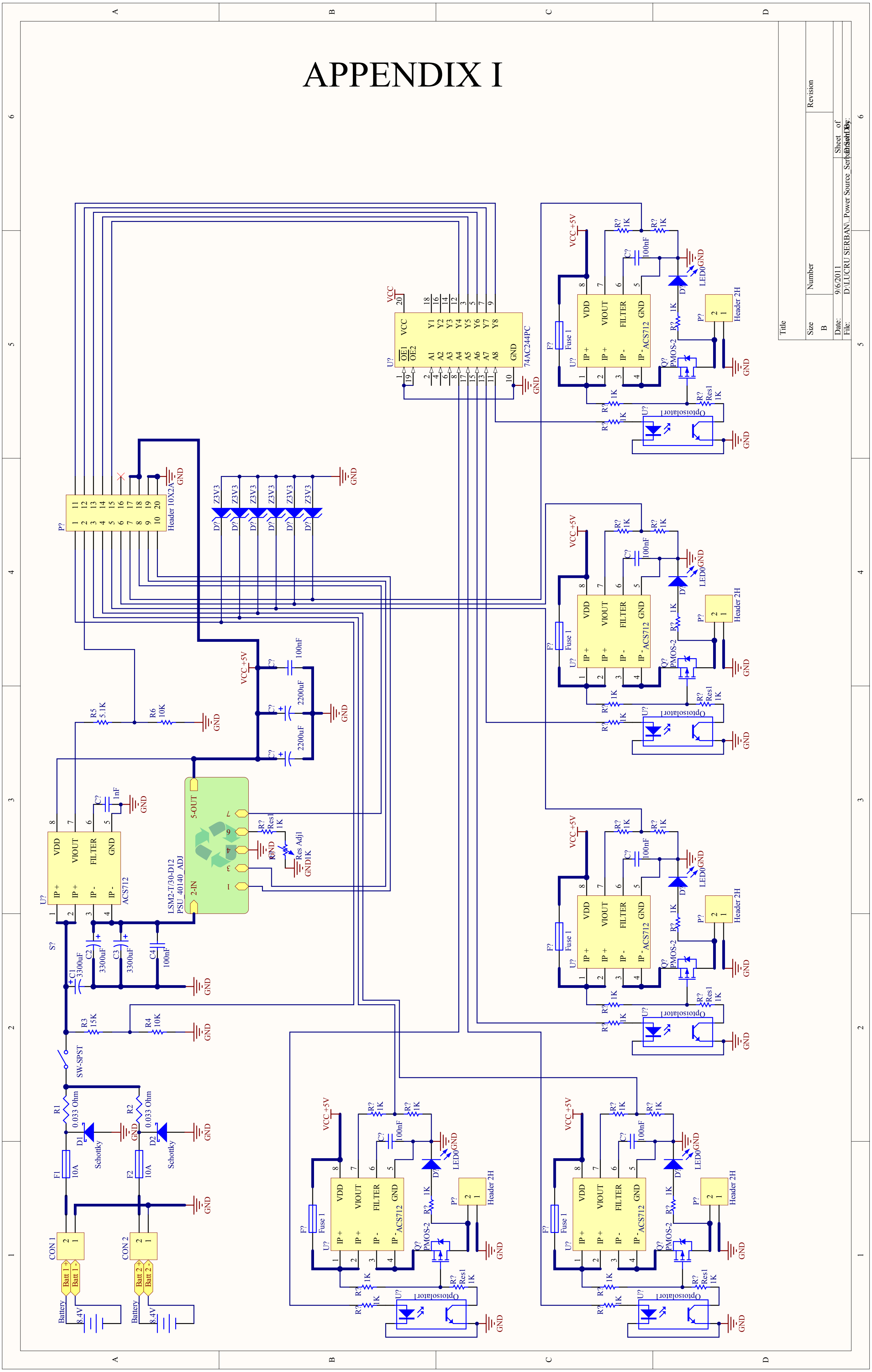
[31] Avanzini, P. Royer, E. Thuilot, *"A global decentralized control strategy for urban vehicle platooning using monocular vision and a laser rangefinder"* In: Control, Automation, Robotics and Vision, 10th International Conference , IEEE, December 2008

[32] Avanzini, and P. Thuilot, "*Urban vehicle platoon using monocular vision: Scale factor estimation"* In: Control Automation Robotics & Vision, 11th International Conference, IEEE, December 2010

[33] Blaise Barney, *"POSIX Threads Programming"*, Lawrence Livermore National Laboratory, Online, Cited August 2011, https://computing.llnl.gov/tutorials/pthreads/

[34] Ayman Abdel-Hamid, *"Computer Network Architecture and Programming - Threads",* Computer Science Department, Virginia Tech, Spring 2006

APPENDIX I



74AC244PC
ACS712
Optoisolator1
PMOS-2
Header 2H
Header 10X2A
LSM2-T/30-D12
PSU_40140_ADJ
5-OUT
2-IN
Res Adj
VCC+5V
GND
Z3V3
SW-SPST
Schottky
0.033 Ohm
Fuse 1
LED0
CON 1
CON 2
Battery
8.4V
Title
Size
Number
Revision
Date:
File:

APPENDIX II

```
/* SAMPLE CODE            */
/* Main processing thread */
int search_no_plate_canditates_thread_1( IplImage* input)
{
   CvSeq* current_contour;
   int i,j;
   CvContourScanner scan_contour;
   No_plate current;
   CvPoint pt[4];
   CvSeq* result;
   double s, t;

   /* Preprocessing Step */
   cvSmooth(input, temp, CV_GAUSSIAN, 7, 7);

   /* Search Loop
                     */
   /* Sample Code, normally separete threads Canny/Adaptive Binary */
   for (j=0; j<2; j++)
   {
         /* Adaptive Binary Thresholding, input for Number Plate
         Detection */
         if(j == 0)
         {
                  cvAdaptiveThreshold(temp, thresh_1, 255,
                  CV_ADAPTIVE_THRESH_MEAN_C, CV_THRESH_BINARY, 7, -2);
                  cvDilate(thresh_1, thresh_1, 0, 1);
         }

         /* Only if number plate was not found, Canny is more
         computation expensive */
         /* Canny Edge Detection, input for Number Plate Detection
         */
         if(j == 1)
         {
                cvCanny(temp, thresh_1, 0, 50, 3);
                cvDilate(thresh_1, thresh_1, 0, 1);
         }

            /* Contour Finding Sequence - Contour Detection Start */
            scan_contour = cvStartFindContours(thresh_1, storage_1,
            sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE,
            cvPoint(0, 0) );

            /* Candidates Search Loop */
            while( (current_contour = cvFindNextContour(
            scan_contour)) != NULL )
        {
            /* Polighon Approximation Step */
```

```
        result = cvApproxPoly( current_contour,
        sizeof(CvContour), storage_1, CV_POLY_APPROX_DP, 2,
        0 );



/* If the contours have 4 major points, convex, and
expected area */
if( result->total == 4 &&
cvContourArea(result,CV_WHOLE_SEQ,0) > 200 &&
cvCheckContourConvexity(result) )
{
    s = 0;

          /* Angle calculation for all 4 corners */
    for( i = 0; i < 5; i++ )
    {
        if( i >= 2 )
        {
             t = angle_detection((CvPoint*)cvGetSeqElem(
             result, i ),(CvPoint*)cvGetSeqElem( result, i-
             2 ),  (CvPoint*)cvGetSeqElem( result, i-1 ));
             s = s > t ? s : t;
        }
    }

  /* Number plate candidate with all the above features
and an angle between sides of 80-100 degrees */
if( s < 0.0225 )
{
     pt[0] = *(CvPoint*)cvGetSeqElem( result, 0 );
     pt[1] = *(CvPoint*)cvGetSeqElem( result, 1 );
     pt[2] = *(CvPoint*)cvGetSeqElem( result, 2 );
     pt[3] = *(CvPoint*)cvGetSeqElem( result, 3 );

          /* Calculates width/ height */
    process_no (&current, pt);

            /* Number Plate with expected ratio */
      if((current.aspect_ratio > 2.5) &&
      (current.aspect_ratio < 6))
    {
       /* Send Speed and Direction to Control Subsystem */
    sendSpeedDirection(current.speed , current.direction );
       printf("--NO plate found --\n");
       /*
       printf("perimeter = %f\n", perimeter);
       printf("Aspect ratio: %f\n", current.aspect_ratio);
       printf("Direction: %f\ndaptive_",
      current.direction);
       */
       printf("Speed: %f\n", current.speed);

              /* End finding sequence */
     current_contour = cvEndFindContours( &scan_contour );
       cvClearMemStorage(storage_1 );
              /* Exit thread with code 0 */
       pthread_exit((void*) 0);
    }
}
```

```
            }
        }
    }

    /* End finding sequence */
    current_contour = cvEndFindContours( &scan_contour );
    cvClearMemStorage(storage_1 );

      /* Exit with code 1, No Number Plate found */
    pthread_exit((void*) 1);

}
```