DEVELOPMENT OF A MULTI-SENSORED AUTONOMOUS GROUND VEHICLE

by

Fahri Serhan Daniş

B.S., Computer Engineering, Galatasaray University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in System and Control Engineering

Boğaziçi University

2009

ACKNOWLEDGEMENTS

First I thank sincerely to my advisor Prof. Dr. H. Levent Akın, for his invaluable guidance, boundless patience and motivation. This thesis would not be possible without his contributions.

My appreciations go to my jury members, Prof. Dr. Oğuz Tosun and Assist. Prof. Dr. Tankut Acarman.

I shared a lot with people of the Artificial Intelligence Laboratory (AILAB) and Cerberus Team. I would like to thank all of them so much for their friendship and support. My special thanks go to Barış Gökçe for his sincere company; Çetin and Tekin Meriçli Bros. for their support in expanding my perspective and various genius contributions especially in the robot construction; Barış Kurt, Can Kavaklıoğlu, Ergin Özkucur, Akın Günay and Derya Çavdar for their help, hands on support, comments and motivation.

I owe so much to my friend Bülent Burak Arslan. I thank him for his triggering contributions in this work. I'm also grateful to my collegues from the Department of Computer Engineering at Galatasaray University, especially to Teoman Naskali for his valuable ideas and support.

Last, but not least, I thank my family: my parents, Selim Daniş and Çiğdem Daniş, for giving me life in the first place, and with my brother, Dilhun Daniş, for educating me, unconditional support and encouragement to pursue my interests. I dedicate this thesis to them and other future members of my family.

This thesis has been supported by Boğaziçi University Research Fund Project under grant BAP06HA102.

ABSTRACT

DEVELOPMENT OF A MULTI-SENSORED AUTONOMOUS GROUND VEHICLE

In this study, we investigated the problem of constructing a small sized drive-by-wire ground vehicle with sensors and the problem of its autonomous navigation in unstructured outdoor environments. Although different sensor installations were tried on the vehicle, the focus was on the camera that would imitate the human perception using monocular cues in the acquired images.

A high importance was given to a depth perception algorithm that is used to estimate the relative maximum distances of the obstacles around. A study on the importance of the features extracted from the images was also included so that the running time of the algoritm could be decreased, which would enable the system run in real world. We have seen that lesser number of features could well be used to estimate the depths in an image.

The experiments were done on a rough terrain with trees and bushes around where the robot also had to cope with the physical conditions while trying to find the best direction to follow. The results were encouraging; the vehicle travelled longer distances; and even traversed the whole test area as the used methods got more developed, however, when the system is considered as a whole, we made an inference that more feasible models are required to compensate for erroneous estimations and, in the end, to make the navigation safer, since the final decision of the vehicle depends on many sequential parameters.

ÖZET

Çok Algılayıcılı Özerk Bir Yer Aracının Geliştirilmesi

Bu çalışmada, bilgisayarla kumanda edilebilen, alıcıları olan küçük boyutlu bir kara aracı inşası problemi ve bu aracın düzenli olmayan dış ortamlarda kendi başına dolaşması problemini araştırdık. Farklı uzaklık algılayıcı kurulumları denendiyse de daha çok kamera ile çalışılmaya odaklandık. Tek kameradan alınan resimlerden elde edilen ipuçları kullanılarak insan algısını taklit edecek algoritmaları temel aldık.

Etraftaki engellere olan en düşük bağıl uzaklıkları tahmin etmek için kullanılan bir derinlik algılama yordamına önem verildi. İmgelerden çıkarılan değişik vasıfların önemine dair ayrıca bir çalışma yapıldı. Böylelikle, gerçek dünya deneylerinde kullanılacak algoritmanın işlem zamanı düşürülebilecekti ve deneyler yapılabilecekti. Daha az vasıflarla da imgeler içindeki uzaklıkların tahmin edilebileceğini gördük.

Deneyler etrafta ağaç ve çalılıklar olan engebeli bir arazide yapıldı. Burada, robot, takip edeceği en iyi yönü ararken aynı zamanda fiziksel şartlarla da başa çıkmak durumundaydı. Deneyler ümit verici sonuçlar verdi. Araç, vasıf çıkarımı modelleri geliştirildikçe daha uzak mesafelere çarpmadan ulaşabiliyordu ve test alanının tamamını geçebiliyordu. Ancak aracın son aldığı karar, uzaklık tahmini ve yönelim gibi bir çok farklı parametreye bağlı olduğu için sistem bir bütün olarak düşünüldüğünde, hataları telafi etmek ve daha güvenli bir seyir için daha uygun modeller gerektiği çıkarımını yaptık.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

C_i	i^{th} chromosome of the population
d	Number of total dimensions
d_b	Base steering direction
d_S	Steering direction
e_h	Estimated distances on an image
f	Feature index
F_i	Initial feature set
F_s	Feature subset
FV	Fitness value
H	Number of windows in a stripe
h	Index of the estimated distances
h_M	Index of the estimated maximum distance
I_Y	Luminance layer matrix of a digital image
I_{Cb}	Blue-difference layer matrix of a digital image
I_{Cr}	Red-difference layer matrix of a digital image
K_A	Constant of the aggressiveness factor
K_B	Constant of the base steering direction
K_D	Constant of the decay parameter of the exponential
k	Number of a subset's dimensions
L_i	Laws' mask number i
M	Multiplier in the genetic algorithm fitness function
M_{linear}	Multiplier from the linear function
$M_{sigmoid}$	Multiplier from the sigmoid
M	Multiplier for the fitness function
N	Instance number in a data set
N_e	Number of intermediate indices between the maximum estimated index and the center of the image
N_{feat}	Number of features
P	Population size of the genetic algorithm

P_b	Number of the best individuals of the population
s_{buf}	Number of elements of the buffer subset
s	Number of elements of the target subset
s_{C_i}	Number of selected features in the i^{th} chromosome of the population
S	Data set matrix
SSE	Sum of squared errors
SSE_f	Sum of squared errors with addition or reduction of feature f
T_{feat}	Time to extract one feature's energy
T_{image}	Time to process an image
$T_{overhead}$	Time comsumed by the processes other than image processing
$\tilde{T}_X$	Input of the training set
$\tilde{T}_Y$	Output of the training set
$\tilde{T}_{Xbuf}$	Temporary training subset buffer input
$\tilde{V}_X$	Input of the validation set
$\tilde{V}_Y$	Output of the validation set
$\tilde{V}_{Xbuf}$	Temporary validation subset buffer input
w	Index of vertical stripes in an image
W	Number of vertical stripes in an image
x_i^t	i^{th} component of the t^{th} instance
ω_f	Weight of feature f
Ω	Weight vector
Ω_s	Weight vector from subset s
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
A	Ampere
ADC	Analog Digital Converter
AGV	Autonomously Guided Vehicle
Ah	Ampere-hour

ATV	All Terrain Vehicle
BNEP	Bluetooth Network Encapsulation Protocol
Cb	Blue-difference Chroma component of an Image
CCD	Charge-Coupled Device
CDC ACM	Communication Device Class Abstract Control Model
CF	Compact Flash
CPU	Central Processing Unit
I/O	Input Output
IDE	Integrated Drive Electronics
IP	Internet Protocol
Cr	Red-difference Chroma component of an image
DARPA	Defense Advanced Research Projects Agency
DBN	Dynamic Bayesian Network
DC	Direct Current
FOV	Field of View
FPS	frames per second
GPL	GNU Public License
GPS	Global Positioning System
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
LADAR	Laser Detection and Ranging
LAN	Local Area Network
LIDAR	Light Detection and Ranging
MB	MegaByte
MUTEX	Mutual Exclusion
MRF	Markov Random Field
Ni-Mh	Nickel-metal hydride
OS	Operating System
PC	Personal Computer
PCA	Principal Components Analysis
PWM	Pulse Width Modulation

R/C	Remote Control
RADAR	Radio Detection and Ranging
RAM	Random Access Memory
RGB	Red Green Blue color space
ROM	Read Only Memory
RNDF	Road Network Definition File
RRT	Rapidly Expanding Random Tree
RS232	Recomended Standard 232
SDL	Simple DirectMedia Layer
SUV	Sport Utility Vehicle
TCP	Transmission Control Protocol
TTL	Transistor–Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter
UC	Urban Challenge
uC	Microcontroller
UCFE	Urban Challenge Final Event
USART	Universal Synchronous and Asynchronous Serial Receiver and Transmitter
USB	Universal Serial Bus
UVC	USB Video Class
VCC	Power Supply Pin
Y	Luminance component of an image
YUV	Luma-Chrominance color space
YUYV	Luma-Chrominance color space

1. INTRODUCTION

In the first decade of the new millenium, a major leap was achieved in the scope and dimensions of robotics. The dominant industrial focus of the robotics expanded into the challenges in unstructured environments. The driverless vehicles would travel long distances by themselves in unstructured environments, and they would even run around in a dense traffic and find their way to accomplish the given tasks without any human intervention. The popular driverless cars in movies and TV series of the 80's started to become no more fictional; they are real now!

The concept of the driverless car covers highly automated cognitive and control technologies, used for both remote controlled systems and fully autonomous systems. In this work, the concept of a driverless car refers to fully autonomous systems, in particular autonomous ground vehicles.

Autonomy refers to the capacity of a rational individual to make an un-coerced decision. In robotics, this characterization of a robot implies that the autonomy is a property of the relation between the designer and the robot. Self-sufficiency, situatedness, learning or development, and evolution increase an agent's degree of autonomy [1].

Autonomous ground vehicles make up a subclass in the domain of intelligent autonomous robots. An intelligent robot is a robot with the means to understand its environment and the ability to succcessfully complete a task despite the changes in the surrounding conditions under which the task is to be performed. The word "vehicle" makes the agent concept approach to the area of transportation, where the agents are tracked or wheeled rather than legged, and safety and speed are more essential since the vehicle is generally a transporter of something important.

In the past, many attempts in the domain of autonomous vehicles were made. The most prominent ones of these attempts are the EUREKA Prometheus Project [2], the

ARGO Project [3] and the DARPA Grand Challenges [4]. The EUREKA Prometheus Project is accepted as the largest research and development project ever in the field of driverless cars. Receiving more than 1 billion dollars from the European Commission, the project defined the state of art of autonomous vehicles in early 90's. The ARGO project followed the Prometheus Project. In 1998, the vision-based automatic vehicle, ARGO, was demonstrated to the public and the research community with a 2000 km tour throughout Italy during which the vehicle drove itself autonomously. The DARPA Grand Challenge was the most recent major advancement in the domain of autonomous vehicles, and is analysed rather wider in the following chapter of this report.

The autonomous vehicles are considered to be used mainly in the transportation systems, the military, and challenging tasks for humans, such as farming, mining, and construction, where the robot will function in place of humans.

In this work, a small sized autonomous vehicle is designed and constructed. Small sized robots are generally used in indoor environments or structured outdoor environments because of the lack of capabilities to cope with the physical conditions, and the lack of embedded high processing power. Besides, the constructed robot, in this work, is supposed to navigate in unstructured outdoor environments, where the terrain, light conditions and the obstacle distribution are very complex. To accomplish this task, the vehicle is designed to have two sensors, a range scanner and a camera. We compare the utilities of the two sensors in the following chapters. We use mainly the camera to acquire distance information from the environment using an algorithm that tries to imitate the human depth perception to some degree.

Depth perception is a fundamental problem in computer vision. In robotic applications, especially when we consider autonomous agents, depth perception is a key issue for the safety of the robot.

There are two other terms related to depth estimation in living creatures: depth perception and depth sensation. An animal capable of moving in its environment must be able to sense the distances to the objects around it. Besides, humans are able to

share their experiences on distances, so the word “perception” is spared for humans whereas the word “sensation” is used for animals [5].

People can estimate depths or relative depths between different locations accurately and quickly not only by looking at the real world but also by looking at the images [6]. When the case is estimating depth on images, we cannot talk about stereo vision. To perceive depths in images, people only use their monocular vision capabilities, which are normally used to assist binocular depth perception, and a person with only one eye functioning can still estimate depths by only using monocular cues, such as motion parallax, depth from motion, color vision, texture gradient, occlusion, persepective, relative size, and peripheral vision, for natural situations such as catching, throwing, aiming, and driving.

In this work, the interest is on how the humans’ perception capabilities can be imitated by machines and the aim is to increase the applicability and the accuracy of a depth perception algorithm previously defined in another work [7], on real robots. Also, a method of using dimensionality reduction techniques on depth perception is proposed. While previously a robot vehicle was to navigate without much consideration on the processing power, our work presents a solution that is supposed to work under strict processing power constraints.

In Chapter 2, we provide some background information on autonomous vehicles and the algorithms that run on these self guided vehicles. The background information comprises the concepts of the domain of intelligent vehicles, their history, and information on construction of an autonomous ground vehicle. We devote several pages of this literature survey on the DARPA Grand Challenges since they are accepted as important milestones in the domain of AGVs. The background also includes the techniques to extract depth information from video images as we will be deeply concerned with this concept in the following chapters.

Chapter 3 is dedicated to explaining the design and construction of a two-sensored autonomous vehicle. The chapter states the problems of the construction of an AGV in

electrical, electronical, computational, and communicational means and the methods pursued to solve these problems on a real world testbed. In addition to the introduction of the hardware for an AGV, an implementation of a complementary software architecture design for a robotic controller is also elaborated here. Information on a supporting software pack is also added in order to not leave the whole architecture incomplete.

In Chapter 4, a recent approach on depth estimation and obstacle avoidance is discussed. The methodology of the depth estimation pointed in the background information is redesigned for the robotic applications and the implementation on a robotic controller is detailed in this chapter.

The thesis is concluded with a final discusion on the construction of an autonomous vehicle and the depth estimation algorithm to be used for outdoor navigation.

2. BACKGROUND

This chapter is dedicated to the previous research literature and the complementary information for the following chapters.

2.1. Early Work on Autonomous Robot Vehicles

This section brings a general view to autonomously guided vehicles, based on recent research results. The papers discussed in this section may seem to be outdated, but they state the general necessities, such as perception, control, or algorithm needs that are still valid in the area.

An intelligent autonomous vehicle is expected to be capable of navigating safely in its environment, sensing the environment as well as its own internal states, interpreting its sensor information to refine its location belief and determining obstacles in its local environment and finally generating guidance and control commands that are adaptive to spatial and temporal variations in the environment (e.g. changes in payload, road surface, acceleration constraints etc.) [8].

A wide range of special purpose and generic autonomously guided vehicles (AGVs) research programs had been initiated for several years, prior to 1992 [8]. The major research programs focused on sensor management, multi-sensor data fusion and machine perception [9, 10] , goal specification and planning [11, 12], navigation [13, 14], guidance and motion control [15, 16, 17], which also cover the programs being worked on nowadays.

In [18], some architectures of autonomous vehicles are stated as to be able to reduce the specific robotic problem complexity, to expand the market for subsystems through the use of control protocols, to provide portability, inter-operability, and modularity in software, to reduce time, cost, risk, and initial investment, and to provide technology transfer between application areas. To supply these needs, a sample design

of a multi-layered architecture is given. This architecture (for real-time intelligent control) is composed of levels and sublevels of sensory acquisition and processing, world modeling, and task decomposition.

For sensing and data fusion, the essentials are defined for any sensor system as accurate modeling of the sensor and its noise, modeling uncertainty of the sensor reading and relating the sensor abilities and limitations to the measurement task. To overcome the limitations of a single sensor, data fusion is considered, being formed of subprocesses, position estimation and signal processing, association or consensus (confliction removal), and attribution and assimilation of data fusion (merging).

For navigational purposes, the considered approach requires world modeling, landmark recognition, and learning the landmarks. Harris and Channley [8] mention also the path planning problem and divide it into two stages, that are the generation of a global strategy in terms of immediate goals and producing commands at low level to achieve these goals. Finally at the lowest level, the servo level, the motion controller must cope with the vehicle dynamics.

Another useful article having a global overview dates back to 2000 [19]. The main focus is on the vision module in the AGV area. It is discussed that an autonomous system should be robust enough, highly reliable, while not costing more than the vehicle's original price and having a user-friendly human-machine interface.

The importance of the vision system comes from on the low cost of the on-board system. The author gives the vision system as the cheapest of all, which will be discussed together with the advantages of different sensors, in Section 2.1.2.

2.1.1. Robotic Platforms

Hardware specific information appears mostly in the introduction and conclusion sections of the papers where the testbeds are specified and the experimental results are presented, except when the subject of the paper is directly on building of the vehicle.

An example of these exceptional papers is the one that focuses on the creation of a low-cost autonomous vehicle [20]. In this example, a small sized autonomous vehicle is constructed from a radio controlled car chassis by equipping it with Global Positioning System (GPS) and ultrasonic sensors. The main task is lowering the cost of the project. Therefore, while the sensors vary in complexity, price, and output format, among the various sensors (touch, infrared, laser range finders, stereo cameras etc.) they decided to use ultrasonic range finders for their ability to perform under low visibility, scalability and of course their low cost. These ultrasonic range finders are located in the front of the vehicle with an offset of 45 degrees from each other. In addition, the paper represents the mathematical model on how to calculate the distance to an obstacle, assuming that these devices emit a 40 kHz ultrasonic pressure wave.

For navigational purposes, the choice is on the GPS receiver which eases the calculation of the traveled path from a starting point, preventing the accumulated error which would emerge from the information bearing upon previous results (compasses, inclinometers, or rotational counters). GPS is less costly than any other selection; however, signal reception may limit the use of GPS systems, which would be avoided by using differential GPS, but at additional expense.

Since the spectrum of the project in [20] is considered to be very narrow, processing power is limited to small-sized micro-controllers. The criteria are that the micro-controller has sufficient number of ports, processor speed to complete tasks within required time limits and adequate memory to run all necessary programs. Additionally, the presence of serial ports, timers, counters, PWMs, and ADCs would improve usability. In this base, Rabbit 2000 [21] is selected as the processor, which can hold up to 50,000 lines of C code, is easy-to-use development environment, has RS232 and I2C drivers, and moreover is inexpensive, while the comparison is not specified.

As some additional but very useful information, the paper provides information about management of electrical power for the vehicle and its embedded hardware. Usage of electrical power is given in detail, and reasons to use any battery or output follow. The most significant and important power-related subject is that the power

supply for the DC and servo motors of the vehicle should be separated from the control hardware power supply [20].

Another such example of R/C car uses vision for the control purposes. The purpose of this work is building an infrastructure for the competition called "micromouse" [22]. They use a toy car called "little R/C buggy" and a CCD video camera. The computation is performed by players' PCs. As the paper is rather dated, and there have been many improvements in the sensor and micro-controller technology, the proposed solutions are acceptable, but can be replaced by much easier solutions.

The third example in this scope is again about building low cost vehicles, but the suggested behavior is the simple reactive behavior [23]. The hardware consists of LEGO bricks with integrated sensors and motors. The objective is to construct two robots that can follow walls and avoid obstacles while moving in an artificial environment. The used sensors include touch sensors, infrared receivers, and transmitter diodes. The interesting part in this article is that one robot uses only analog components like resistors, diodes, transistors, and relays, whereas the other has its controller board. The more complex robot is said to react more robustly; however, for our purposes the robustness should be obtained in unstructured environments and the behavior should be more complex than a simple reaction.

A very simple example involves a R/C car, an 8-bit micro-controller, and a GPS receiver [24], assembled into an AGV for educational purposes. Since the purpose is education, the device should be expendable, and thus it should be as cheap as possible. Using just the GPS data, state estimation, navigation, and control routines are performed with the vehicle.

2.1.2. Sensors

Sensors are (and need to be) widely used in robotics, but tactile sensors and acoustic sensors are of no use in automotive applications because of vehicles' high speed and their low detection range [19], but on model vehicles or smaller mobile

robotic applications there are examples of range sensors like sonar sensors installations [25] [26], due to their low cost.

Laser-based sensors and millimeter wave radars detect the distance by the travel time of the signal they emit and the reflection from the object. These are called active sensors. The drawbacks are their low resolution, and slow scanning speed. The millimeter-wave radars are more robust to rain and fog, but more expensive. Visual sensors are the passive sensors. They do not need to alter the environment so they have advantage over the active sensors in some way. Image scanning is fast enough for intelligent vehicle applications and they can be used for specific applications such as lane localization, traffic sign recognition, and obstacle identification, where visual information plays the fundamental role, but vision sensors are less robust to fog, night, or direct sun-shine conditions [19].

Active sensors can also measure the quantities like movement directly by performing less computation compared to visual perception. On the other hand, active sensors are not scalable, that is, when multiple intelligent vehicles travel in the same environment, the interference between these sensors can be critical. Foreseeing the massive usage of AGVs, passive sensors seem to be advantageous [19].

Since visual sensors are less robust to lighting conditions, their usage is prefered in artificially illuminated environments, especially indoor environments. In such conditions, much more efficiency is obtained after properly calibrating the camera. In [27], the camera is used to create maps and make the robot localize and navigate autonomously using a single camera.

Different types of cameras also increase the efficiency in specific ways. For example in [28], they use a pinhole camera to perceive depth, using the mechanism of the eyeball as a basis. In [29], among the various other sensor types, like GPS, pedal pressure sensors, inertial forces, vehicle internal state sensors, microphones etc. , different camera types are also used to collect data. The aim of the project is to enhance the driving safety for road vehicles and developing active safety systems, so that two

cameras, for night vision and day vision, are configured to track the face of the driver from both sides to detect tiredness and sleepiness. A third camera with a large field of view is also pointed to the road ahead to capture the road view.

In [24], the designed vehicle uses a single GPS sensor. With no additional sensors they make the vehicle navigate to the selected points. The sensor provides the vehicle's position, speed, and heading. The accuracy of the GPS receiver used in this project is 15 feet range at minimum, which leads to a bad resolution for such a small vehicle, but the sensory data is enhanced by making estimates on localization using Kalman filter.

2.1.3. Processing Power

When the processing of large volumes of data is considered, there are two possibilities to cope with the real-time processing problem. Either powerful architectures optimized for real-time applications can be an answer to this problem, or we may find other ways to reduce the data volume size to be processed. Using the fast technological evolution with this reduction we may reach real-time processing power at some degree with high volumes of data, but the architectural approach, or in other words, supplying much more processing power would not be economically feasible, thus searching ways to enhance the algorithms have always been considered to be much more feasible [30].

2.2. DARPA Grand Challenge Overview

In this section, we present a compilation of the DARPA Grand Challenge technical papers and other related papers to extract necessary information on the construction of an autonomously guided ground vehicle. The scope of the section varies from the hardware chosen by the teams to the high level software architectures that combine the information gathered from various sensors.

The first Grand Challenge was announced by the Defense Advanced Research Projects Agency (DARPA) in 2003. The aim was to develop autonomous vehicles that could navigate in the desert or on the roads at high speeds. The generation of

the competition was a response to a congressional mandate that a third of the U.S. military ground vehicles be unmanned by 2015 [4].

2.2.1. Grand Challenge 2004

The first Grand Challenge was run in 2004, and no competitor attained to win the race while the Red Team made the best progress by having their M998 HMMWV vehicle, SandStorm, travel more than 11 km in the desert [31]. The team describes the techniques they used in the construction of the vehicle superficially, but it is apparent that the sensing and processing did highly depend on the active sensors, such as the long and short range Light Detection and Ranging (LIDAR) and Radio Detection And Ranging (RADAR) systems. Even though no team was successful enough to win the race, SandStorm set a new benchmark for autonomous capability and provided a template on how to win the race [32].

2.2.2. Grand Challenge 2005

The next year, a similar challenge was completed by five vehicles, in which Stanley, a Volkswagen Touareg, converted to an autonomous vehicle by Stanford University researchers, won the race by arriving at the finish line under 7 hours on the course of 244 km, before SandStorm and its cousin, H1ghlander. The set of various sensors mounted on Stanley, included five SICK laser range finders, a color camera, and two antennae of a forward-pointed RADAR system for the environmental perception. The vehicle also had global position retriever sensors like GPSs and compasses, and an inertial measurement unit in order to obtain the internal state information of the vehicle. An outstanding character of Stanley was its variable-height air suspension and diesel-powered engine, thatwas extremely suitable for off-road terrain navigation. In general the teams preferred grand off-road vehicles [33, 34, 35, 36]. This has two main causes: the race took place in the desert and the off-road vehicles provide ground clearance and stability to cope with the rugged conditions, and also the vehicles had large payload capacities [37, 38].

The two Red Team vehicles were the followers of Stanley. We see that the two vehicles of the Red Team had nearly the same construction scheme for the vehicle hardware, the sensors and the software architectures [37, 38]. Again they used similar sensor suites to that of the Stanford Team. We can say that the selected sensors are mostly range scan typed sensors, like radar and laser scanning based sensors. Generally, a placement that provides redundant frontal environmental information was chosen. For better coverage of the environment some teams used sensor support suites, such as gimbals, that corrects the angle of a long range scan sensor [36]. While the range scan sensors were mainly used for obstacle detection to avoid collisions, they also provided the terrain topography and the estimated road condition to select the appropriate velocity. The video based sensors were of low usage compared to the laser based range scanners, but there were some teams that used stereo cameras efficiently [39] and colored mono cameras [37] or both [40, 41], even though we notice that these approaches did not bring much advantage to finish the race.

All vehicles in the Grand Challenge 2005 needed to possess an advanced hardware set that could afford to process high volumes of data in low latencies. Generally, several gigahertz machines were mounted on the vehicles, with memories in the order of several gigabytes and with data capacity of terabytes to supply redundancy in processing and data logging [37, 40, 41, 42, 43]. Sensing, path planning, and control of the vehicles could be possible with such powerful machines, and the redundancy in processing power brought reliability.

We notice that a standard robotic architecture is adopted by many of the teams. This architecture includes, from top to down, mission planning, behavior planning, motion planning, and finally control of the vehicle. The acquisition of sensor data and fusion of these data serve for all of the parts of the architecture (Figure 2.1).

2.2.3. Grand Challenge 2007: Urban Challenge

With the success in the Grand Challenges, DARPA organized a third event: the Urban Challenge. This time the autonomous vehicles were expected to drive 97 km in

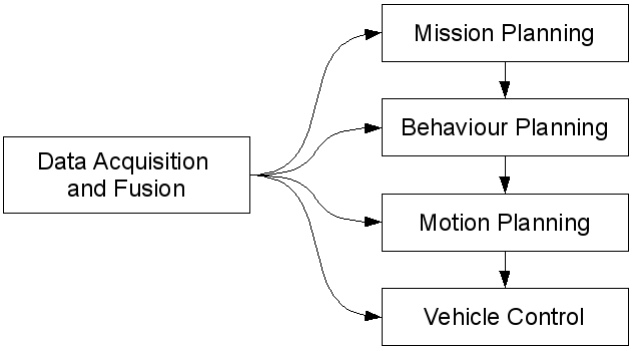


Figure 2.1. Standard autonomous robot architecture

an urban environment (an old air force base), interacting with other vehicles, obeying the traffic regulations, and they were due to complete the course under 6 hours. This challenge attracted much more interest from around the world, with 89 registered teams [4].

The urbanization of the races required much more developed sensing and intelligence. The infrastructure of the sensory hardware was arranged to cover a greater volume of the environment, and also a more complex intelligence was added compared to the previous races. In contrast to the desert conditions where the robots operated in an isolated environment, in the urban areas it is more probable for an autonomous vehicle to collide with stationary obstacles, other vehicles, or in the worst case the pedestrians, but despite the high level intelligence requirement of the Urban Challenge race, they could not interact with the traffic lights, and they operated poorly around pedestrians. It is still believed that there is much more work ahead before the self-driving cars technology becomes commercialized [44].

The winner of the race was the Tartan Racing Team, with their multi-model approach to the DARPA Urban Challenge on their vehicle, Boss [45]. The autonomous operation theory, which we have mentioned as the standard architecture of an autonomous robot at the beginning, was also adopted by the winner team in the Urban Challenge. They implemented this theory with the modules, from top to down, mission planning, behavior generation, motion planning and mechatronics. The perception and

world modeling module is stated to be an information supplier to all of these modules. While the module names may vary in other teams' designs, the followed ways to solve the problems are similar, with an alike top-down architecture.

From the 89 teams, only 53 were given the opportunity to demonstrate their ability in the scenarios consisting of passing stationary cars and interacting appropriately at intersections. Further narrowing the field with 36 vehicles, a qualification event was held that required the robots to merge into, turn across dense moving traffic, find their ways in the road map while avoiding other obstacles, behave correctly in the traffic at a four-way intersection, and finally demonstrate finding a way around an unexpected blockage [46]. Only 11 vehicles passing successfully these challenges would qualify for the Urban Challenge Final Event (UCFE).

The final event consisted of three missions defined for each team by the mission definition file. This mission-based structure required also to pass some checkpoints throughout the courses. This forced the robots to have a mission planner that computes the possible routes to the next checkpoint. For Boss, this mission planner would find the optimal path as an analogy to a human who would plan a route from his or her current position to a grocery store or a gas station, based on the knowledge of blockages, speed limit, and nominal time to make maneuvers [46]. The robots were to generate a graph from the road network definition file (RNDF) in the first place and then solve the optimality problem accordingly and replan in case of unexpected situations. While an RNDF graph extraction is an essential work, the contents of the mission planner varies by the teams. For example, the Stanford Racing Team used a less detailed, but a priori a similar global road planner using dynamic programming [47]. The team AnnieWay developed a rule-based planner for the mission after a series of data fusion, data mapping, and behavior analysis [48].

Every Urban Challenge vehicle has somehow a module for behavioral planner that formulates a problem definition for the motion planner to solve based on the strategic information provided by the mission planner. The behavioral system can be thought as an action selection problem solver. This system is responsible for obeying the rules of

the road, producing a behavior profile, and defining the general motion of the vehicle in both roads (lane driving and intersection handling) and zones (zone positioning) [49].

The motion planner's primary objective is to avoid static or dynamic obstacles while achieving desired goals, especially for structured driving. For unstructured driving, even though there may be no obstacles around, the vehicle has to adjust its position, orientation and direction to achieve a parking position which is a checkpoint in the race. In the work of Urmson et al. [46], the motion planner behaves differently from the behavior planner. While traveling on roads the desired lane implicitly provides a path for the vehicle (the centerline of the lane), however, there are no lanes in the zone navigation. Thus while driving on roads the trajectories are evaluated against their proximity to static and dynamic obstacles, their distances from the centerline path, and their smoothness. On the contrary, for the zone navigation the Tartan Racing developed a lattice search space of position, orientation, and speed, and solved the problem by searching the optimal trajectory using an algorithm called Anytime D* [46]. Like the Tartan Racing Team, the Stanford Racing regarded the zone navigation as a more complex problem, and generated the trajectories using Rapidly Expanding Random Tree (RRT) algorithm, where the trajectories are generated using the physical vehicle model [50]. It is worth mentioning that it is the motion planner that handles the intersection reasoning when the vehicle stops at the intersection (tactical planner in the paper [51]). Each maneuver is initially checked with simple decision rules to determine whether it has a high probability of resulting in a collision or violating the rules of the road. If a maneuver fails this first test, a more in-depth Monte Carlo analysis is performed and the possible future behaviors are sampled according to a Dynamic Bayesian Network (DBN) [51].

In the Grand Challenge 2005, the restrictions on the vehicle types only contained the sizes and weights. The vehicles should not be heavier than 20 tons, larger than 10 feet, and higher than nine meters. The operation of the vehicle should conform to any regulations or restrictions imposed by the applicable land-use authority, so that the vehicles must be propelled and steered principally by traction of the ground and they must not damage the environment or infrastructures at the qualifacation site

[52]. Although there seemed to be some restrictions on the vehicle type, the teams had the chance to make selections in a very large spectrum of vehicles, but in the Urban Challenge, the pool of vehicle types were reduced to around midsized commecial automobiles. All terrain vehicles (ATVs) [53, 54] and golf cart-type vehicles that were used in the previous races were not allowed in the UC. This time the restrictions included the minimum regulations, such as a minimum wheel base of 72 inches (1.82 meters), a minimum weight of 2000 lbs (907 kg). The vehicles should also be able to make a U-turn in a limited area [55].

Considering these regulations and rules, the majority of the teams in the UC chose large urban automobile style vehicles, like mid-size sport utility vehicles (SUVs) [45, 49, 51, 56, 57], large station wagons [47, 48] or hatchbacks [58]. The common prominent features of the selected vehicles are their heights that enables to cover a larger area with sensors mounted on the roof and their hybrid structures that can easily be modified to be driven-by-wire.

Compared to the previous Grand Challenges, there are great developments and investments on sensor selection and sensor placement issues. While the coverage of the sensors was generally expanded, especially for the backwards of the vehicle which might have been discarded on the desert, we see that the teams chose more expensive infrastructures in terms of cost and complexity.

The fundamental functions that are expected to be performed by the vehicles' sensory infrastructure can be summarized as data fusion, and tracking of moving obstacles, detection of static obstacles, road shape estimation and detection of road shape features. While online learning and filtering methods are highly used to accomplish these functions, offline methods, like Haar Wavelet, were also implemented for the road shape estimation and the road shape features detection tasks [45]. The range scanner sensors vary by their field of views, angular resolutions, and range. Some superior sensors have the ability to cover the whole area around the vehicle with the different vertical level scanning capability but with low ranges. Some of them have a range of 300 m but only a limited resolution and coverage. To gather the maximum amount

of information about the surrounding, these different types were placed around the vehicle [51].

The main interest was on the active range scanning sensors again, but this time the usage of video sensors was inevitable, because of the necessity to detect road lines and signs [51]. As an example, in [48], they used two stereo camera systems to obtain valid depth information where they implemented a stereo self-calibration algorithm called a robust Iterated Extended Kalman Filter to allow continuous update of camera parameters without having to store all the previous images. While the other vehicles might also be detected by camera systems [49], generally a segmentation in the point cloud supplied by the range scanners was used by the teams [48]. For the road lane detection, the team Ben Franklin made a Laser Detection and Ranging (LADAR) Reflectivity Analysis and used the LADAR to detect the lines depending on the different reflection characteristics of different colors on the asphalt [59].

The teams' confidence in the various characteristics of different sensors stands out. Some teams approach the problem as a software engineering problem [57], since the individual hardware components are available off the shelf. They used only three range scanning sensors and tried to solve the problems using their advanced software framework.

As an exception to the expensive sensory systems, the Team MIT chose several inexpensive sensors mounted on the vehicle periphery. The calibration of these sensors were made with a new cross-model calibration technique [56]. They believed that cheap sensors with correct placements would replace more expensive fewer sensor kits. This model would provide redundancy in covering the surrounding and would also prevent a single point of failure situation caused by the malfunction of a single expensive sensor.

2.3. Distance Extraction From Textural Information and Obstacle Detection

A visually guided autonomous robot needs to sense some marks that will supply it depth information to safely navigate in its environment. For indoor environments, there are hand made structures, such as wall endings, corners, doors etc, that can be used as landmarks. Similarly for the roads, these landmarks are the lines or the borders of the road, but for the unstructured terrains, the robot has to find its own landmarks to acquire safety information from the environment. Moreover, when the monocular vision is considered ordinary computer vision techniques would not be sufficient to extract this information reliably since the obstacles in outdoor environments do not follow a general pattern. Furthermore, the changing light conditions would badly affect the learned patterns [60].

Considering these constraints of the unstructured outdoor environments, we require another method to extract depth information in the images involving some meta information from the outdoor images, rather than searching single patterns in the images. The considered methods are the texture analysis techniques [7].

2.3.1. Textures and Texture Energies

2.3.1.1. Texture. As we will be concerned with texture analysis in the following chapters, we need to define what texture is. In the general concept, the word texture is regarded as an ensemble of neighboring objects sharing some common features. These features may be differentially common, that is, having large differences between high and low points such as rough or bumpy for the touch sense, or proportionally common, that is, having little differences between high and low points such as silky [61]. In computer graphics, texture is defined as an image applied regularly repeatedly to a polygon to create the appearance of a surface, or basically as a colour map [62].

In the computer vision domain these features that determine a texture correspond to neighboring pixels where the highs and lows are brightness values or gray

levels instead of elevation changes. It is easy to understand what a texture is, but not as easy to define it. A more regular definition of the texture is the variation in intensity of a particular surface or region of an image. This statement is still ambigious about what is described by the physical object observed or the image derived from it. Gathering these statements, we attain that it is the roughness of the surface or the structure or composition of the material that gives rise to visual properties [61]. Thus texture analysis basically aims to determine the boundaries between different textures, which often signify the boundaries of real objects. Under this assumption, statistically analyzing the textures of 2D images will give us computable information on real world correspondences.

The definition implies that texture cannot exist in a pattern of uniform intensity. A texture does not say anything about how the intensity might be expected to vary or how we might recognize or describe it. It may vary in numerous ways if the variation does not have sufficient uniformity, the texture may not be characterized sufficiently to permit recognition [61].

A textural pattern is not all random or not all regular. There exist degrees of randomness and of regularity that have to be measured and compared when characterizing a texture. This regularity or randomness emerges from the fact that these images are derived from tiny objects or components that are themselves similar, such as grass or terrain but their formation is random or regular.

Texture analysis comprises the analysis of the textural elements (texels) similar in some way - size, degrees of uniformity, orientation, distance or direction variations, magnitude, background content, contrast - that clearly describe the various degrees of regularity or randomness. While the texel classification is an important sub-domain of texture analysis, we won't be very concerned in it. The notion we will be using is the texture energy.

2.3.1.2. Texture Energy Measures. Laws' approach to textures has led to other developments that provide a systematic, adaptive means of conducting texture analysis [61]. He presented a novel texture energy approach, using basic filters that were the common Gaussian, edge detector, and Laplacian filters. These filters were designed to highlight points of high texture energy in the image. He identified some properties as playing an important role in describing texture: uniformity, density, coarseness, roughness, regularity, linearity, directionality, direction, frequency, and phase [63].

The Laws' masks are derived from the following basic 1x3 masks, by convolving them among themselves,

$$L_3 = [\ 1 \quad 2 \quad 1\ ] \tag{2.1}$$

$$E_3 = [\ -1 \quad 0 \quad 1\ ] \tag{2.2}$$

$$S_3 = [\ -1 \quad 2 \quad -1\ ] \tag{2.3}$$

The initial letters indicate *Local averaging*, *Edge Detection* and *Spot Detection* respectively. By convolving these three masks in pairs, convolving a vertical 1D kernel with a horizontal 1D kernel, we obtain a complete set of 3x3 masks (Table 2.1) [64]:

Table 2.1. The nine 3x3 Laws masks

$L3^T L3$		
1	2	1
2	4	2
1	2	1

$L3^T E3$		
-1	0	1
-2	0	2
-1	0	1

$L3^T S3$		
-1	2	-1
-2	4	-2
-1	2	-1

$E3^T L3$		
-1	-2	-1
0	0	0
1	2	1

$E3^T E3$		
1	0	-1
0	0	0
-1	0	1

$E3^T S3$		
1	-2	1
0	0	0
-1	2	-1

$S3^T L3$		
-1	-2	-1
2	4	2
-1	-2	-1

$S3^T E3$		
1	0	-1
-2	0	2
1	0	-1

$S3^T S3$		
1	-2	1
-2	4	-2
1	-2	1

The set includes one mask ($L3^T L3$) whose components do not average to zero. This is regarded as less useful for texture analysis since it gives results dependent more on image intensity than on texture. The remaining masks are sensitive to edge points, spots, lines and their combinations [61].

To find the local magnitudes of these quantities, (edginess, spots, linearity etc), they smooth these masks over the original 2-D intensity image regions that are greater than the mask sizes, and a vector image is created with components representing energies of different types, to estimate texture energy. Originally, in [65], Laws used both squared magnitudes and absolute magnitudes, and the squared magnitude corresponds to true energy and gives better response, but absolute magnitudes (Equation 2.4) require less computation [61].

$$E(l,m) = \sum_{i=l-p}^{l+p} \sum_{j=m-p}^{m+p} |F(i,j)| \tag{2.4}$$

The feature vector computed from the original image by applying Laws' Masks was developed in [66], by using higher sized edge detectors. The texture information is mostly contained in the image intensity channel (the Y channel of a YCbCr image), and the Laws' masks are applied onto this channel and to capture the haze reflected in the low frequency information in the color channels, the averaging mask ($L3^T L3$) was applied on the color channels (Cb and Cr channels). The texture gradient was finally computed by applying the newly constructed edge detectors on the intensity channel. The energy was calculated in [66] as previously.

2.3.2. Distance Extraction and Obstacle Detection

For depth estimation generally range sensors and/or stereo vision are used. In computer vision, obtaining 3D depth from images is a basic problem. Most work on this

problem has focused on stereo vision, structure from motion, or depth from defocusing.

Monocular vision is considered to be obviously less costly in economical means than using range sensors or complex camera systems, but it is also a difficult task. A group from the Stanford University has developed a depth learning algorithm and driven a remote control car at high speeds through unstructured outdoor environments using monocular vision [7]. To achieve this, several thousands of outdoor images were captured and correlated with laser scanner data, which provided distances to the nearest obstacles at a resolution of 0.01 radians. These images were divided into vertical stripes that correspond to different steering directions and each stripe was labeled with log-distance to the nearest obstacle. Moreover, overlapping windows were taken into account to calculate some coefficients representing texture energies and texture gradients, that is, relating the stripes with their neighborhoods. To calculate these energies and gradients, Laws' masks [64] were applied to intensity and color channel images of the original image. Finally training was performed by using standard linear regression, taking the feature vector for each stripe as input and the nearest obstacle distance as output.

The obstacle information supplied by the algorithm was used as input to the reinforcement learning based control of the vehicle. They modeled how aggressively the steering is performed by reinforcement learning, speed of the car as gains and crashes as penalties. The tests run with different obstacle densities and types, and different terrain types. The average speed of the vehicles was around 5 m/s and they claim to have driven the car more than 100 meters without crashing into any obstacle.

The same group improved their depth learning algorithm using monocular vision [66]. In their work, they consider a supervised learning approach to estimate 3D depth from monocular images. For the training purposes, a 3D range scanner is used to collect data. With the real values of the distances known, the corresponding image is divided into small patches, each of which is labeled with a single depth value. For these labeled images, feature vectors are built using absolute depth and relative depth features (the distance measured by the scanner and magnitude of the difference in depth between

two patches respectively), and the features that contain the three types of local cues: texture variations, texture gradients, and haze, using various techniques for texture energy computation and edge filters within Laws' masks. Finally a 646 dimensional feature vector for each patch containing both local and global features is formed.

Additionally they formed a relation between the patches, assuming that the depth information of two patches of an image would be highly correlated. They modeled a relation between the depth of a patch and the depths of its neighboring patches. These models are two probabilistic models, a jointly Gaussian Markov Random Field (MRF) and Laplacian MRF. Testing their models on real world test set images they claim to have around 10 - 15 per cent of errors in forest, campus, and indoor environments.

There are also applications of optical flow used for obstacle detection. In one such application [67], the flow at the edge of the image is extracted using the virtual intensity gradient along the edge. By the proposed algorithm, they claim to have reconstructed the 3D obstacle information from the data that a mono-camera system supplies.

2.4. Obstacle Avoidance

All mobile robots require some kind of collision avoidance system, ranging from primitive algorithms that detect an obstacle and stop the mobile vehicle in order to avoid a collision to complex algorithms that detect an obstacle and, for example, make the robot go around that obstacle. The primitive ones that react to sensor data are also called reflexive algorithms and as the algorithmic complexity increases the algorithms tend to make plans on the paths.

Reflex control is a fast, on-line and fail-safe avoidance method. Additionally, the configuration space is limited and the robot makes decisions in this small space. While the controller does not improve commands that would result in collision, that is, the robot makes strict movements to stay in the limited world even under high speed motions [68], the robot is unable to solve problems efficiently under more complex distribution of obstacles.

Besides the purely reflexive obstacle avoidance approaches such as the ones in [26] and [68], there exist simple but efficient obstacle avoidance techniques. One of the most cited approach for obstacle avoidance was suggested by Khatib in [69]. The proposed concept, "artificial potential field" was adopted by many researchers on not only manipulators but also on mobile robots in dynamic environments. The focus of the research, as stated in the article, is developing a collision-free path for the agent, accomplishing its main task free from any risk of collision. The algorithm mainly depends on the operational space formulation, which describe mathematically the localization of the end effector of the manipulator or of a mobile robot. The artificial potential field describes an agent that moves in a field of forces. The position to be reached is an attractive pole and obstacles constitute the repulsive surfaces. The path is planned accordingly.

Another similar method is called "elastic band theory" [70]. An elastic band is a deformable collision-free path. First a path-planner provides a continuous path in which the robot will not crash into any obstacles, but in real world, because of incomplete and inaccurate information, the collisions may still occur at local changes. The theory brings a control over this previously planned path by local noises, in other words, the trajectory that the robot will follow is deformed like an elastic band is pulled by some forces at sides without losing continuity. The modifications on the trajectory are done according to sensory data in real-time, obtaining a collision-free, shortest, smooth trajectory for the robot at the end. By some critical modifications, the method may be also considered as an emergency run-away from the previously determined path [71]. Another application of the same method adopting the potential field method is [72], where the mobile robot steers towards the target, while simultaneously detecting unknown obstacles and avoiding collisions using histogram grid based world model and updating this model according to range data via onboard range sensors.

Having learned the road information, obstacle information and the vehicle's localization, the vehicle should be directed in the desired direction with the desired speed. Because of the high nonlinear characterization of the AGVs caused by extraneous forces, winds, friction, harsh outdoor environments and alike, a control strategy must

be able to deal with them. The method proposed by Kodagoda et al. [73] deals with this problem using fuzzy (mamdani inference strategy) control, proposing a control structure over the vehicle parameters to stabilize vehicle speed and steering angle.

3. TESTBED ROBOT DESIGN

3.1. Problem Definition and Constraints

The tests of the depth estimation algorithm suggested in Chapter 4 requires a mobile robot that is able to navigate autonomously in the environment it is trained for. The tests are performed in outdoor environments. So the robot also has to be durable to rough terrain. Thus, the problem can be briefly defined as driving a robot vehicle autonomously in outdoor environments.

In order to solve the autonomy problem, before all else, we have to have a reliable robot with its actuators controllable from an onboard computer and with sensors, its values of which can be read from that onboard computer that serves as the main controller. The problem of construction of a robotic agent can be discussed under various sub-problems, such as data acquisition from the sensors, motor driving interface, processing power and electrical power provision. These sub-problems are divided into two main sections as hardware and software components, and each of these problems are explained, and the proposed solutions and the implemented techniques are given in the corresponding sections.

3.2. Hardware Components

The robot hardware mainly consists of two parts: the chassis with the actuators and the electronic components. The chassis of the robot is the mobile part of a radio controlled toy car system already equipped with a steering servo motor and a DC motor which form the actuators of the robot. The other electrical and electronic components attached to the vehicle are a processing unit, a webcam, a laser range scanner, a bluetooth adapter, a motor control interface circuit, and the battery packs. The placement of the components of the robot is given in Figure 3.3. The packed vehicle can be viewed in Figure 3.1 and the hardware architecture is illustrated in Figure 3.2.

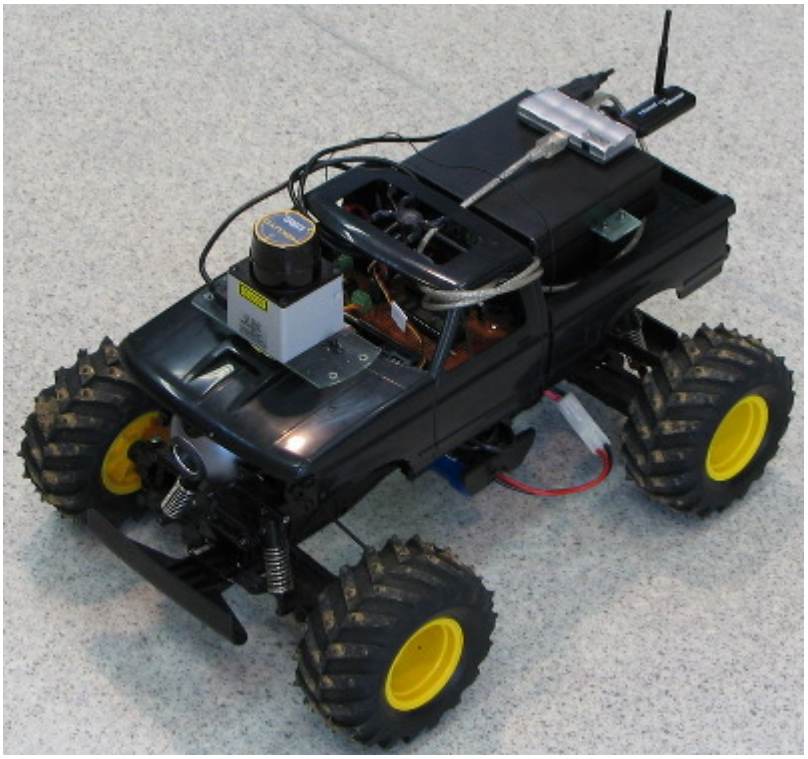


Figure 3.1. The robot with all components mounted and connected

3.2.1. The Robot and Motor Control Circuit

3.2.1.1. The Chassis. We did not construct the chassis by ourselves. From the available commercial alternatives, an electric powered R/C offroad truck, a Tamiya Blackfoot Extreme (Figure 3.1) from Tamiya Inc was chosen. While the vehicle comes normally unassembled as a kit, a preassembled vehicle was used as the chassis of the robot vehicle, thanks to Abdullah Akçe, who had already assembled this vehicle in his senior project [74]. The technical specification of the vehicle is given in Table 3.1 [75].

Utilization of such a vehicle was very advantageous. Since it is an electric car, which is clean compared to a vehicle with a gas engine, the majority of the tests involving the control of the vehicle could be made in the laboratory environment. Also, by the structure of its polycarbonate skeleton and monster truck wheels, the vehicle was tested to be durable to most inappropriate outdoor environments containing rocks and grass.

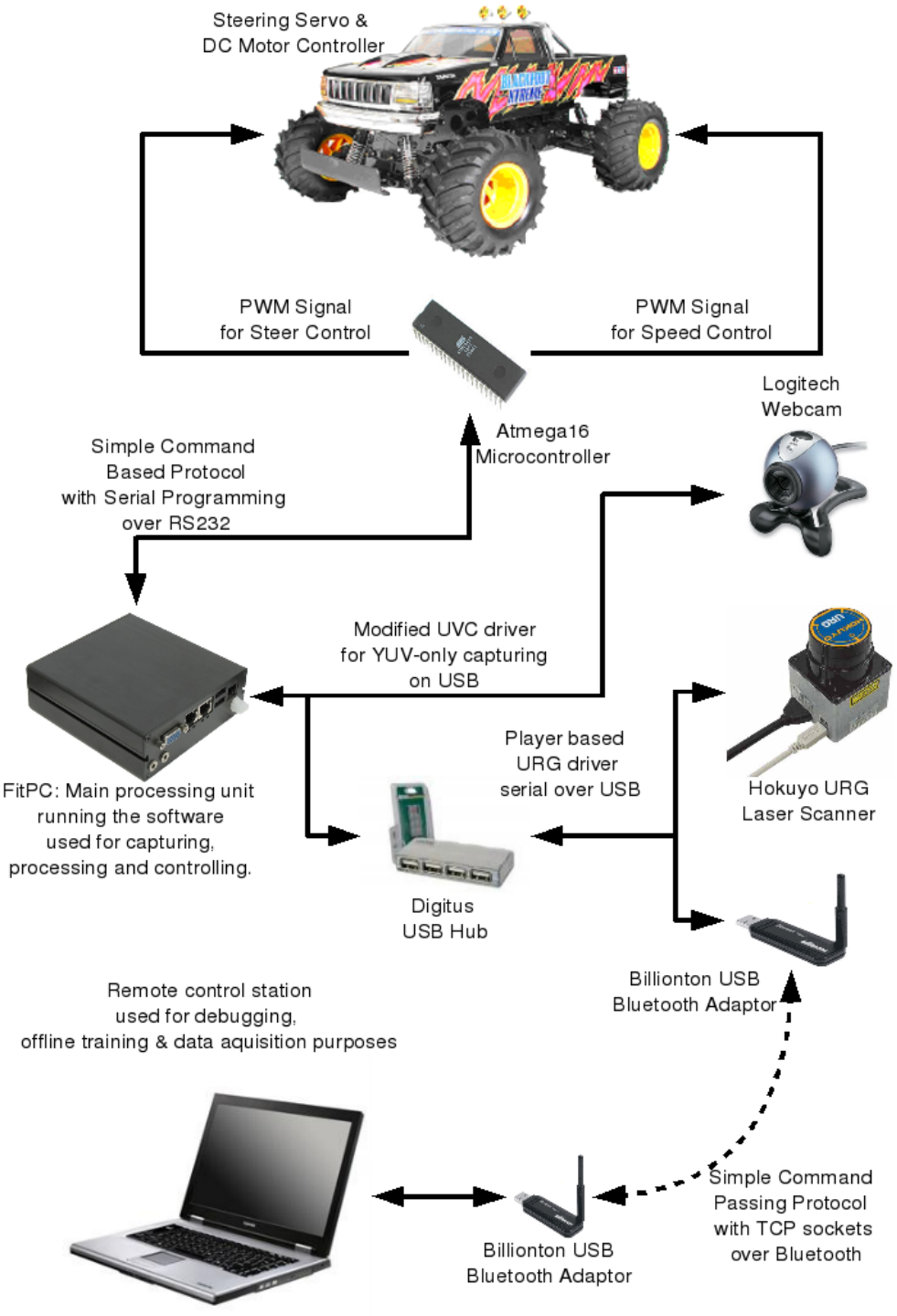


Figure 3.2. Hardware architecture of the robot vehicle with the protocols linking each component

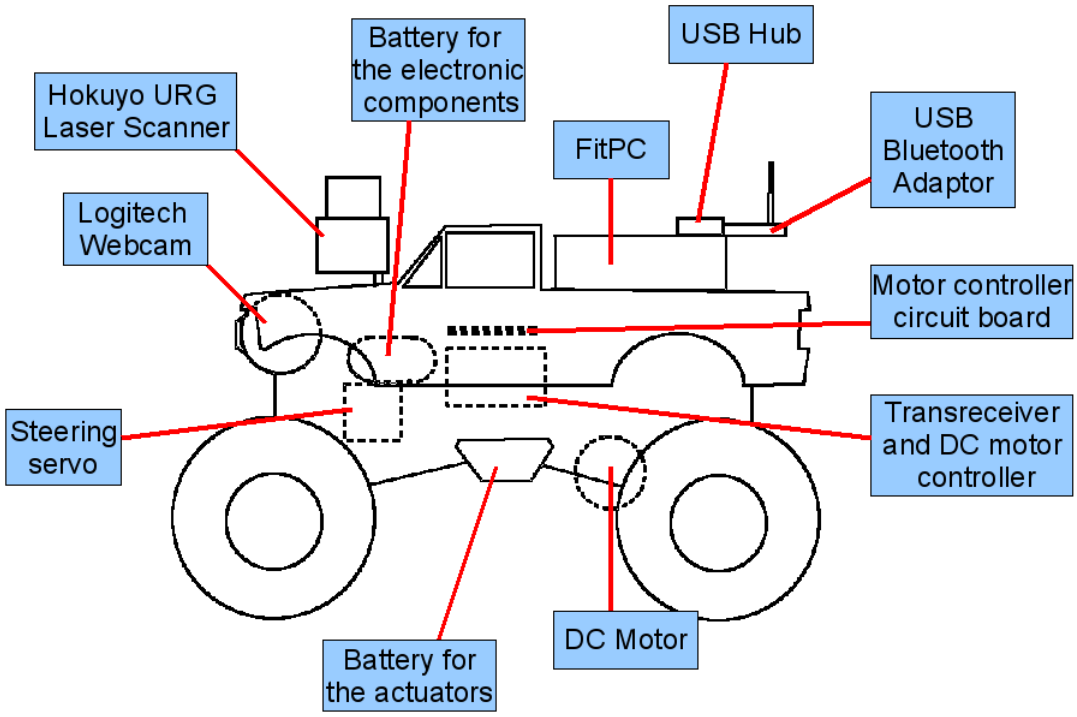


Figure 3.3. Placement of the components on the vehicle

Table 3.1. Technical specification of Tamiya Blackfoot Extreme

model number	58312
model name	Blackfoot Extreme
scale	1/10 Monster Truck
body	ABS molded hard shell, chromeplated acessories
chassis desc.	Polycarbonate monocoque tub
suspension	Four wheel double wishbone
motor	RS540SH
drive	2WD rear
width	310mm
length	445mm
height	267mm
wheel base	246mm
minimum ground clearance	47mm
tires	130mm, all terrain lugged

3.2.1.2. The Motors. The vehicle motors are the two actuators that can be controlled by the user given appropriate signals. One of the two motors is a DC motor that is attached to the rear wheels and that lets the vehicle move in forward and backward directions. On the chassis, the vehicle has also a DC motor controller, that transforms the voltage levels to Pulse Width Modulation (PWM) signals. The other motor is a servo motor that makes the front wheels turn left and right functioning as a steering controller which is also driven by PWM signals. Thus, the steering servo and the DC motor controller are both driven by PWM signals.

3.2.1.3. Motor Controller Circuit. PWM of a signal involves the modulation of the duty cycle to convey information over a channel or to control the amount of power sent to a load [76]. The modulated signal uses a square wave whose pulse width, that is the high level voltage in our case, is modulated in the average value of the waveform.

In the original configuration, when the vehicle is driven by the remote control handset, the handset's transmitter sends radio signals to the transreceiver of the vehicle. The transreceiver generates the corresponding PWM signals and sends them to the steering servo and the DC motor controller over the three colored cables (red: VCC, black: ground, yellow: PWM signal). What is needed to drive the motors by other

means is to intercept and modify this system and generate our own PWM signals identical to the ones that transreceiver produces to drive the motors [75].

Figure 3.4. AVR Atmel ATMEGA16 Microcontroller

Since a personal computer (PC) is not capable of directly generating PWM signals on its own, we had to place another interfacing processor that would generate the PWM signals and that could also communicate with the main processing unit. The chosen processor was an AVR Atmel ATMega16 Microcontroller (Figure 3.4), an 8-bit microcontroller with 16K bytes in-system programmable flash, which is capable of producing four different PWM outputs at the same time. The Atmega16 is also Universal Asynchronous Receiver/Transmitter (UART) compatible having an Universal Synchronous and Asynchronous Serial Receiver and Transmitter (USART) interface that translates data between parallel and serial forms. This makes it possible for the microcontroller to communicate with any USART capable device over serial line [77].

Using an oscilloscope the modulation pulse width the vehicle's transreceiver generates was observed (Figure 3.5). It was also visualized that the modulated signal's width would vary between 0.5 and 0.8 milliseconds with a duty cycle (period) of 6.4 milliseconds. The microcontroller was then programmed as to be able to generate modulation signals between these width values.

The microcontroller was also made to communicate with a PC over a serial line. We used a MAX232 integrated circuit (IC) to convert TTL level serial signals to RS232 signals for them to be handled by a PC. The whole circuit power is supplied over an IC 78N05, a 5 volt regulator durable up to two amperes of current with a heatsink, used for

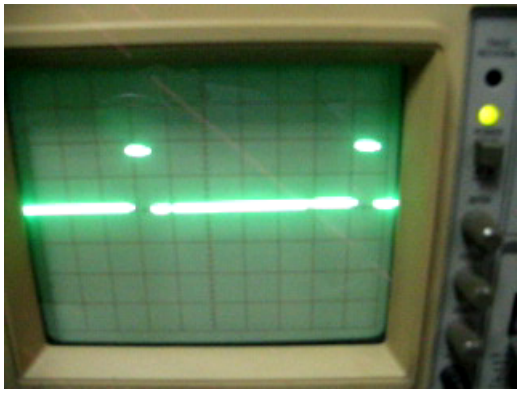


Figure 3.5. PWM signal on oscilloscope

regulating the battery voltage of 7.2 volts to five volts. Not only the microcontroller or the integrated circuits require a 5 volt power supply, but also the onboard processing unit (FitPC) and the laser range scanner (URG) run on 5 volts, so the problem of power requirement of more than one device was solved at a time. The schematic of the custom circuit designed for the vehicle control is given in Appendix A and the circuit board itself can be viewed in Figure 3.6.

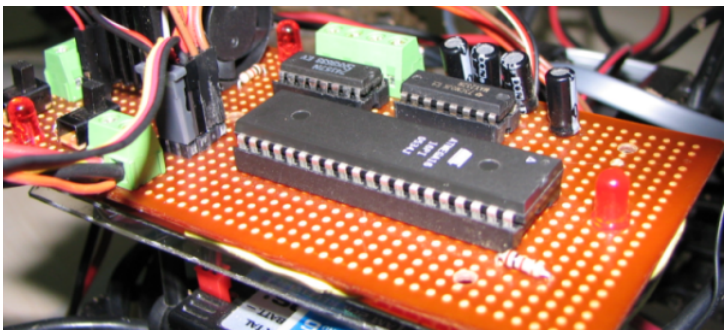


Figure 3.6. The motor controller circuit, mounted in the middle of the car on the transreceiver of the remote control

3.2.2. Sensors

Two sensors are attached to our robotic vehicle: a webcam and a laser range finder. Both of the sensors are fully functional with their drivers ready for the operating system.

3.2.2.1. Camera. The camera, mounted to the front of the vehicle, is a Logitech QuickCam Pro 5000 Webcam (Figure 3.7) that connects to a PC over USB. It is capable of retrieving images at 30 fps and with resolutions ranging from 160x120 up to 640x480 pixels [78]. The specifications of the camera states that the camera is only compatible with the MS Windows XP and later operating systems, but there is a project to supply Linux driver for the USB Video Class (UVC) family webcams [79], in which Logitech QuickCam Pro 5000 takes place. We made the operating system recognize the webcam using this driver [80]. Moreover, it was tested to grab video streams using the software project "luvcview (SDL Usb Video Class grabber)" [81]. The "luvcview" source code was largely modified to meet the vehicle's needs - grab and write images at 30 fps with a resolution of 320x240 and the resulting functions was merged with the vehicle software, depending on the General Public License (GPL) the program comes with.

Figure 3.7. Logitech QuickCam Pro 5000 Webcam

3.2.2.2. Laser Range Scanner. The second sensor attached to the vehicle is a Hokuyo URG-04LX Laser Range Scanner (Figure 3.8) [82]. The range scanner has two options to connect to a PC. As well as it may be connected through a serial line, the USB connectivity option which is in fact a serial to USB interface, was chosen for there were no more serial ports on the FitPC and this solution saved from extra wiring. The sensor is mounted on the hood, the most convenient place to be fixed to make it close to the camera. Furthermore, the sensor is directly attached to the bars of the chassis that support the hood and the sensor is reinforced by attaching it to the hood.

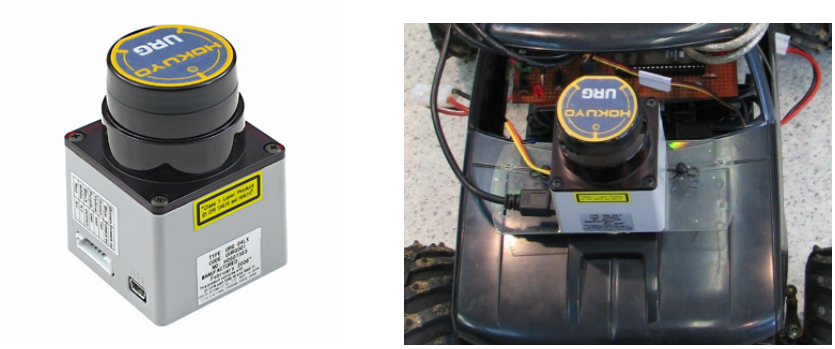
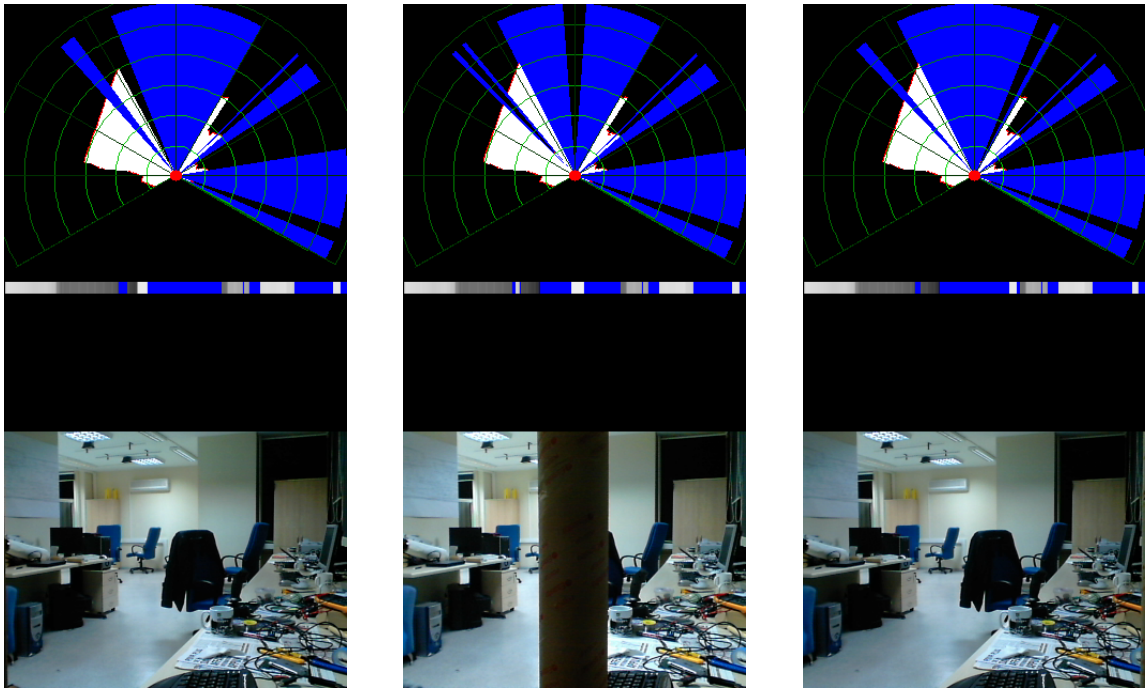


Figure 3.8. Hokuyo URG-04LX Laser Range Scanner

The driver used for the sensor is the Universal Serial Bus Communication Device Class Abstract Control Model (USB CDC ACM) driver that exposes the USB device as a virtual serial port to the operating system and that comes as a part of the Linux Kernel (version 2.6.26) [83]. Finally, over the device driver, a grabber interface based on the related driver of the Player Project [84] was implemented and we managed to collect laser scanner data at 10 fps with a resolution of 0.35 degrees and with a total field of view of 240 degrees of the front of the vehicle.

3.2.2.3. Camera and Laser Scanner Data Match. Two correlation problems emerge with the sensor outputs: timing and orientation. The timing problem is finding a temporal relation between the video image files and the laser scanner data files, that is to say, finding the matching two files of different types. This problem is a dynamic problem that depends on the vehicle's speed at the time the files are created. A proposed solution is presented as the matcher program in control station in Section 3.3.5.2.

The orientation problem is a calibration problem. It is related to finding the angle of the laser range scanner that corresponds to the camera's field of view. This problem was solved by fine tuning the laser scanner's position after mounting the camera on the bumper. The fine tuning was performed via the visualization of the both of the sensor outputs simultaneously. The camera's center is stated about 60 millimeters farther and 100 millimeters lower than the range scanner's and their positions have



(a) The bar is on the left edge of the camera view

(b) The bar is in the middle of the camera view

(c) The bar is on the right edge of the camera view

Figure 3.9. Calibration of the laser scanner orientation

no lateral difference. Assuming that these distances are negligible by the orientation measurement, a cylindrical bar was set, assured to be standing straight up in front of the camera. Finely measuring and restationing the bar to the midpoint of the camera, the laser scanner view was visualized. The laser range scanner was repositioned about an orientation that the bar's view in the scanner view was stationed exactly in the middle of the scanner's field of view (Figure 3.9(b)). The laser range scanner was mounted with the corresponding orientation. An extra effort was made to measure the angular range of the camera's field of view by this technique. The same bar was placed at the edges of the camera's sight, and finally the camera's field of view was measured as 46 degrees total, equally divided on each side (Figure 3.9(a) and 3.9(c)).

3.2.3. Processing Power and Data Capacity

By definition, the autonomy in various domains such as psychology, economics, mathematical analysis, or robotics, is directly related to the concept of independence.

In robotics, especially in the context of the autonomous mobile robots, autonomy means independence from external control [1]. The autonomy is a matter of degree, where no total autonomy does not exist. The autonomy of an agent depends on how much resource this agent can acquire from the world. In our case, this resource is the information related to the environment that the robot resides in. It has to acquire this information on its own and interact with the environment via its own decision capability. These requirements led us to an on-board processing unit of the robot vehicle. These requirements explain well why an on-board processing power is needed on the vehicle. Also, for faster critical decisions, to avoid large data transfer problems that may occur between the vehicle and a central processing unit and the increase in the point of failures, an onboard robotic processing unit is a must to have.

Much consideration was given to the processing power of the vehicle. The main problems were that the processing hardware had to be small in volume and weight so that the vehicle could carry it easily that the electrical consumption had to be as low as possible, it should offer more than one USB ports and at least one serial interface, it should be durable to continuously rocking feature of the vehicle on the field, and it should offer as much processing power as possible. Considering all these challenges, we reduced the options to a Fujitsu-Siemens Personal Digital Assistant (Figure 3.10(a)), a pack of Gumstix (Figure 3.10(b)), and a FitPC (Figure 3.11) due to their small scale. The specifications are given in Table 3.2.

Table 3.2. Comparison of the specifications of the three processing power candidates

Component	FitPC	Fujitsu-Siemens Loox	Gumstix Package
Processor	AMD Geode LX800 500 MHz	Intel Xscale PXA270 520 Mhz	Marvell Xscale PXA270 416 Mhz
Memory	256 MB of DDR	64 MB of DDR	64 MB of DDR
Data Hold	40 GB IDE Hard disk	128 MB SDCard	16 MB Flash
USB	2.0 - 2 ports	1.1 - 1 port with expansion	2.0 - 1 port
Serial Port	1 port	not available	3 ports
Wireless	not available	802.11 b/g - Bluetooth - IRDA	Bluetooth
Weight	300 g	160 g	$\sim$15 g
Battery	not available	1200 mAh	not available
Opr. Sys.	any x86-based OS	Windows Mobile 5	Linux kernel 2.6

(a) Fujitsu-Siemens P. Loox

(b) Gumstix Verdex

Figure 3.10. Processing unit options

All devices run at close CPU speeds ranging from 400 and 520 mHz. The Gumstix [85] is the most prominent option to be used in robotic applications by the virtue of its smallest scale among others and its expandable structure, but as we are dealing with high volumes of data, its lack of data storage capacity voted negatively for the Gumstix. Moreover, communication emerges as a problem because of the limited connection interface between the Gumstix and another PC. When the same conditions are considered for the Pocket Loox [86], it has an expandable flash card slot which would let much more data to be stored. Furthermore, it has its own battery pack and it is ready to be programmed, however, the Pocket PC is not really aimed for robotic applications, its system cannot be modified or tweaked as desired and its communication ports are insufficient for our sensors.

Figure 3.11. FitPC

The FitPC was selected as the main processing unit of the vehicle because of its convenience. The drawbacks of the FitPC are its higher physical volume and weight compared to other small scale computers, its lack of wireless connectivitiy, and its mechanical hard disk that could get broken during tests. Nonetheless, the FitPC offers two USB Host slots which can also be expanded by a USB Hub (Figure 3.12) where a USB Bluetooth adapter was installed to provide wireless connection. While the USB slot expansion is performed on one USB slot, the other USB slot is used to connect the camera, considering that the densest data flow is on this connection. The hard disk was also replaced by a Compact Flash Card over CF to IDE converter (Figure 3.13) to avoid the vibration limitations of the mechanical disk. Thus all the drawbacks of the FitPC to be a candidate for the processing unit of the robot vehicle are overcome.

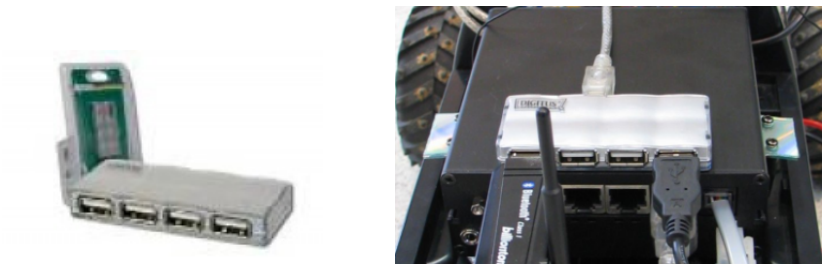


Figure 3.12. Digitus USB Hub

Figure 3.13. CF to IDE Converter with a Compact Flash Card installed

The FitPC originally comes with a Gentoo Linux operating system [87], which is a Linux distribution that can be automatically optimized and easily customized for any application. It depends on the packages that are compiled against the subarchitecture of the processor, which makes it flexible and fast compared to the distributions that uses precompiled packages. To make the system fit in the compact flash card, the package manager related files were left out and finally a system of about 500 megabytes with

all the needed libraries, editors, compiler related files, the vehicle software, and other assistant objects was set up. With this system the compilation of the software can be done onboard. It also permits debugging and sending or receiving files over network and even development of the software by the remote connection capability.

3.2.4. Electrical Requirements

The vehicle is an electric car. Besides, it is equipped with processing units and sensors, which makes the electrical supply the key issue. In Table 3.3 we give the approximate electrical consumption that we have measured in ampers by each component. The total electrical requirement is around 1.7 A. When this amount is added to the current that the motors need, a single battery of three amperehours that comes with the vehicle does not suffice. Furthermore, the motors' consumption is not stable and during initial tests this instability caused insufficient feeding for the electronical components, which also caused lock-ups on the computer.

Table 3.3. Electrical current requirements by the electronical components

Component	Current (A)
the FitPC	0.83
USB Hub + Webcam	0.24
Bluetooth Adapter	0.12
Laser Range Scanner	0.43
The MicroController circuitry	0.05
Total	1.67

To cope with the electricity requirement, another 7.2 volts Ni-Mh 6-cell battery pack of seven amperehours was added to feed the electronical components of the vehicle (the FitPC, the Laser Range Scanner, etc.) (Figure 3.14). The battery pack is directly connected to the circuit board where it is regulated to five volts and where it is distributed to the FitPC, the laser range scanner, the microcontroller and the USB Hub. We have also noticed that mounting another battery pack on the vehicle made it react more stably by increasing the weight. One drawback of adding more weight on the vehicle was the limitation in steering compared to the unequipped state of the

vehicle, but the tests showed that the vehicle could still react well on sharp turning situations. The motors were left to be fed by the vehicle's original 7.2 volt Ni-Mh 6-cell battery pack of three amperehours. The electrical connection schematic can be viewed in Figure 3.15.

Figure 3.14. Battery packs for the robot

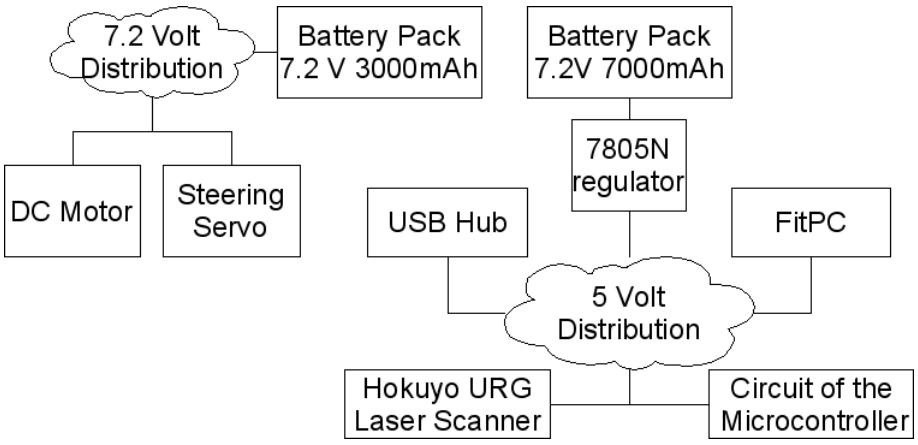


Figure 3.15. Electric distribution on the vehicle

3.2.5. Communication Components

As the constructed robot is a mobile robot, wireless communication is a must to have. In the section on processing units (Section 3.2.3), we compared three small sized units, two having their own wireless communication options, either bluetooth or wireless LAN, however, because the FitPC was selected as the processing unit, an adapter had to be added. While, for the FitPC, the wired ethernet connection is always available and was efficiently used in the process of software development, we added a Billionton USB Bluetooth Adapter (Figure 3.2.5), whose specifications state that the range of the adapter is 100 meters. We tested a successful connectivity up to 80 meters.

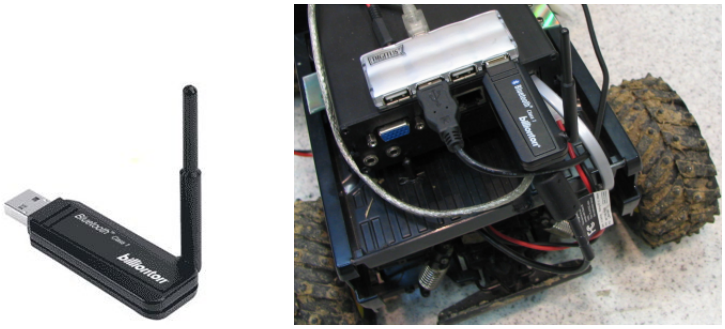


Figure 3.16. Billionton Bluetooth USB Adapter

The connection between the FitPC and a faraway control station is an adhoc connection between two bluetooth adapters, one for each machine. Communication is established over IP network using the Bluetooth Network Encapsulation Protocol (BNEP) [88]. The FitPC boots into a listening mode of the bluetooth service. When a request is received on bluetooth, the bnep module awakens and establishes the connection, specifying a static IP address for the device.

3.3. Software Components

The vehicle software has three mandatory jobs: data acquisition, data processing and vehicle control. Having succesfully implemented these main jobs, extra utilities were added for debugging purposes: data dumping, data viewing, manual control, and reporting. These debug related jobs are also expanded in the control station with various assistive programs.

The FitPC software is structured in multiple threads to maintain the syncronization between these multiple jobs. The communication between different threads is controlled by the mutual exclusion (MUTEX) algorithms to avoid simultaneous use of the global memory segments. The thread architecture is given in Figure 3.17.

All of the software has been written in C/C++, using various editors like *kwrite*, *mcedit* and *kdevelop*. To maintain current and historical versions of source files of the project software we use *Subversion*, an open source version control system [89].

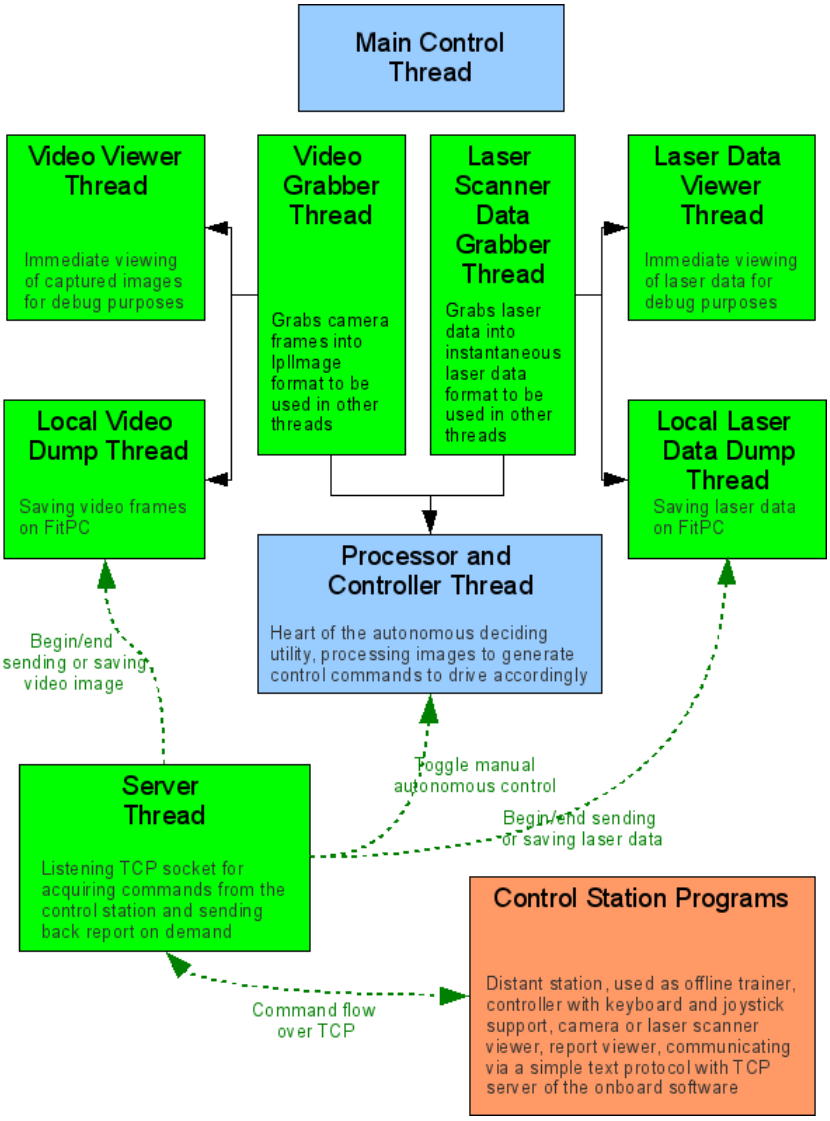


Figure 3.17. Software architecture

3.3.1. Actuator Control Software of the Microcontroller

We stated that the motors were driven by PWM signals. Having found the pulse width values of the required signals in Section 3.2.1.3, the microcontroller was programmed to generate the corresponding signals on the two PWM output pins. The calibration of the signals was achieved by setting the related pins' I/O flags. At the final state, the signals have 80 and 50 controllable levels of width. The pin with more precise resolution (80 levels) is connected to the DC motor controller with 40 levels of forward and 40 levels of backward motion, and the other pin is connected to the

steering servo which controls the amount of wheel turn in 25 levels to left and 25 levels to right.

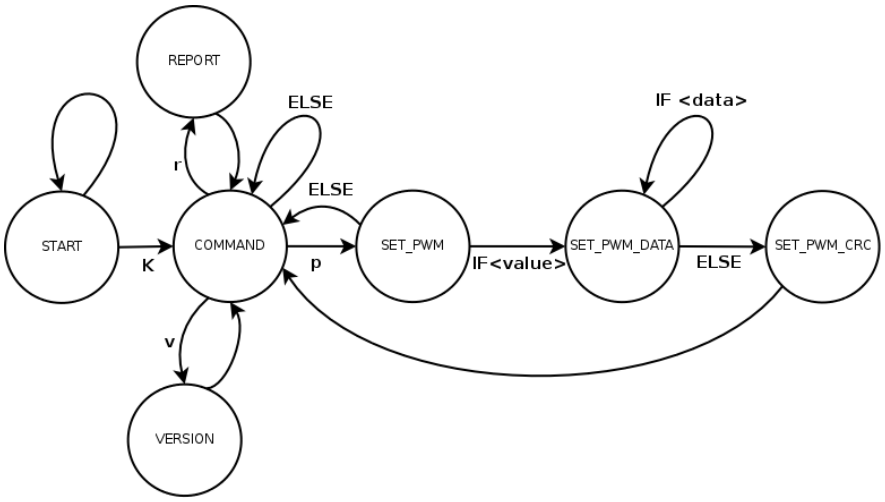


Figure 3.18. Processing of a command on the microcontroller

The software of the microcontroller has also the UART routines to communicate with a serial interfaced computer. When a message is received over the serial line, the main loop passes through different states. According to the headers of the message the microcontroller responds back to the message source or sets the PWM signals. The finite state machine given in Figure 3.18 obeys the protocol given in Appendix B.

3.3.2. Data Acquisition

As mentioned in the hardware section (Section 3.2.2), there are two types of sensors mounted on the vehicle: a webcam and a laser range scanner. The two capturings are performed under two grabber threads detailed in the following sections. The following sections also discuss the visualization and data dumping threads.

3.3.2.1. Camera. A thread, called the Video Grabber Thread, is spared for the video image capturing. The camera data was acquired over the USB connection. The camera was programmed to send 30 frames per second at a resolution of 320x240. The images supplied by the camera driver are of format YUV422 (or YUYV) which corresponds to

two bytes of color information for each pixel. We can calculate the data transfer rate on the camera connection as 4.5 megabytes per second.

The software has the ability (the Video Dumper Thread) to dump the images into the flash memory of the FitPC. Each frame is written on the flash under the file name of the time in the POSIX time format and the microseconds the image was captured at. The dumping is performed in raw format with no compression, not to lose any information and not to waste processing power. Thus all the information coming from the camera is saved into the flash, that is to be used to simulate the camera in further applications.

Furthermore, the software is able to convert the raw video format (YUV422) into YUV444 format for the obstacle detection algorithm and RGB24 format to be able to visualize for debug purposes. In order to visualize the images live, we utilized the Video Viewer Thread that uses Simple DirectMedia Layer library (SDL) [90] to print the images onto the screen.

3.3.2.2. Laser Range Scanner. The range scanner supplies distance information by using laser beams in order to determine the distances to reflective objects. The information corresponds to the area in the front of the vehicle with 270 degrees and with 5600 millimeters at maximum. We can collect data at 10 frames per second, and each frame consists of the distance values of 770 different orientations with 0.006 radians of difference between orientations. The data transfer rate on the range scanner connection can be calculated as 15 kilobytes of data, which is a negligible amount compared to the video data.

It is observed that a total of 30 degrees at the beginning and the end of the range data is erroneous and only the 240 degrees of central part with 682 bytes are correct. So, only this part of the data is captured using the Laser Distance Data Grabber Thread. The captured raw range data can also be written into the disk by the use of the Laser Data Dump Thread and can be viewed via the Laser Data Viewer Thread.

3.3.2.3. Disk Write Speed Constraints. Acquiring data at high speeds (∼4.5 megabytes per second) was a challenge. For the original hard disk of the FitPC was a mechanical one, it was replaced with a CF Card which had a capacity of one gigabyte. The write speed of the flash memories constitude a bottleneck for the data acquisition. Various tests was made to measure the write speed of the installed CF Card and of various USB flash memories (using the test tool *dd* of UNIX) Table 3.4. The free space left from the system files on the CF card could be extended up to 500 megabytes, which corresponds roughly to 120 seconds of data acquired from the sensors. The write speed of the CF Card was sufficient to sustain the required data acquisition speed, however, the lack of space on the CF Card forced us to attach another external flash memory on USB or to lower the grabber rate, which is not a desired situation.

Table 3.4. Write speeds of various medium types

The medium	Average Measured Write Speed (MB/s)
Hard Disk Drive on IDE	40.00
Sandisk Compact Flash Card on IDE	5.61
First USB Flash Memory on USB	2.76
Second USB Flash Memory on USB	0.732

3.3.3. Processing and Control

A single thread called the Processor Thread is considered for the processing and the vehicle control purposes because the control of the vehicle depends on the extracted environmental information and that they have to be placed one after another. There are two options for the autonomy of the vehicle in this thread. The vehicle can be driven either using distance estimation on visual information supplied by the Video Grabber or using laser scanner distance values supplied by the Laser Data Grabber. The detailed processing algorithms and control implementations are given in the obstacle avoidance chapter.

3.3.4. Communication

There are three seperate levels of communication interfaces of the the project. At the lowest level, as we have already mentioned in the motor controller circuit of the hardware section (Section 3.2.1.3), we have a simple PWM signal based protocol between vehicle motors and the microcontroller, the structure to set the motors to desired states. The microcontroller is fed with commands from the main processing unit, over a serial based communication. While for autonomous driving these two communication structures may be sufficient, for debugging purposes we had designed and implemented another communication infrastructure between the processing unit and an offboard control station.

3.3.4.1. Between the FitPC and the Micro-Controller. The communication between FitPC and the microcontroller is maintained over a serial data line. This part contains the microcontroller functions and the serial protocol used to maintain the communication between the microcontroller and the onboard computer. The FitPC sends the request commands to the microcontroller over this communication line, and the microcontroller processes the received command. Accordingly, the microcontroller sets a motor's PWM value or reports the actual state of itself, and finally returns a command, that states that the command has been executed.

With the routines used to generate the PWM, the microcontroller was also programmed as to be able to receive commands from the processing unit (FitPC). Using the UART routines, a protocol was designed to provide robust serial communication between two processors. The communication protocol is explained in detail in B.

The FitPC was designed to send corresponding DC motor commands (go backward, go forward, and stop) with a resolution of 50 levels and steering servo commands (turn wheels left, turn wheels right and get wheels straight) with a resolution of 60 levels, compatible with the values that the microcontroller can generate (Section 3.3.1).

3.3.4.2. Between the Control Station and the FitPC. For debug, manual control and data collection purposes, a communication protocol was designed between the onboard processing unit and a remote station unit. The two computers are supposed to share the same network and to use the Internet Protocol (IP). This connection model allows us to make connection using any means of modern network communication models such as Ethernet, Wireless LAN, or Bluetooth. Thus, while debugging, we would use direct cable connection over Ethernet, we could well use IP over Bluetooth in the case of cablefree connection when the vehicle is mobile.

On the FitPC, a thread called the Server Thread is commenced as a part of the controller software. The Server Thread runs as a listening TCP server, that accepts connections on a specified port. The control station has also a client program that is compatible with the Server Thread of the FitPC.

The TCP connection provides a control interface for the vehicle by a remote computer. Only using this stationary computer's keyboard, the vehicle can be driven at a fixed speed with fixed steering directions. If a joystick is detected, these maneuvers can be performed in an analog manner instead of fixed values. An emergency brake was also added in this communication structure to make the vehicle deccelerate as fast as possible.

For debugging and manual management, the Server Thread was given the capability to set or unset various flags that holds the control of three other threads. These controllable treads include the dumper threads and the Processor Thread. The dumper threads can be switched between saving images into the disk, no operation and sending images to the connected control station, though the last option has not been implemented yet. The Processor Thread was switched between manual piloting, automous behaviour, and both.

The Server Thread can also send the vehicle's actual state of the vehicle. The whole protocol is specified in Appendix C.

3.3.5. Control and Development Station

A variety of offboard programs were also designed in the scope of the project. These programs assist the user in offline debugging by simulating the collected data, in feature selection from a large spectrum of features and in training. The user can also control the vehicle from an offboard computer for both data collection and emergency conditions. The majority of the source code was written by C/C++, except for the feature selection tools which were coded in *MATLAB*.

3.3.5.1. Viewer. The robot is capable of collecting video images and scanned range data simultaneously from the environment it navigates through. In order to view these data, a viewer program compatible with the software of the FitPC was implemented. The viewer shows the images in RGB color space and the formatted range scanner data one after another simulating the original view of the vehicle (Figure 3.19).

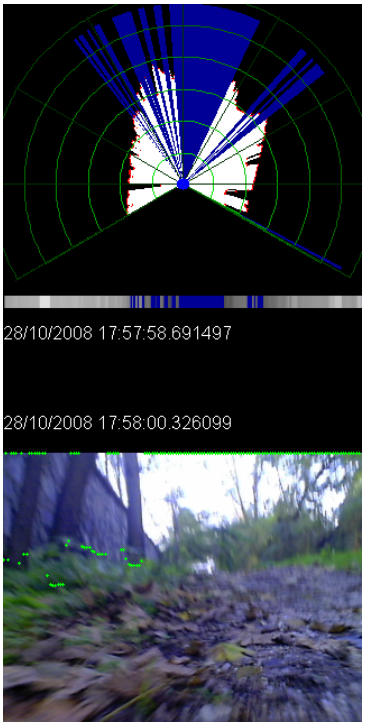


Figure 3.19. A screenshot of the viewer showing the video images and the range scanner data simultaneously

The viewer is not used only for viewing the data. Since it could simulate the robot view well, the software of the FitPC could be developed and tested using the viewer structure before being put onto the robot. For example, the software used for selecting the best steering direction was first developed with the viewer program, without actually needing to run the robot to get feedback. After being developed to some degree, the algorithms were tested on the robot. The robot code needed usually only fine tunings after being put on the robot.

3.3.5.2. Matcher. The data acquired from the two sensors are saved with timestamps with a resolution of microseconds. It is very improbable that images of two different types are taken exactly at the same time. So, a relation should be considered between these time stamps. The relation degree is defined as having no more than 20 miliseconds between a video image time stamp and a laser scanner image time stamp. A matcher was implemented to match these temporally close data points.

3.3.5.3. Labeler. The labeler program lets the user label the images with distance values by visually evaluating the stripes. The program has two modes, namely the navigation mode and the labeling mode.

In the navigation mode the images are displayed one after another in time order. The user can choose which image to label by navigating through the data set. If there are distance values already assigned for the stripes, they are also overlaid on the image.

In the labeling mode, the user can assign the distance values for each stripe on an unlabeled image or change the values of a previously labeled image (Figure 3.20(a)). The maximum distance that can be set is 30 meters.

If the estimation argument and the weight vector are also supplied in the program, the labeler displays the estimated distances of each stripe (Figure 3.20(b)).

(a) Labeling mode: The user is expected to set the distance values

(b) Navigation Mode: The user can navigate through the data set and visualize the estimated values

Figure 3.20. The labeler

3.3.5.4. Extractor. In the following chapter, we will be interested in finding the best features that keep most of the information related to the depth in a stripe of an image, but a search space must be determined and generated previously so that the feature selection methods can be used.

The feature extractor tool generates such a space. The generation is performed according to the patch types that are explained in Section 4.3.2. The extractor gets the labeled images as input and generates $(d+1)$-dimensional feature vectors as output. Each feature vector corresponds to a stripe in the image. The excess dimension is the depth information labeled by the supervisor.

3.3.5.5. Feature Selectors. The control station includes two subset selection tools. These tools are written as MATLAB functions and their algorithms are detailed in the dimensionality reduction section (Section 4.4).

3.3.5.6. TCP Client. The client on the control station was designed to communicate with the Server Thread of the FitPC software. It was supposed to acquire the state data of the robot in real time as the user cannot connect a monitor on the FitPC when the robot is mobile, however, the client program evolved to respond to different needs. The features of the client are parallel to the features of the Server Thread defined in Section 3.3.4.2.

4. DEPTH ESTIMATION AND OBSTACLE AVOIDANCE

4.1. Problem Definition

Using the range sensors to measure the distance to an object is quite straightforward. It is like using an echo, similar to bats sending sonar waves around and perceiving the surrounding obstacles. With the active sensors like light or laser based scanners, nowadays it is relatively easy to interpret the obstacle information compared to vision based interpreting. We had already pointed to various uses of these sensors in Chapter 2.

The fundamental drawback of these sensors is their cost. With high precision and range capabilities, the cost of these sensors increases drastically. For example, for the DARPA Grand Challenges many more million dollars were spent for the vehicles to win a reward of two million dollars.

Another drawback of these range sensors emerges if, for example, multiple numerous autonomous vehicles with these sensors are considered to navigate in a close neighborhood. The sonar based sensors are not even be considered due to their slow emission rate. When the subjects are laser or light based scanners, the emission rate is far more sufficient, but with the interference between multiple active sensors, high noise would be inevitable in such a case [19].

Besides their cost and noise drawbacks, the range scanner sensors suffer from their low ranges. The vehicles in the Grand Challenges used different scanners for different values of distance perception. The long range scanners and short range scanners were used complementarily to detect obstacles at different ranges. This suffering is also discussed in the following section by the help of the testbed robot we have designed.

Considering the high cost and low range of the laser scanner, a passive sensor like a camera can be used to estimate depth information in an image. The problem in the

camera case is called the monocular depth estimation. A person with one eye closed can still comment on the relative distance of an object standing in his or her field of view. With distance information estimated he or she can make inference on a path to follow in a random environment.

Odometry is another problem. The vehicle requires other sensor installations to achieve such odometry estimations. Our vehicle lacks an encoder for the wheels to estimate the vehicle angular speed or a battery state supplier to estimate the DC motor power so that the vehicle speed can be estimated over vehicle dynamics. It also lacks a GPS sensor at the moment to estimate the orientation of the vehicle. In indoor environments, the GPS sensor would not be usable. Under these circumstances, the odometry information that can be achieved from the commands given by the microcontroller would not be reliable, and rather would be risky to use in any obstacle avoidance method.

Because of the lack of even a little reliable odometry information, a path planning algorithm was not considered. The problem considered is rather making a robot vehicle detect the less risky direction and steer into this direction using computer vision as the tool. The problem can be divided into two parts. In the first place, the depth estimation problem is covered, and with the estimated distance values the robot vehicle will need to steer accordingly.

4.1.1. Autonomous driving using range scanner only

The first tests of the autonomous navigation were made using the laser scanner. The laser scanner we have provides information with a maximum range of 5.6 meters. We designed an obstacle avoidance algorithm that worked succesfully in indoor environments. The algorithm searches a drivable space in a maneuverable subspace of the scanned space of the laser scanner. It is a very primitive version of the method used in [91]. The algorithm is given in Figure 4.1. It takes the instantaneous laser images as input, and returns the steering directions as output.

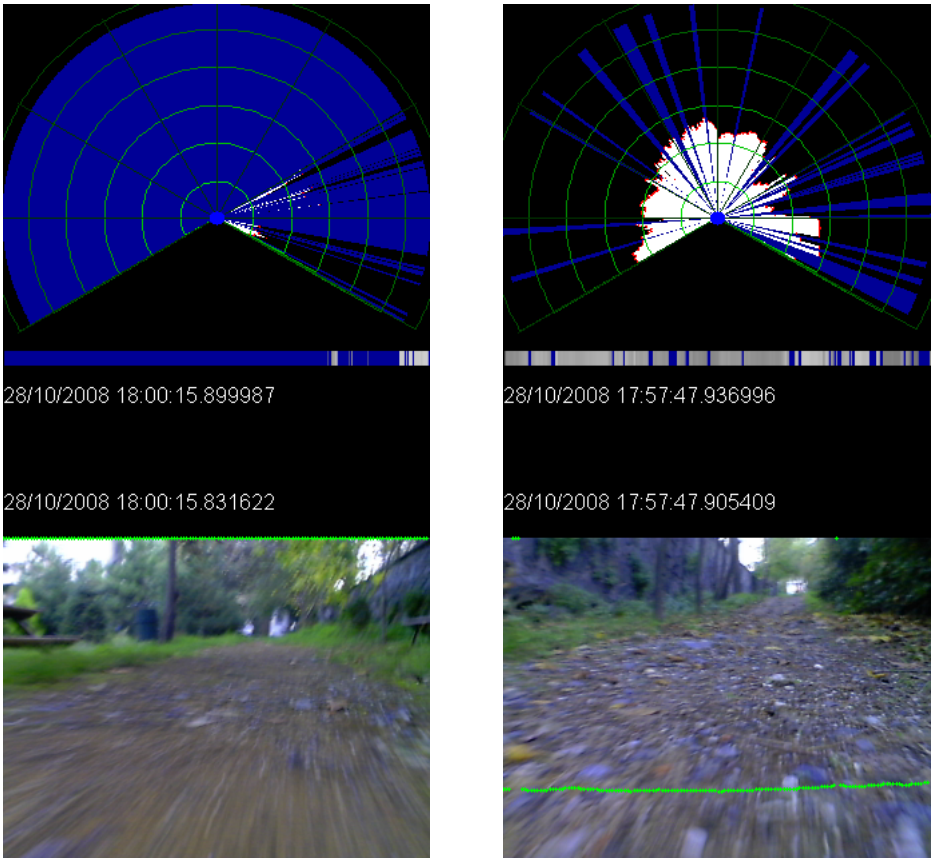
Input: *LaserInstant* {Instantaneous laser scan image}
Output: *SteerDirection* {Steering direction}
1: Initialize a *GapVector*
2: **while** end of the LaserInstant not reached **do**
3: **if** *GapInstance*.size ¿ PASSABLE **then**
4: Push *GapInstance*s into *GapVector*
5: **end if**
6: Get the center (*GapInstance*.center), border indices (*GapInstance*.start and *GapInstance*.end), bordering obstacle distances (*GapInstance*.startDist and *GapInstance*.endDist) of *LargestGap*
7: **end while**
8: **if** *GapVector*.size is zero *or* (*LargestGap*.start ¡ STARTCHECK *and* *LargestGap*.startDist ¡ STARTDISTCHECK) *or* (*LargestGap*.end ¡ ENDCHECK *and* *LargestGap*.endDist ¡ ENDDISTCHECK) **then**
9: return NONPASSABLE
10: **else**
11: return ReactiveSteerLaser(*LargestGap*)
12: **end if**

Figure 4.1. The algorithm that transforms the laser scan image to the steering angle

In the algorithm, a distance of risk was specified (*HIT_THRESHOLD*), so that the vehicle refers the obstacles closer than this distance as non-passable places. The 1D image of the laser scanner is divided into gaps found by a search of the passable directions, in $O(n)$. In one pass of the image, the possible gaps, *GapInstance*s, are found and added to the *GapVector*. After the pass, a steering model, similar to the model in Section 4.6, is supplied with the largest of the gaps (*LargestGap*), which includes the size, distance values at the borders, and the center index of the gap. If a *PASSABLE LargestGap* is found, the algorithm returns a steering direction and the robot is expected to steer along the center index, otherwise the robot is expected to stop.

In indoor tests, the robot wandered around without touching the obstacles, only with some small accidents especially with chairs and windows, however, driving with the scanner caused problems in outdoor environments. The required range information

is longer than the acquired information in indoor environments. In outdoor tests, there were cases in which the robot did not detect any obstacles around (Figure 4.2(a)), and although the desired behaviour is a left/right steered movement, the robot moved, for example, into bushes where it would get stuck and stop. Another issue is the characteristics of the unstructured environment. Because of the bumpy structure of the environment, the vehicle would get into a variety of pitch angles. The changes in the pitch caused the vehicle's laser scanner to scan the ground or the sky (Figure 4.2(b)), returning erroneous information, which results with a stuck condition or an accident.



(a) The range scanner cannot sense the obstacles more distant than its limited range

(b) An example of wrong interpreted range scanner data in front of a ramp

Figure 4.2. Examples from the test area: simultaneous range scan data and video images

4.2. Range Information Extraction

One of the aims of this study was to automatize the labeling and offline training procedure, and, if successful, to design an online training procedure, however, at the very first steps, it was observed that the training would fail against the matched data.

The laser scanner data was matched against the video images if there were at most 20 milliseconds of time gap between each other. The visualization of the overlaid data looked well to the eye as can be seen in Figure 3.19, but the data set contained a very large set of feature vectors matching to the maximum range of the laser scanner. Since the majority of the distance values in the unstructured environment would be more than 5.6 meters, training against the laser scanner data would not be realistic. Thus another kind of supervisor is required.

Humans perceive the depth correctly and quickly. Since the binocular vision is the dominant issue in perceiving depth, the perceptual cues such as color constancy, brightness constancy, and size constancy, play an important role in depth estimation [6], especially in one eyed case or in estimating depths in monocular images. These cues help people in estimating depth, even the ones with impaired vision. Our eyes are fully trained to estimate the depths using the cues, which in a more explanatory manner, can be listed as the relative object sizes, linear and aerial perspective properties, angular variations, and luminosity and shading variations [92].

The failure in the laser based automatized labeling procedure led us to labeling by hand, assuming the human view and judgement as an expert. First, different distance measurements were made in the test environment and the images were labeled by the expert depending on the almost correct depth perception of humans.

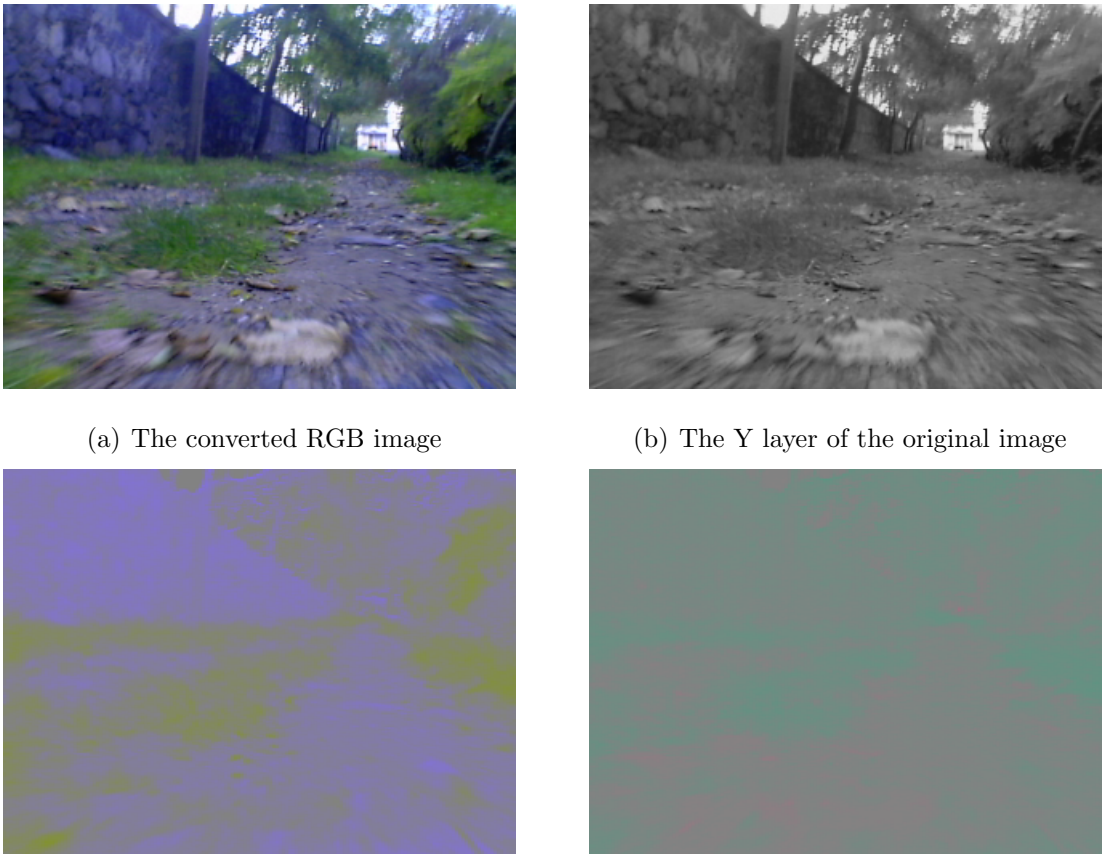
In the following section we discuss about the extraction of the feature vector from the image cues for each stripe of the image. For each image stripe w, the labeled data are added to the end of the vector as the $(d+1)^{\text{th}}$ dimension or the supervisor information of the feature vector.

4.3. Texture Energy Vector Extraction

We are dealing with three questions of computer vision in vector extraction: the color space to perform the search in, the parts of image to consider useful, and the information to be extracted from these parts.

4.3.1. Color Spaces

The camera driver supplies image frames in YUV format. The YUV format is not precisely defined in the technical and scientific literature. The name YUV is accepted as a name denoting a whole family of luminance and chrominance colorspaces. Besides, the YCbCr format was defined for television use in the ITU-R BT.601 standart for use with digital component video.



(a) The converted RGB image

(b) The Y layer of the original image

(c) The Cb layer of the original image

(d) The Cr layer of the original image

Figure 4.3. A conversion from the RGB color space to the YCbCr color space

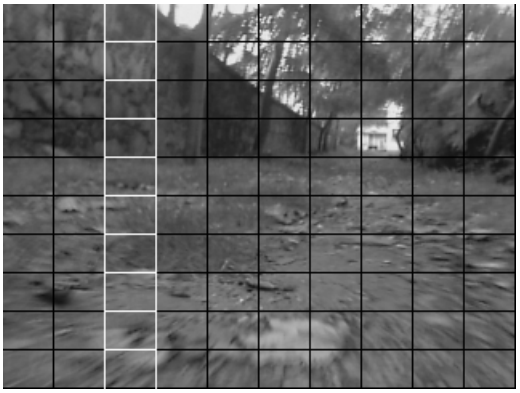
YCbCr is often confused with the YUV color space, and typically the terms YCbCr and YUV are used interchangeably, leading to some confusion. When referring to signals in video or digital form, the term YUV mostly means YCbCr. In this project, the images of two bytes (YUV422) supplied by the camera driver is first transformed to the images of three bytes (YUV444) to make it compatible with the sizes. We we will refer to these images as YCbCr images throughout the report.

The YCbCr format is composed of three layers, Y, Cb, and Cr as its name refers. The layer Y corresponds to the luminance or light intensity and the layers Cb and Cr correspond to the blue-difference and red-difference chroma components. A conversion from the RGB color space to the YCbCr color space is shown on an image taken from an unstructured environment in Figure 4.3.

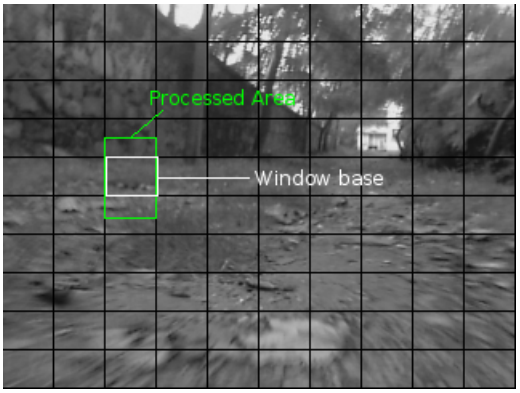
4.3.2. Image Patches

The images are first divided into W vertical stripes, so that each stripe corresponds to different directions. We use different variations of selection methods which is used to choose which information to include in the feature vectors about each stripe:

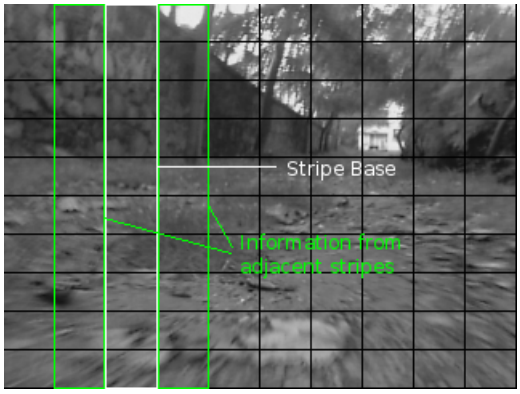
- In the first of the variations we divide each stripe into H smaller windows vertically. Each window has the size of exactly $\frac{1}{H}$ of the original image and no overlapping is permitted (Figure 4.4(a)). The feature vector for the subject stripe includes the features extracted from these separate windows.
- In a different approach, the stripes are divided again into H windows, but this time a window overlaps over the vertically adjacent windows, having a bigger vertical size than $\frac{1}{H}$ of the vertical size of the image (Figure 4.4(b)). The stripe's feature vector is constructed from the features of each h window again.
- As a third approach, the stripe's feature vector is constructed to include some global information. The stripe is divided again into H overlapping windows. In the construction of the stripe's feature vector, the information of its left and right stripes are also put into the vector of the subject image (Figure 4.4(c)). Thus the final vector has more dimensions compared to the previous two approaches.



(a) 10 window divisions on each stripe



(b) Overlapping windows



(c) Information from adjacent stripes

Figure 4.4. Three different patch selections

4.3.3. Feature Extraction

Our hypothesis is that a monocular color image hides the distance information in itself, and that if a human can more or less estimate the distances on an image there should be a way to make the machines make this estimation the same way humans do.

We are searching for some higher level information in an image that would help estimate the distances. For the extraction of this higher level information, texture analysis methods are considered. Among the various texture analysis methods, we will be interested in Laws' texture energy masks. These masks were previously defined in Section 2.3.1.

Laws' energy masks (Table 2.1) are applied on the image patches picked in the previous section. Eleven different features are extracted from a patch. Nine of these features are found by convolving nine Laws' masks with the Y layer (I_Y) of the patch. The remaining two features are computed by convolving the first mask with the Cb and Cr layers (I_{Cb} and I_{Cr}) of the subject patch. Finally an 11-dimensioned vector is generated for each patch after summing up the values of the output images of the convolution procedure (Equation 4.1). The feature extraction procedure is illustrated in (Figure 4.5). The operator '$*$' denotes the convolution masking operation, and the fs the features.

$$\begin{aligned} F(f) &= \sum_i \sum_j I_Y * L_f, \quad f = 1, 2, \ldots, 9 \\ F(10) &= \sum_i \sum_j I_{Cb} * L_1 \\ F(11) &= \sum_i \sum_j I_{Cr} * L_1 \end{aligned} \tag{4.1}$$

The images produced by the convolution masks expose diverse information types about the characteristics of the patches. By the nature of the masks, which are actually different kinds of edge detection, spot detection, and averaging masks, we may define these information types as luminosity, spot density, and edge density. In Table 4.1, these characteristics with the corresponding masks are listed.

Table 4.1. The features and the extracted information

Feature No	Mask	Characteristic
1	$L3^T L3$	Luminosity measure
2	$L3^T E3$	Amount of vertical edges
3	$L3^T S3$	Amount of bright spots in horizontal direction
4	$E3^T L3$	Amount of horizontal edges
5	$E3^T E3$	Amount of corners
6	$E3^T S3$	Amount of corners in vertical direction
7	$S3^T L3$	Amount of bright spots in vertical direction
8	$S3^T E3$	Amount of corners in horizontal direction
9	$S3^T S3$	Amount of local bright spots
10	$L3^T L3$	Blue chrominance measure
11	$L3^T L3$	Red chrominance measure

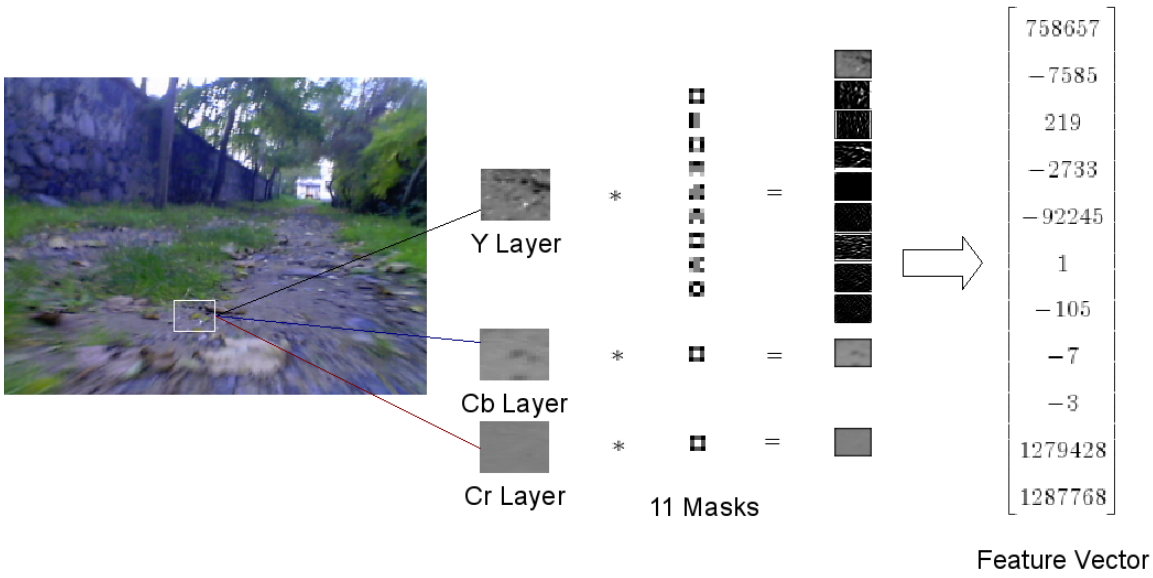


Figure 4.5. The feature extraction procedure for a patch of the image.

4.4. Dimensionality Reduction

The extraction of a feature corresponds to a convolution on a part of an image and a summation of all the values in the output of this convolution. The time to extract one feature's energy of a 32x24 sized window on an image (T_{feat}) was measured as 0.0004 seconds on average. This number would be multiplied by the number of features (N_{feat}) to be extracted in each slice of the image and again the final number would be multiplied by the number of windows (H) and the number of slices (W) in the image to obtain the time to process each image (T_{image}). The overhead of the preprocessing and parallel threads' effects ($T_{overhead}$) has to be added onto this time. The final calculated time (Equation 4.2) to process a single image with 110 features per slice was around five seconds, which was an extremely long time for a vehicle to drive at a speed of one meter per second. In [7], the vectors of each stripe have 429 dimensions. Each vector is composed of the associated stripe's window texture energies and the two adjacent stripes' energies. This methodology takes too much time on the processing unit to calculate all the features' energies.

$$T_{image} = T_{feat} \times N_{feat} \times H \times W + T_{overhead} \tag{4.2}$$

The aim is to have the vehicle process a new image per 200 millimeters at minimum while driving at one meter per second. That corresponds to five frames per second of processing rate. Under these constrained circumstances the feature vector's size had to be considerably decreased. Sequential measurements taken by decreasing the number of features randomly for each window gave a solution of 150 features. This corresponds to 15 features p stripe of the image if the image is divided into 10 vertical stripes with window sizes of 32x24 pixels.

Dimensionality reduction is a must in the process of training to have a reasonable processing time for the estimation of the distances on a video image. In [74], a similar dimensionality analysis on the feature vector is shown. The applied strategy was the Principal Components Analysis (PCA). The normalized eigenvalues of the decomposition shows that 10 values make up the 91.5 per cent and the smaller values do not contribute more than one per cent. So it was deduced that the dimensionality can be reduced to 10, however, because the PCA makes a mapping from the inputs in the original d-dimensional space to a new space of lesser dimensionality, the newly generated vectors are just the linear combinations of the original vectors.

While the PCA is one of the best known and most widely used feature extraction methods and moreover it is accepted as a strong method in vision-based algorithms like face recognition [93], its drawback in the purpose of this project is its being a linear projection method. We are not interested in finding a new set of k dimensions that are the combination of the original d dimensions. We are interested in finding k of the d dimensions that give us most of the information and we have to discard the other $(d - k)$ dimensions. This method corresponds to the feature selection in the dimensionality reduction methods [94].

Among the feature selection methods we chose the subset selection methods, in particular a combination of the sequential forward selection and the sequential backward elimination. In the subset selection methods, the problem is finding the best subset of the set of features that contribute most to accuracy. There are 2^{d} possible subsets of d variables, it is not reasonable to test for all of the subsets as d will not be

a small number (in this case $d \geq 110$). Some heuristics must be employed to get a reasonable solution in reasonable time. The result may not be optimal but the calculation time need to be in polynomial order. Besides the ordinary subset selection methods, we also included a feature selection method using genetic algorithms, which is found to yield better results compared to the subset selection methods.

4.4.1. Forward Selection

For the heuristic, the subset is initialized with having no features and one feature that decreases the error the most is added at a time. As such a heuristic search is a greedy search, the optimality cannot be guaranteed. Furthermore, because we cannot trust to a single subset returned by this method, we will need to strengthen this method at the end.

The data set (Equation 4.3) has N instances, each having d features of input (or d-dimensioned) and one feature of output, the $(d+1)^{\text{st}}$ feature, used for supervision.

$$S = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_d^1 & x_{d+1}^1 \\ x_1^2 & x_2^2 & x_3^2 & \cdots & x_d^2 & x_{d+1}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & x_3^N & \cdots & x_d^N & x_{d+1}^N \end{bmatrix} \tag{4.3}$$

The proposed process starts with the normalization of the full feature set S. The $(d+1)$-dimensioned instance vectors are normalized into the interval of $[-1, 1]$ by being divided by the maximum of the absolute co-dimension feature values (Equation 4.4).

$$x_k^t = x_k^t / \max_t(|x_k^t|), \quad \text{where } k = 1, 2, \ldots, d+1 \quad \text{and} \quad t = 1, 2, \ldots, N \tag{4.4}$$

The dataset is then reconstructed by randomizing the order of N instances to obtain a different instance distribution each time the process is repeated. The ran-

domized set is divided into two as a training set (Equation 4.5) and a validation set (Equation 4.6) each having $N/2$ instances. $\tilde{T}_X$ and $\tilde{V}_X$ are respectively the inputs of the training and the validation sets, expanded with the column of '1's that will serve as the linearity bias in the regression. $\tilde{T}_Y$ and $\tilde{V}_Y$ are the single columned output vectors.

$$\tilde{T}_X = \begin{bmatrix} 1 & \tilde{x}_1^1 & \tilde{x}_2^1 & \tilde{x}_3^1 & \cdots & \tilde{x}_d^1 \\ 1 & \tilde{x}_1^2 & \tilde{x}_2^2 & \tilde{x}_3^2 & \cdots & \tilde{x}_d^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \tilde{x}_1^{\frac{N}{2}} & \tilde{x}_2^{\frac{N}{2}} & \tilde{x}_3^{\frac{N}{2}} & \cdots & \tilde{x}_d^{\frac{N}{2}} \end{bmatrix} \quad \tilde{T}_Y = \begin{bmatrix} \tilde{x}_{d+1}^1 \\ \tilde{x}_{d+1}^2 \\ \vdots \\ \tilde{x}_{d+1}^{\frac{N}{2}} \end{bmatrix} \tag{4.5}$$

$$\tilde{V}_X = \begin{bmatrix} 1 & \tilde{x}_1^{\frac{N}{2}+1} & \tilde{x}_2^{\frac{N}{2}+1} & \tilde{x}_3^{\frac{N}{2}+1} & \cdots & \tilde{x}_d^{\frac{N}{2}+1} \\ 1 & \tilde{x}_1^{\frac{N}{2}+2} & \tilde{x}_2^{\frac{N}{2}+2} & \tilde{x}_3^{\frac{N}{2}+2} & \cdots & \tilde{x}_d^{\frac{N}{2}+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \tilde{x}_1^N & \tilde{x}_2^N & \tilde{x}_3^N & \cdots & \tilde{x}_d^N \end{bmatrix} \quad \tilde{V}_Y = \begin{bmatrix} \tilde{x}_{d+1}^{\frac{N}{2}+1} \\ \tilde{x}_{d+1}^{\frac{N}{2}+2} \\ \vdots \\ \tilde{x}_{d+1}^N \end{bmatrix} \tag{4.6}$$

A feature set full of vector numbers and a subset having no variables is initialized (Equation 4.7).

$$F_i = \{1, 2, \ldots, d\} \quad F_s = \emptyset \tag{4.7}$$

Every remaining feature, f, in the original set F_i is added temporarily into the subset F_s. For each addition, a multidimensional linear regression is applied on the training subset buffer, $\tilde{T}_{Xbuf}$, composed of the previously selected dimensions with the temporary feature f, to find a weight vector Ω_f of dimension $s_{buf} + 1$. The ordinary least squares is used to solve this overdetermined system of the regression analysis (Equation 4.8).

$$\Omega_f = (\tilde{T}_{Xbuf}^T \times \tilde{T}_{Xbuf})^{-1} \times \tilde{T}_{Xbuf}^T \times \tilde{T}_Y \tag{4.8}$$

The weight vector, Ω_s, found for each subset with different feature additions is then used to estimate the output on the validation subset buffer, $\tilde{V}_{Xbuf}$. A sum of squared errors is calculated on the validation set to complete the cross validation (Equation 4.9).

$$SSE_f = (\tilde{V}_T - \tilde{V}_{Xbuf}\Omega_f)^T(\tilde{V}_T - \tilde{V}_{Xbuf}\Omega_f) \tag{4.9}$$

From the temporarily added features, the one that decreases the validation error most is added into the subset F_s, and removed from the original set F_i (Figure 4.6).

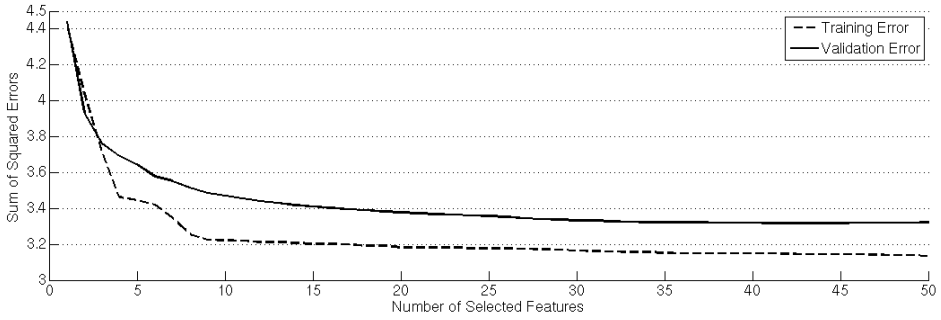


Figure 4.6. Sum of squared errors of a forward subset selection process

This forward selection process is repeated until there are $s \geq 15$ features selected in the subset and the whole process is repeated for a specified number of times to generate different subsets. By the nature of the greedy search used to select which feature to add and the randomization in separating the cross validation sets, the subsets' elements shows a variation of different feature indices and ordering. In the set of these different subsets, a frequency ordering of the features is made, that is, the features are voted by their selection times, and they are ordered from the most selected one to the least selected one. Finally, the leading 15 features are taken as the winners to be used in the real process.

4.4.1.1. Example of an unstructured environment. A data set with 1010 instances each having 110 features extracted from 101 images in an offroad environment was gener-

ated. An example result after 100 runs of the forward subset selection can be viewed in Figure 4.7. Each run was iterated to fill the subset with 50 features. It is clearly seen that some features do not play an important role in the training as they are selected for a few times, while some features are selected in the most of the runs. A list of the winners of this example run is given in Table 4.2.

Table 4.2. Ordering of the selected features by their frequency

Feature number in a stripe	1	32	55	44	33	109	21	45	72	11	65	22	31	10	7
Feature type	1	10	11	11	11	10	10	1	6	11	10	11	9	10	7

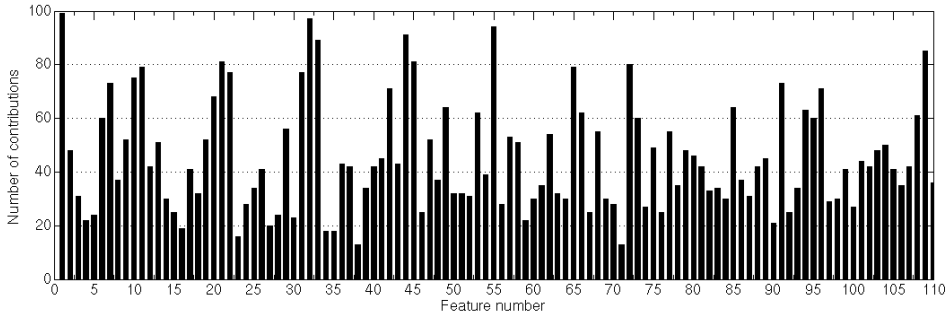


Figure 4.7. Number of contributions of each feature in a forward subset selection with 100 runs and 110 initial features

Considering that the combination of multiple features can decrease the error much more than adding them one by one, we could have taken the best feature combination by looking at the error of the 15^{th} step of the iterations Figure 4.8. The combination making the least error on the validation set is regarded as the winner this time, but the run numbers should again be limited to a specific number so that they finish in a reasonable time.

On the set of 110 features for each stripe of the image, it is interesting that, in both of the selection methods, these winner features have another common property. Most of them are extracted by applying the 11^{th}, 10^{th} and 1^{st} features of the image patches, and they correspond to the application of Laws' first mask ($L3^T L3$) on three layers of the YCbCr image (Table 4.3). We can conclude that features extracted from

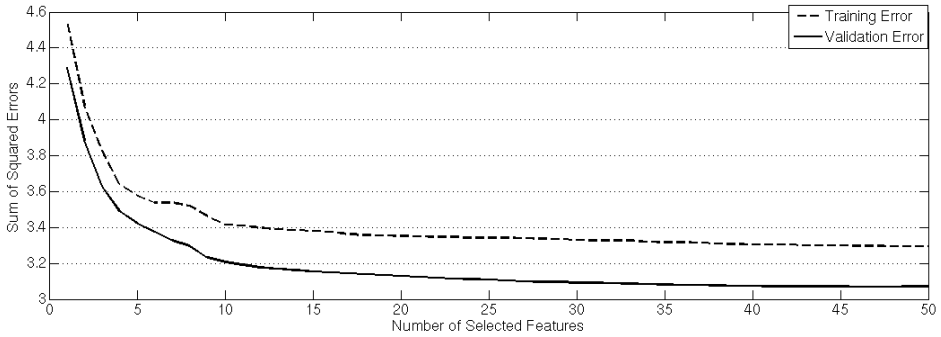


Figure 4.8. The best (least) of the sum of squared errors in 100 runs of the forward subset selection process

different layers of the image play an important role in the distance extraction algorithm, but of course this is data set specific and cannot be generalized.

Table 4.3. Features that decrease the error the most

Feature number in a stripe	55	1	33	45	89	77	44	21	32	91	31	72	7	49	2
Feature type	11	1	11	1	1	11	11	10	10	3	9	6	7	5	2

4.4.2. Backward Elimination

The backward subset search is the inverse of the forward search with respect to adding features into the subset. Just like the forward one, the backward search starts with a subset containing all the features and the number of features is decreased one by one throwing away the less contributing features. The complexity of the backward search has the same order of complexity as the forward search except that the system has to be trained with more features which would be more costly. The forward search is more preferable as the system is expected to have many useless features [93].

We use the backward elimination method to strengthen the method of forward searching. As mentioned in the previous section, a combination of multiple features may decrease the error much more than their single effects. In the forward search, the features were selected one by one, which could prevent the possible combined features'

effects. Making a backward subset elimination, we keep these features together, and help them be selected more frequently.

The same data set formation is used as in the forward subset selection (Equations 4.3, 4.4, 4.5 and 4.6). The same set full of feature numbers and a subset having no variables are initialized (Equation 4.7), however, this time the features are temporarily removed from the original set, F_i. The sum of squared errors (Equation 4.9) is calculated following the least squares method to find the weight vector each time (Equation 4.8). The process is iterated until there are a specific number of features remaining in F_i. The whole process is repeated again for a specific number of times to generate a distribution of distinct subset selections.

4.4.2.1. Example of an unstructured environment. On the same data set of 1010 instances, a backward subset elimination was run for 100 times. An example result of error variation can be viewed in Figure 4.9. There were 110 features in the set F_i initially. The runs were iterated until there are 50 features left in the set. Although the complexity is in the same order as the forward search, the number of iterations and the size of the sets increases the process time noticeably. The feature selection frequency graph is given in Figure 4.10 and their ordering is given in Table 4.4.

Table 4.4. Ordering of the selected features by their frequency

Feature number in a stripe	32	44	91	11	109	22	10	89	12	21	1	45	65	16	33
Feature type	10	11	3	11	10	11	10	1	1	10	1	1	10	5	11

To find the best subset of 15 features, the whole process had to be rerun for $s = 15$, which takes much more time. The error variation of the best subset, that has the least variation error at the 95^{th} $(110 - 15)$ iteration, can be viewed in Figure 4.11. The selected features of this run are given in Table 4.5. It can be seen that in the backward elimination the feature types that are contributing the most are parallel to the types found for the forward selection, except for that in the backward elimination the feature spectrum is wider.

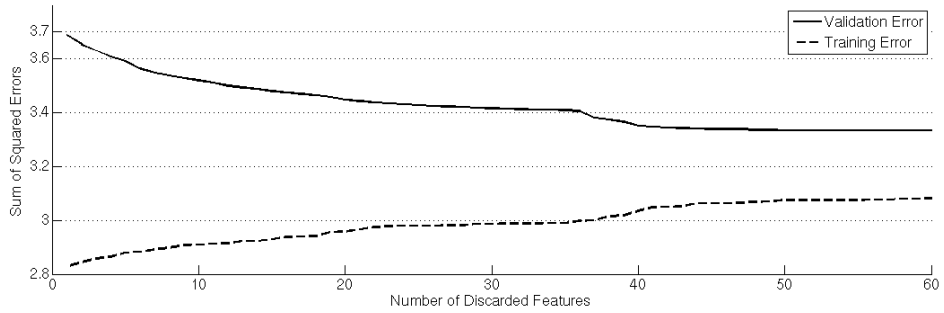


Figure 4.9. Sum of squared errors of a backward elimination process

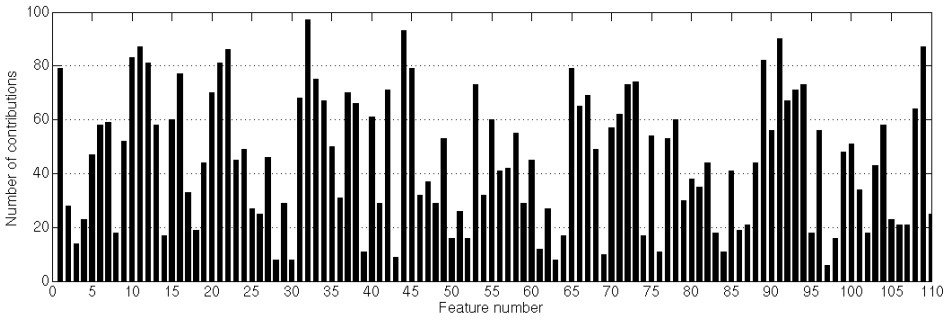


Figure 4.10. Number of contributions of each feature in a backward elimination with 100 runs and 110 initial features

4.4.3. Combining the Two Methods

Based on the results of the two search methods in opposite directions, another feature selection was made to determine the features that will be used in the actual training. It was assumed that the forward subset selection was powerful when there are many useless features to be discarded, and that the backward elimination method would keep the combined effects of multiple features together. Having run multiple times these two methods for $s \geq 15$ features in the subsets, the subsets are combined to search for which features are used most of the time. The frequency ordering made in the previous sections, is repeated again in the combined set of selected features. The contribution graph of the features is given in Figure 4.12 and the winning features that will be used for the processing in the example environment is listed in Table 4.6.

Table 4.5. Features that decrease the error the most

Feature number in a stripe	5	12	21	31	32	33	37	44	45	65	88	94	95	100	109
Feature type	5	1	10	9	10	11	4	11	1	10	11	6	7	1	10

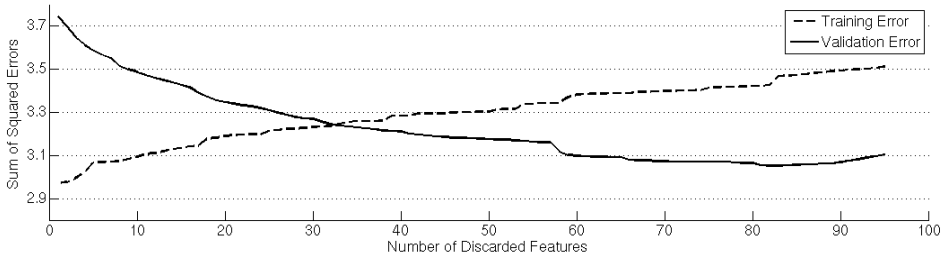


Figure 4.11. The best (least) of the sum of squared errors in 100 runs of the backward elimination process

An analogy can be made between the whole process and a tournament. A number of competitors (features) attend to the competition. They win some points by the errors they make in the validation set in different series of competitons. At the semifinals a list of the winners is acquired and in the final they are evaluated by the total points they won in each competition. The fifteen winners are the competitors that have won most of the points.

The dimensionality reduction process is run on the labeled images that have been taken from the environment in which the robot will be run autonomously. As it takes a long time to find the features the process is run offboard on a faster computer preferably. The selected features depend on the environment the images taken in, so that the vehicle has to be run in each environment previously to collect environmental data. It is also likely that the winner features on the data of two different environments or of two different data sets of the same environment would generate different formations.

Table 4.6. Selected features for the real processing

Feature number in a stripe	32	44	91	11	109	22	10	89	12	21	1	45	65	16	33
Feature Type	10	11	3	11	10	11	10	1	1	10	1	1	10	5	11

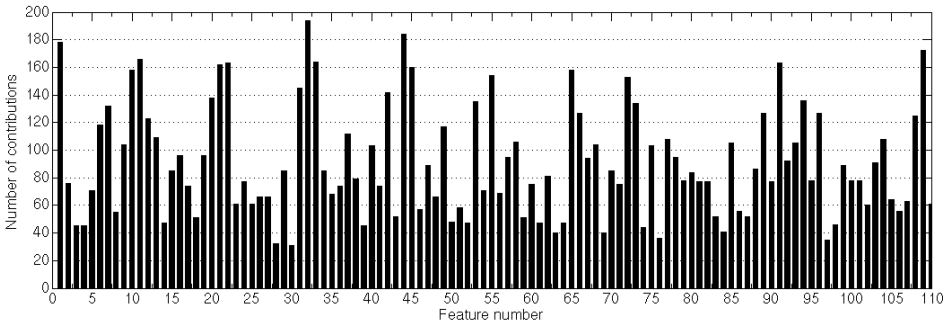


Figure 4.12. Sum of the number of contributions of each feature in two subset selection methods with 110 features after 100 runs of each one

4.4.4. Genetic Algorithm

The sequential forward selection and the sequential backward elimination are two non-exhaustive search procedures [95]. In these procedures the whole state space is not searched, and the optimal solution chance may be lost at the very first selection or elimination steps, because of the lack of backtracking options and because of the nature of the greedy method. The selected features are not removed anymore or the eliminated features are not selected any more.

Another method for feature selection can be considered to give chance to the previously eliminated features. A genetic algorithm can be used for feature selection. Whereas the optimality is not again guaranteed, the genetic algorithm selects in a more flexible way. The search space is not narrowed according to the step. The crossover and mutation operators add the tendency to expand the search space by adding features in a random but reasonable way [96].

$$C_i = \overbrace{\begin{bmatrix} 0 & 1 & 0 & 1 & \dots & 0 & 1 & 0 & 0 \end{bmatrix}}^{d \text{ features}}$$

Figure 4.13. The chromosome prototype

4.4.4.1. Chromosomes. In genetic algorithms, the chromosomes refer to the patterns of the individuals. For feature selection, a chromosome has a dimension d which will be reduced. Each gene that represents a feature's presence is considered as a binary value that decides whether the corresponding feature will be used or not in the error evaluation process (Figure 4.13).

4.4.4.2. Initialization. A population with P individuals is generated at the beginning of each run of the process. The chromosomes are initialized as to have '0's for each feature (or gene). A number, $s \geq 15$, of '1's is distributed randomly into the chromosomes of the initial population.

At the beginning, the data set that will help calculate the error of each individual is also randomized by the instance index to have a more homogeneous set.

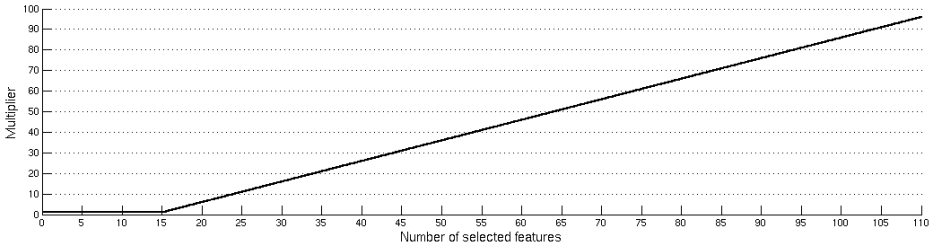
4.4.4.3. Evaluation. In order to evaluate the individuals of a population, a fitness function that uses the cross-validation method, and gives importance to less number of selected features in a chromosome is designed.

The cross-validation is performed on the data set that was initialized at the beginning. Without randomizing any more the data set, the same ditribution of the set is used for each run of the algorithm, to make the errors depend only on the feature selection. The data set is divided into two sets, as training and validation sets, where the training set is used to find a weight vector, Ω, with the least squares method, and the validation set used to calculate the error for each individual using this weight vector, like in the subset selection methods.

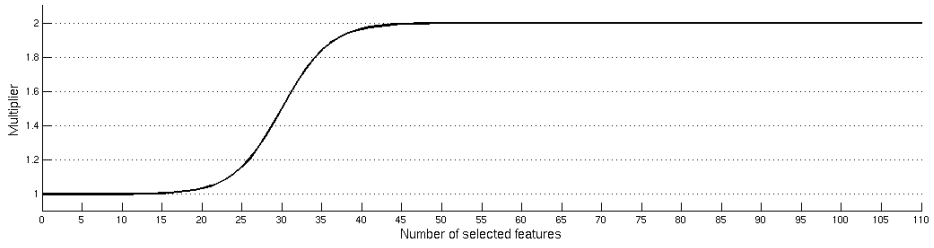
We also used a multiplier to add the effect of the selected features' size to the fitness function (Equation 4.10). As the hard constraint of $s \geq 15$ was previously determined, a factor that boosts the error for higher number of '1's is considered. Two different functions are used for this purpose. The first one is a combined function of two linear partial functions (Figure 4.14(a)), one of which does not change the error if the sum of the '1's in a chromosome is less than or equal to 15, and the other part is a linearly increasing function. The second multiplier is considered as a custom sigmoid function (Equation 4.11 and Figure 4.14(b)). The sigmoid is constructed to have the values close to 1 for the dimensions less than about 15, and for the higher number of selected features, the sigmoid returns higher values that boost the error.

$$M_{linear} = \begin{cases} 1, & \text{if } s \leq 15 \\ s - 14, & \text{if } s > 15 \end{cases} \tag{4.10}$$

$$M_{sigmoid} = 1 + \frac{1}{1 + e^{11 - \frac{s}{3}}} \tag{4.11}$$



(a) The linear function with two parts



(b) The custom sigmoid function

Figure 4.14. Error multipliers for the fitness function

The fitness function is finally determined as the error from the cross-validation times the multiplier from the multiplier function (Equation 4.12). The fitness value is denoted as FV. The multiplier M is one of the two multiplier functions (linear or sigmoid), and the sum of the chromosome s_{C_i} is given as a parameter to this multiplier. The $SSE(C_i)$ denotes the sum of squared errors of the chromosome C_i.

$$FV(C_i) = M(s_{C_i}) \times SSE(C_i) \tag{4.12}$$

4.4.4.4. Regeneration. A new population is generated using P_b of the best individuals of the population, that yield the least fitness values. Random pairs are selected from these P_b individuals that will play the role of parents for the new population. On the selected pairs, a single point crossover operator is applied. The location of the crossover is also determined randomly. Two new children are constructed from a pair. Each child is exposed to a mutation operator with a small probability, where a random gene of the chromosome is exchanged by its inverse, that is, if there's a '1' in the selected location of the chromosome, it is changed to a '0'. The chromosome generation procedure is repeated until the population size P is reached.

In our study, we used a population size of $P = 1000$ and the size of the best individuals were taken as $P_b = 30$. The mutation probability was 0.002. The initial population chromosomes included 60 randomly distributed '1's. The procedure was run until there was no more development in the error. The whole procedure was run for 30 times, and the best of 30 individuals that are selected as the best of each run is selected as the winning individual. The features selected by the winning individual are used in the training and in the software of the processing unit (Figure 4.15).

4.5. Training Model

Like in dimensionality reduction, the training of the robot vision depends on the multivariate linear regression, for, in [7], a linear model is found to be working as good as a more complex one.

Input: data set

Output: feature numbers

1: Initialize a population of size P

2: Evaluate the population

3: **while** The least error does not reduce **do**

4: Select P_b best of the individuals in the current population

5: Perform single point crossover and mutation on the selected individuals

6: Evaluate the population

7: **end while**

Figure 4.15. The genetic algorithm procedure

The training model uses the selected features from the output of the dimensionality reduction process, either one of the forward and backward methods or their combination. We prefered to use the first $s = 15$ features in our work as the robot's processing capability allows this many features to be processed in a reasonable time, which was explained in Section 4.4.

A data set S_s containing the s features selected after the reduction process is reconstructed (using the extractor software), that is, each stripe vector was reduced to a dimension of s to contain the concerned features. The vectors are not normalized with the maxima of the absolute values, as the forward estimation will be embedded into the processing unit. Extra computations are avoided considering the process restrictions. The same data set is used as before (Equation 4.13).

$$S_s = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_s^1 & x_{s+1}^1 \\ x_1^2 & x_2^2 & \cdots & x_s^2 & x_{s+1}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \cdots & x_s^N & x_{s+1}^N \end{bmatrix} \tag{4.13}$$

The model parameter vector $\Omega_s = [\omega_0 \ \omega_1 \ \cdots \ \omega_s \ ]$ are found using the whole set. Without dividing the data set, S_s, into any training or validation sets, we train the input dimensions against the output supervisor values to obtain a weight vector Ω_s.

The multivariate linear regression is used to find the line parameters ω_s that minimize the error most between each data element and the corresponding line value. We are interested in fitting a line on a multidimensional input vector X_s to find the line that is closest to the output vector points Y_s. The equation of the line is given in Equation 4.14.

$$y^t = \omega_0 + \omega_1 x_1^t + \omega_2 x_2^t + \cdots + +\omega_s x_s^t \tag{4.14}$$

The method of least squres is used to minimize the error (Equation 4.15).

$$\sum_{t=1}^{N} \varepsilon_t^2 = \sum_{t=1}^{N} (y^t - \omega_0 - \omega_1 x_1^t - \cdots - \omega_s^t x_s^t) \tag{4.15}$$

in matrix notation, $\hat{Y}$ denotes the estimated output value:

$$E = (Y - \overbrace{X\Omega}^{\hat{Y}})^T (Y - X\Omega) \tag{4.16}$$

Taking the derivative with respect to Ω, and without much more going into the details, we obtain:

$$\Omega = (X^T X)^{-1} X^T Y \tag{4.17}$$

It is very important to note that taking the inverse of $X^T X$ may fail on digital domains although it does not yield a singular matrix. This is because of the computers' limited precision capability. In such cases, taking the pseudoinverse of the matrix ($A^\dagger$) instead of the regular inverse would be more appropriate. In this work, we used Moore-Penrose pseudoinverse instead of the regular inverse computation [97, 98].

4.6. Reactive Steering Model

The vehicle is expected to navigate into an estimated safe direction. To behave as desired, a reactive steering model, which reacts immediately to the result of the distance estimation algorithm was designed. The model assumes that the stripe having the maximum distance value of the W estimated distances in an image corresponds to the best passage.

The wheels of the vehicle could be set to 50 different integer values, d_S ranging from -25 to 25, where the negative values correspond to the left steering, the positive values to the right steering. The stripe indices, w, range from 0 to 9. The index having the maximum of the estimated depth values, $h_M = argmax_h(e_h)$, is multiplied by a constant, K_B, to determine a base steering direction, d_B. The estimated depth values determine the aggressiveness of the steering. If the estimated distance values from the stripe that has the maximum value to the center of the image are low, which means that there are possible obstacles in front of the vehicle, the norm of the base steering direction is increased by an aggressiveness factor, K_A, inverse exponentially proportioned by the averaged values of these intermediate distances (Equation 4.18). N_e denotes the number of the intermediate distances and K_D is the decay parameter of the exponential.

$$d_S = \lfloor \overbrace{K_B(h_M - 4.5)}^{\text{base steering direction}} \overbrace{(1 + K_A e^{-K_D x})}^{\text{aggressiveness factor}} \rfloor, \text{ with } x = \begin{cases} \frac{1}{N_e} \sum\limits_{h_M \leq h \leq 4} e_h & \text{if } h_M \leq 4 \\ \frac{1}{N_e} \sum\limits_{5 \leq h \leq h_M} e_h & \text{if } h_M \geq 5 \end{cases} \tag{4.18}$$

In the implemented model, we limited the base steering direction to $[-15, 15]$ and the aggressiveness to $[0, 1.5]$, so that the steering direction would be limited to the controllable limits, $[-25, 25]$. To calibrate the steering model as desired, the constants are set to $K_B = \frac{10}{3}$, $K_A = 1.5$ and $K_D = 2000$.

In Figure 4.16, an explanatory example of the estimated values on an image is given. The farthest path is estimated as the first slice with a value of 15494.1 millimeters. Because the selected slice is the left most one, the base steering direction is set with a high normed negative value (-15 units). Moreover the depth of the slices in the adjacent slices are relatively nearer, which boosts the norm of the steering direction. The actual steering direction in this slice with the above configuration was calculated as -20 units, which would make the vehicle make a rapid left turn.

Figure 4.16. An example of the estimated distances

4.7. Summary of the Whole Procedure

- Acquisition of the images from the environment.
- Labeling of randomly selected images (Section 4.2).
- Extraction of the features from the labeled images (Section 4.3).
- Reducing the dimensionality using subset selection methods or genetic algorithms to find the most important features (Section 4.4).
- Training of a new weight vector on the whole labeled image set (Section 4.5).
- Modification of the robot software to use the selected features and installation of the weight vector.
- Running tests with the robot using the steering model (Section 4.6).

4.8. Tests and Results

4.8.1. Test Environment

The majority of the tests were made in an environment that can be accepted as unstructured. The test area has only the walls, garbage cans and fixed tables which can be referred as hand made structures (Figure 4.17). The length of the test area is about 110 meters. The width varies from four meters to 10 meters. The area has also height variations where the slope may increase up to 30 degrees. The terrain of the test area is grassy soil. Along the length of the area, the grass has decayed because of being treaded over by the people who uses the area as a pathway. The grass on the grassy area, the rocks, and the mud on the naked area make the terrain more rough and challenging. A view of the area generated by Google Earth software [99] is shown in Figure 4.18.

Figure 4.17. Views from the test area

4.8.2. Error Metrics

We consider three types of error metrics:

- The error between the actual labeled stripe distance and the estimated distance

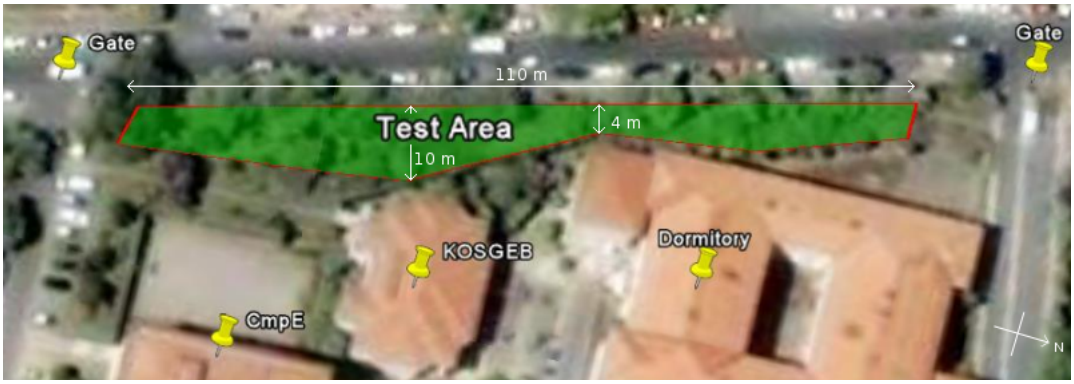


Figure 4.18. The test area

value of the same stripe is taken as the first error metric. The sum of the residuals of these values show how the computer estimates the human visual perception (Figure 4.19).

- The error between the number of the stripe having the maximum labeled distance and the number of the stripe having the maximum estimated distance is the second error metric. It is supposed to show the error in detecting the safest path (Figure 4.19).
- The difference between the length of the area and the traveled distance in the runs shows the third error metric. We measure how the system works as a whole by this metric.

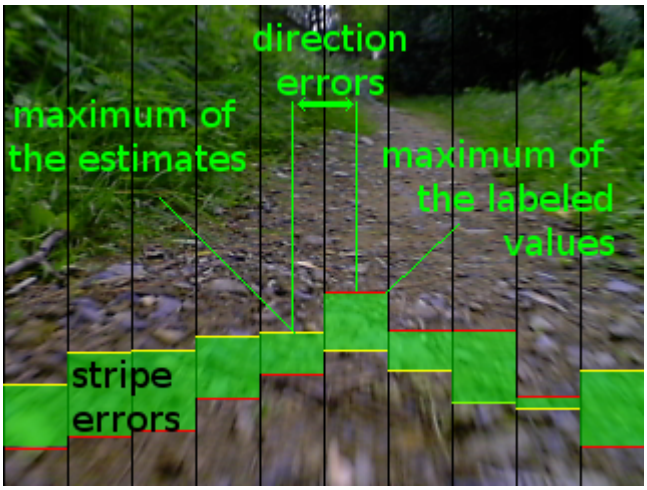


Figure 4.19. The first and the second error metrics

4.8.3. Test 1

The first set of the tests were performed on the test area after the vehicle was trained with the offroad images previously acquired from the area. We used 110 feature vectors for each stripe and 100 images to generate a search space. This makes a total of 1000 110-dimensional feature vectors. The combined subset selection method was applied on the data set, and the vectors were reduced to 15 dimensions which was then used to train the vehicle's software (Table 4.7).

Table 4.7. 15 features used for the first test (reduced from 110 features)

Feature number in a stripe	32	44	1	109	11	33	22	92	21	45	10	65	55	72	31
Feature type	10	11	1	10	11	11	11	3	10	1	10	10	11	6	9

The training was done with a linear regression with 15 features as input and labeled distance values as output to find a weight vector Ω_s, that minimizes the error the most. Although we do not need to make a cross-validation, we trained the vector on a division of 200 stripes of the whole set of 1000 stripes and plotted the validation errors, for the weight vector be more accurate. In Figure 4.20, these residuals are shown. The minimum, average and maximum errors calculated on this set are respectively 5, 3119 and 12405 millimeters.

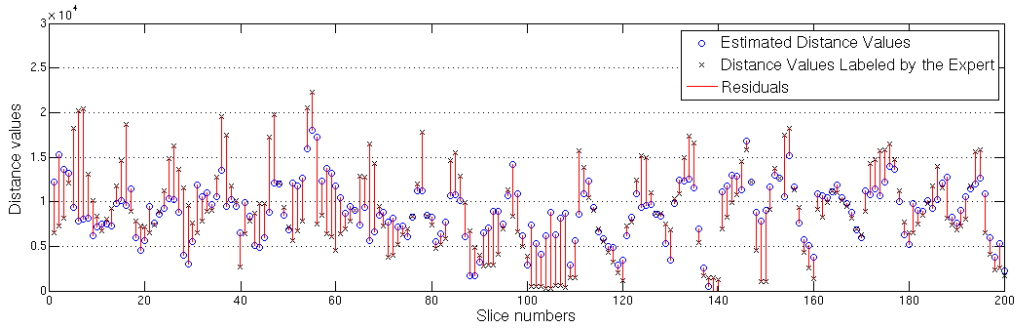


Figure 4.20. A sample of the residuals generated after the labeling for the first test

For the second metric on the first class of test data, the absolute path selection errors for 30 images are given in Figure 4.21. The maximum error and the minimum

error are respectively 9 and 0, as expected. An error of 9 corresponds to the condition that the labeled path as maximum on one end of the image is estimated at the other end of the image and an error of 0 corresponds to the condition that the path of the maximum distance is correctly estimated. The mean of these differences was calculated as 2.66.

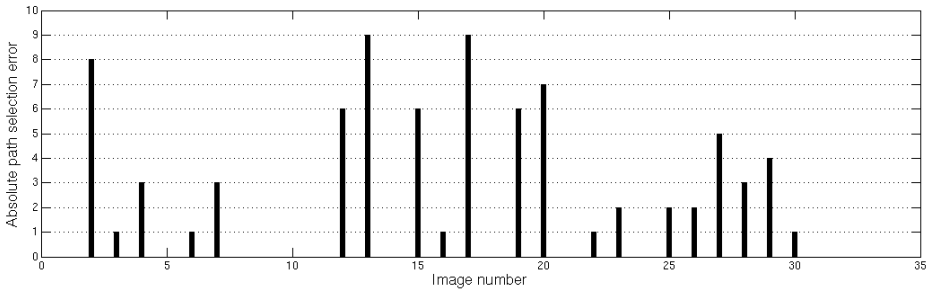


Figure 4.21. Differences of the selected paths between estimated and labeled images (30 images)

The vehicle steer control was done with the reactive steering model, which was specified in Section 4.6. For the third metric, the vehicle was put into autonomous mode to perform ten runs in the field starting from an offset of 30 meters from the south end of the test area. Table 4.8 shows the results of the test with the causes of stopping.

Table 4.8. Real world results from the first test

Run number	Appoximate length travelled (m)	Termination cause
1	72	reached the end of the area
2	38	stuck into mud
3	10	right turn after the start caused by wrong estimation
4	52	a fallen branch locked the wheels
5	15	hit a tree
6	25	went into a bush
7	30	hit a small tree
8	72	reached the end of the area
9	25	the rear wheels stuck on a rock
10	35	stopped at the ramp because of low battery for motors
Avg	34.7	

Total distance expected to be covered was 72 meters, but the vehicle made an error of 34.6 meters on average, while the average covered distance of the runs was 34.7. For the labeling procedure, only two image sets acquired from two different day light conditions were used to label a total of 100 images only, which is later discovered to be a small number to create a generic reduction of feature vectors and a weight vector. It was also noticed that the actual distances in the labeling procedure were given too near according to the real world measurement. The maximum distance was accepted as 30 meters, which was considered to be another reason that the test failed.

4.8.4. Test 2

Having realized that the labeling was far from accuracy of the actual values, the procedure was repeated on the images taken from six different image sets belonging to six different days. Thus, the data set had more images than the one in the previous test. Additionally, the maximum distance was increased to 60 meters, a more realistic value for the area. We needed to relabel 230 pictures to generate the data set. We also eliminated the features generated by the convolution with the gaussian kernel (feature numbers 1, 10 and 11) depending on [7], so the reduction was applied on the 80 features where the three features of each window $(110 - 3 \times 10)$was eliminated. Finally we obtained 15 features that contribute the most in the result by the combined subset selection method (Table 4.9).

Table 4.9. 15 features used for the second test (reduced from 80 features)

Feature number in a stripe	52	4	28	76	22	36	3	6	78	37	56	34	62	74	9
Feature type	5	5	5	5	7	5	4	7	7	6	9	3	7	3	2

We see that starting with a smaller set of initial features, the fifth and the seventh feature types contribute most in the result. Note that the fifth corresponded the corner density and the seventh the bright spots in the vertical direction.

In Figure 4.22, the residuals are shown. Minimum, average and maximum errors calculated on this set are respectively 10, 4931 and 10860 millimeters. We see that

while the average error was more than the average error obtained in the first data set, the maximum error was far less. We may say that the tendency in the estimated values are towards the more close distances, which boost the error.

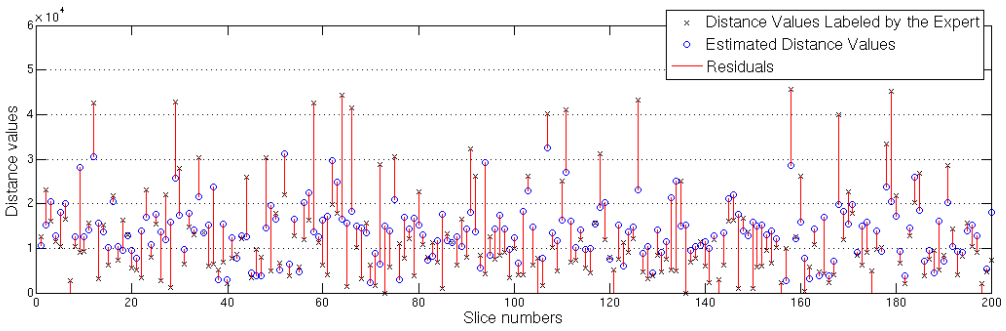


Figure 4.22. A sample of the residuals generated after the labeling for the second test

The overall performance was better than the performance of the first test, but was not good as expected, and there was a noticable tendency to catch the left side of the soil area. Later it was found that the steering angle of the wheels should have been recalibrated to prevent this tendency.

For the second metric, the absolute path selection errors for 30 images are given in Figure 4.23. The mean of these differences was calculated as 2.3, a better steering estimation.

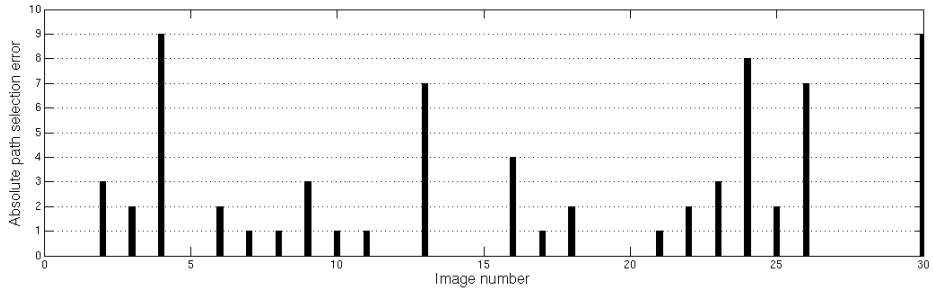


Figure 4.23. Differences of the selected paths between estimated and labeled images (30 images)

Table 4.10. Real world results from the second test

Run number	Appoximate length travelled (m)	Termination cause
1	78	hit a thin tree near the end of the area
2	18	hit the wall
3	45	correct distance seleciton, but wrong way
4	28	hit a thin tree
5	62	end of the area
6	33	hit a rock
7	32	hit a tree
8	33	reached the end of the area
9	83	out of sight of the expert (end of the area)
10	50	reached the end of the road
Avg	38.8	

The total distance expected to be covered was about 85 meters with different start points in the area. The vehicle was driven with three times more faster than the speed in the first test, which caused the vehicle react later than the image it processed. The average error was 38.8 meters, and the vehicle could travel 43.7 meters on average (Table 4.10). Whereas the test shows that there was not a significant development from the first test to the second, we may say that the vehicle's increased speed affected negatively the results.

4.8.5. Test 3

For the third test, the genetic algorithm, proposed in Section 4.4.4, was used for the dimensionality reduction. The initial feature vector for each stripe included again 110 features that belongs to the windows on the stripe itself. We used the same data set as the one in the second test: 230 labeled images acquired from six different days and hours. The data set consisted 2300 feature vectors.

The initialization population was generated with 40 random features. The final size of the selected features was determined by the error of the fitness function. For the fitness function the sigmoid multiplier was employed. The mutation factor was taken as 0.02. A population included 50 individuals and new chromosomes were regenerated

from the best 20 individuals of the current population that made the least error by the selected fitness function.

We made the genetic algorithm run for 30 times and gathered the best chromosome from each run. Among the 30 runs, again the best individual was selected as the winner. By the help of the sigmoid multiplier, the size of the selected features could oscillate around 15, and in the end, the best of these sizes were selected as 16. The selected features are given in Table 4.11.

Table 4.11. 16 features used for the third test (reduced from 110 features)

Feature number in a stripe	1	5	10	22	23	27	34	45	53	55	66	67	81	88	100	103
Feature type	1	5	10	11	1	5	1	1	9	11	11	1	4	11	1	4

The gaussian kernels are again dominant in the selected features, but the gaussian kernel based feature on the Cr layer was selected mostly, and we may generalize that the features 1, 4, 5, and 11 are the most common features selected by the genetic algorithm we have implemented.

In Figure 4.24, the residuals can be visualized. Minimum, average and maximum errors calculated on the set are respectively 60, 6207, and 31612 millimeters. All of the errors are higher than the ones of the previous tests.

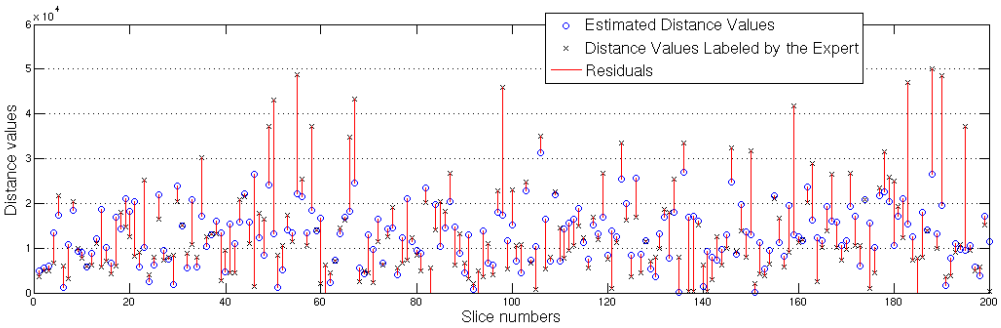


Figure 4.24. A sample of the residuals generated after the labeling for the third test

Although the local errors were higher compared to the errors of the previous tests, the second metric, the absolute path selection errors for 30 images were much better. The algorithm was more succesful in finding a safe direction to navigate into (Figure 4.25). The mean of the differences was calculated as 1.533, a much better steering estimation.

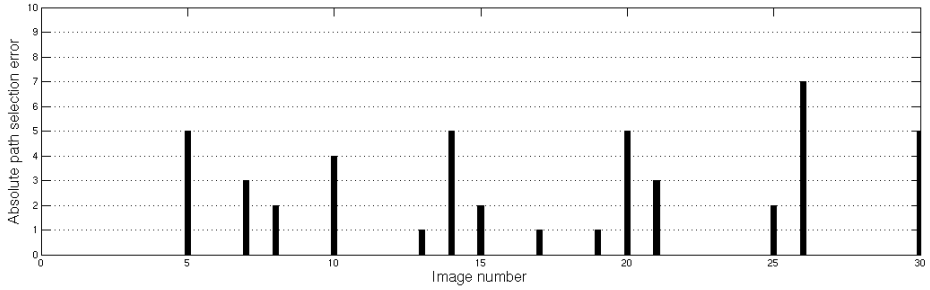


Figure 4.25. Differences of the selected paths between estimated and labeled images (30 images)

Table 4.12. Real world results from the third test

Run number	Travelled Distance(m)	Termination cause
1	13	hit a thin tree
2	42	crashed into a bright wall
3	13	stopped by a rock
4	85	end of the area
5	80	end of the area
6	57	hit a rock
7	67	into the bushes
8	85	end of the area
9	85	end of the area
10	64	end of the area
Avg	59.1	

The enhancement in the steering direction estimation also helped the third metric yield better results. The vehicle could navigate to the end of the area most of the time, and the average travelled distance was calculated as 59.1 meters. Taking the maximum distance as 85 meters, the average error was calculated as 25.9 meters.

The test shows a significant development from the previous tests to the third. The development factor is considered as the usage of genetic algorithms instead of subset selection methods.

4.8.6. Test 4

In the fourth test, we used the genetic algorithm once more to find the most important features from the 110-dimensional space. The search was made on the same data set of 2300 instances. The differences of this test from the previous one were the size of randomly distributed selected features in the chromosomes, the size of the populations, the mutation probability, and the multiplier of the fitness function.

We initialized the chromosomes with 60 random features, that is, the initial population of the genetic algorithm consisted of the individuals that had 60 randomly distribuete '1's. Each population had 100 individuals and a new population was generated from the best 20 individuals using crossover and mutation operators. The mutation operator had a probability of 0.002, which is 10 times smaller than the one of the previous test. Finally the multiplier factor of the fitness was taken as the two part linear function which was defined in Section 4.4.4.

We made 30 runs of the genetic algorithm. By the nature of the two part linear function, the feature size did not oscillate much around 15. It converged fastly to 15. After 30 runs, the best of 30 winning chromosomes determined the features to be used in weight vector calculation. These features are given in Table 4.13.

Table 4.13. 15 features used for the fourth test (reduced from 110 features)

Feature number in a stripe	5	12	19	21	22	25	31	34	45	54	61	67	78	82	88
Feature type	5	1	8	10	11	3	9	1	1	10	6	1	1	5	11

We see that the spectrum of the selected genes is larger than the one in the previous tests. Although the gaussian kernel based features (1, 10, and 11) take most of the place in the set, the variation of the feature types is noticeable.

The residuals are shown in Figure 4.26. Minimum, average and maximum residuals are respectively 25, 7211, and 39962 millimeters. These values are the highest, so the worst among the four test results.

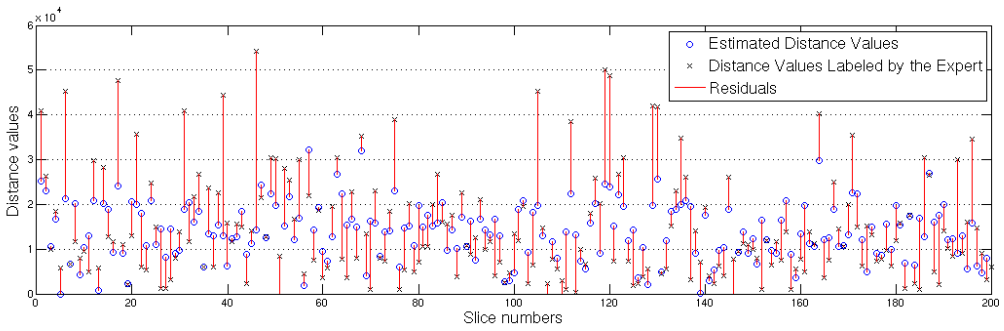


Figure 4.26. A sample of the residuals generated after the labeling for the fourth test

The second metric is shown in Figure 4.27. The average error in direction selection was 1.26, which means that we got better results in steering direction selection.

In Table 4.14, the effect of the second metric can be observed. The vehicle traveled more making the longest average travel distance as 73.7 meters after 10 runs. The vehicle showed a more stable navigation mostly trying to stay in the grassless part of the area.

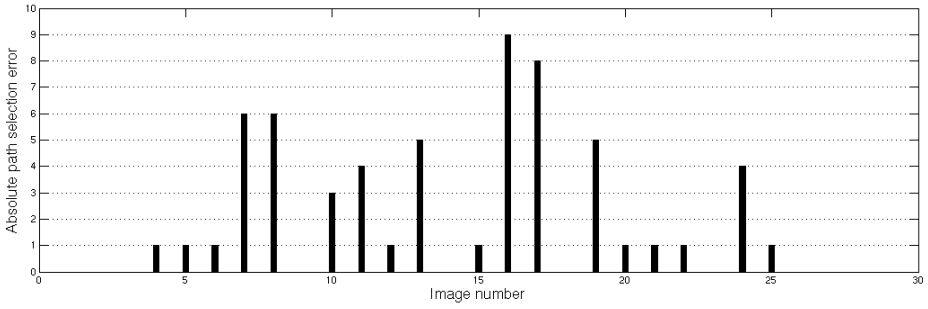


Figure 4.27. Differences of the selected paths between estimated and labeled images (30 images)

Table 4.14. Real world results from the fourth test

Run number	Travelled Distance(m)	Termination cause
1	35	hit a tree
2	95	end of the area
3	70	stuck into mud
4	95	end of the area
5	95	end of the area
6	19	hit a rock
7	95	end of the area
8	95	end of the area
9	95	end of the area
10	43	crashed into the wall
Avg	73.7	

4.9. Discussion

The overall performance of the vehicle depends on the initial feature space, the reduction method, size of the training data, speed of the vehicle, battery state of the vehicle, and the steering model. In the tests we tried different types of features selection methods. The first metric (residuals) shows us that the depth estimation by the subset selection methods worked better, but the overall performance of the feature selection using genetic algorithms was far more successful. Direction selection errors were less, and distances travelled in the test area without crashing into any obstacles were longer in the case of the genetic algorithm.

There are many more things that affect the performance of the vehicle. For example, the vehicle's speed was increased in the second test. Thus, we may regard the results of the second test as better than the first test, although there does not seems to be much difference in between. We observed that a small enhancement in the reduction method results in a small enhancement in the overall performance. There is more research to be done in this domain.

When we compare the last two tests, we notice that in the fourth test the search space was expanded by the population size of the genetic algorithm, and the initial pop-

ulation's feature numbers. The fitness function multipliers play also other important roles in affecting the overall performance.

5. CONCLUSIONS

We studied the autonomous vehicle concept in two parts. In the first place we showed the steps of the infrastructure construction process by solving electrical, electronical, computational, and communicational sub-problems. If, for example, we decide to make another autonomous robotic agent, we will have to follow the same methodology; drive the actuators by a computer, read the sensors from a computer, and interpret sensory data to the actuator controller. While the first two are considered as merely small implementation problems and regared as straightforward in many aspects, the interpretation of the data has a very large scaled research domain containing many disciplines from cognitive science to microcontrollers.

In order to interpret these data, we searched on the model of human perception system. We are not yet capable of simulating the system as a whole (the eyes with the visual cortex), but we may make some assumptions on how the obstacles are perceived by our brain. In this way, we took the example of the depth perception capabilities with only single eyes. The eye could be simulated at some degree by the camera technology, but the interpretation of the visual images is a widely open research area.

In this open research direction, we made our attempt in the domain of autonomous vehicles by this study. We searched for what part of the images are required for a robotic vehicle to accurately estimate the depth in front of it. From the selected parts of the images monocular cues that are also believed to be used by the human perception system, were extracted as image features. The followed methodology differs from common computer vision techniques. While in computer vision, we generally search for patterns in the image that lead us to some familiar features, in this work, we used computer vision techniques to transform the visual information to some meta information that is analyzed by the machine learning techniques. The aim was to find a generic method that we could use in most of the environments having different obstacle distribution and different light conditions.

We clearly made some advancements in this research domain, by contributing some uses of dimensionality reduction techniques in this problem. The robot could only navigate on visual sensors unless it had to process big amounts of data. It was then made available for the robot to run the algorithms in an acceptable time. We finally noticed that different models, different parts of the images and different features generated different errors. For example, the size of the training sets was a signicative factor. The size of the features also influenced badly as the subset selectors selected less number of features.

As a whole, we obtained encouraging results, experiencing that our robot could travel safely in a rough unstructured environment with changing light conditions. The robot made this navigation using its passive sensor, the camera. We believe that passive sensors are sufficient to percept all the environmental features, but at the moment, these data look too complex or too simple, respectively inefficient to process or uninformative about how to search for familiar patterns. With more researches in this imitation domain, we believe that more efficient and more generic algorithms will emerge.

APPENDIX A: Control Circuit Board

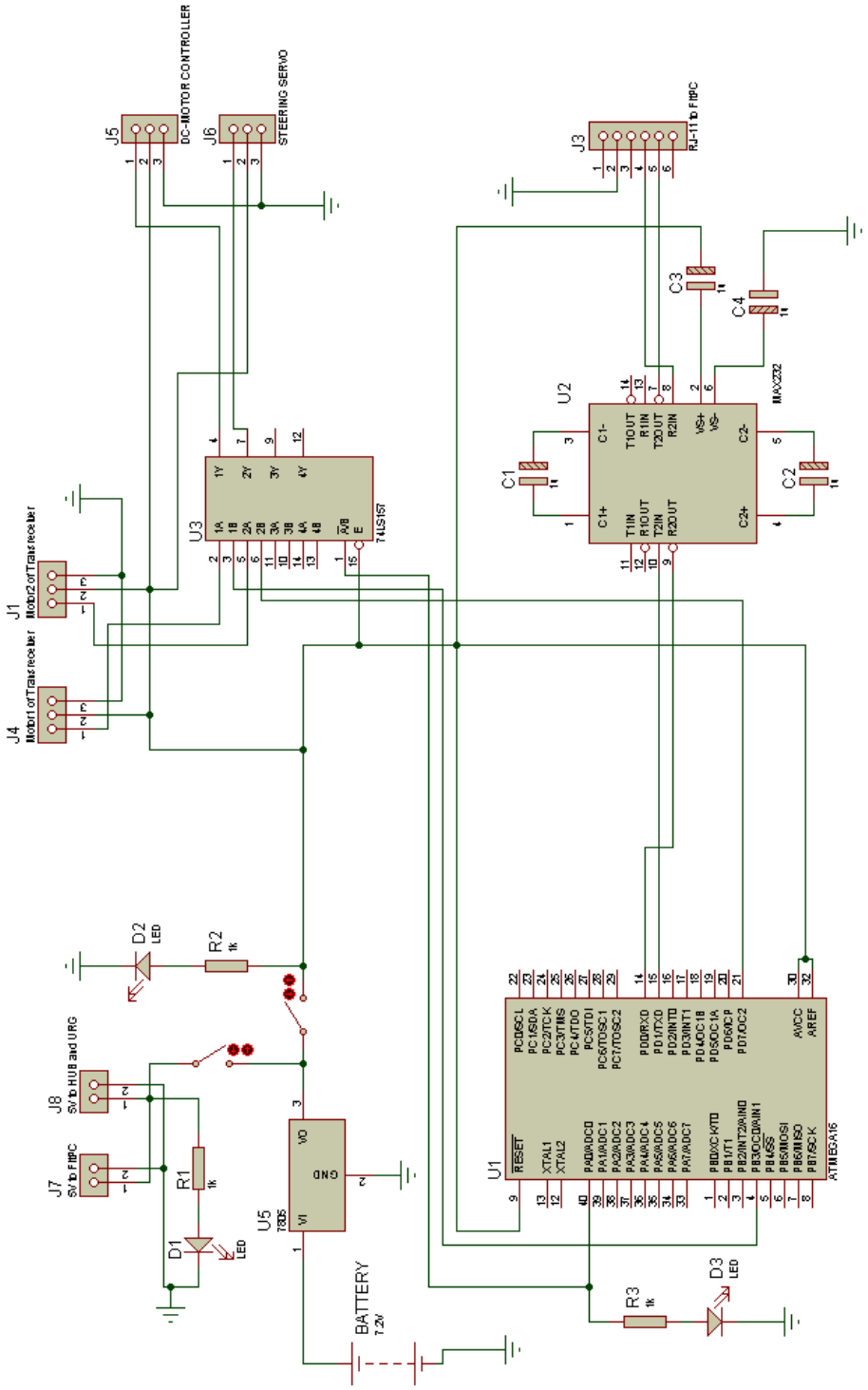


Figure A.1. Schematic of the custom circuit board

APPENDIX B: Protocol between the Microcontroller and the FitPC

The processing unit of the vehicle communicates with the microcontroller board over an RS-232 connection, using a custom transmission protocol. The processing unit sends abstracted motor commands to the microcontroller and the microcontroller transforms these commands into PWM signals to drive the motors. This section describes the designed communication protocol.

B.1. Command Format

The general command format is like the following sentence: <Start> <Command> <Parameters> <CRC>

The sentence is composed of different byte combinations of four different types. Each of three of these types, the start byte, the command byte and the CRC byte, are of one byte long and the parameters type is variable in length. The first two bytes can be regarded as the header, and the last byte as the footer. These three must be always present for the message sentence to be processable.

The start byte, which denotes the direction of the message, is the ASCII character 'K' for the commands that goes from the PC to the microcontroller. The responses are returned with a start byte of the character 'R'. The reason to consider different start bytes is to recontrol the command flow whenever there occurs a connection problem.

The command byte denotes the given command, and the parameters section was designed for the latter expansion capability.

The last byte, the CRC (Cyclic Redundancy Check), holds the computed CRC values from the command and parameters bytes. This is used to detect the problems in the transfer.

B.2. Command Types

B.2.1. Version

The command byte is sent to the microcontroller from the processing unit to learn the version of the firmware on the microcontroller at that moment. While this command type was designed for the version information acquisition, the actual usage was to test the connectivity and the messaging infrastructure between the microcontroller and the FitPC (Table B.1).

Table B.1. A version request example of the communication protocol

	Start	Command	Parameters	CRC
Request	K	v		%
Answer	R	i	0 x y	<CRC>

When the command byte of ASCII 'v' is sent to the microcontroller, it returns a message that includes the version and the revision numbers as x and y respectively.

B.2.2. Set PWM

When the connectivity test over version check is successful the connection is assumed to have established. The motor commands can be sent after then.

The command byte 'p' denotes the PWM command sentence in the protocol. The parameters starts with the bitmask byte that selects the PWM output channel connected to the motors. The PWM value in one byte follows this mask. If this sentence is successfully sent to the microcontroller, the microcontroller generates the associated PWM signal on the PWM channel that is selected by the bitmask which is then expected to make the motors turn as desired (Table B.2).

Table B.2. Example of setting the PWM value in the communication protocol

	Start	Command	Parameters	CRC
	Set the first PWM channel to 255			
Request	K	p	0x01 0xff	<CRC>
Answer	R	p	0x01 0xff	<CRC>
	Set the second PWM channel to 26			
Request	K	p	0x02 0x1a	<CRC>
Answer	R	p	0x02 0x1a	<CRC>

B.2.3. Report

The report command returns the last set PWM values. The command byte is 'r' with no parameters, and the return command byte is 'd' followed by the values of the PWM channels in order (Table B.3).

Table B.3. A report request example in the communication protocol

	Start	Command	Parameters	CRC
Request	K	r		<CRC>
Answer	R	d	0x38 0x78	<CRC>

APPENDIX C: Protocol between the FitPC and the Control Station

The message sentence sent from the control station to the Server Thread of the FitPC is composed of any sequence of the different tuples specified in Table C.1. Each tuple has a length of two bytes, first byte denoting the type of the tuple and the second the value. The whole sentence can be at most 30 bytes long, that can hold at most 15 tuples at a time.

The messaging is one-sided, from the control station to the FitPC, except for the report request. When the report is requested from the FitPC, a report structure is returned to the control station (Table C.2). The report has the same tuple structure as the messaging protocol but with a fixed size.

Table C.1. Request tuples from the FitPC to the control station

Type in ASCII	Value in hexadecimal	Function
B	0x01	Set the brake on
B	0x00	Set the brake off
T	0x00-0xff	Go forward (+) or backward (-) at a specified speed (signed)
S	0x00-0xff	Steer left (-) or right (+) with a specified direction (signed)
H	0x01	Set the horn on (not implemented)
H	0x00	Set the horn off (not implemented)
C	0x00	Set the camera dump off
C	0x01	Set the local camera dump on (save into the disk)
C	0x02	Set the network camera dump on (over network - not implemented)
L	0x00	Set the laser dump off
L	0x01	Set the local laser dump on (save into the disk)
L	0x02	Set the network laser dump on (over network - not implemented)
A	0x01	Toggle autonomous control on/off
M	0x01	Toggle manual control on/off
R	0x01	Request report

Table C.2. Respond tuples from the control station to the FitPC

Type in ASCII	Value in hexadecimal or in boolean	State
T	0x00 - 0xff	The speed the vehicle is last set to
S	0x00 - 0xff	The steer direction the vehicle is last set to
B	0/1	Brake status
A	0/1	Autonomy status
M	0/1	Manual control status
W	0/1	Video image writing status
L	0/1	Laser image writing status
V	0/1	Onboard video viewer status
R	0/1	Onboard laser viewer status
N	0/1	Video sending status
E	0/1	Laser sending status

REFERENCES

1. Pfeifer, R. and C. Scheier, *Understanding Intelligence*, MIT Press, Boston, MA, USA, 1999.

2. Dickmanns, E. D., "Vision for Ground Vehicles: History and Prospects", *International Journal of Vehicle Autonomous Systems*, Vol. 1, pp. 1 – 44, August 2003.

3. Broggi, A., M. Bertozzi, and A. Fascioli, "Architectural Issues on Vision-based Automatic Vehicle Guidance: the Experience of the ARGO Project", *Real Time Imaging Journal*, Vol. 6, No. 4, pp. 313–324, August 2000.

4. DARPA Grand Challenge Website, "What is the Urban Challenge?", 2008, `http://www.darpa.mil/grandchallenge/overview.asp`.

5. Hacker, P. M. S., "Is There Anything It Is Like To Be a Bat?", *Philosophy*, Vol. 77, No. 300, pp. 157–174, 2002.

6. Li, B., D. Xu, S. Feng, A. Wu, and X. Yang, "Perceptual Depth Estimation from a Single 2D Image Based on Visual Perception Theory", *PCM*, pp. 88–95, 2006.

7. Michels, J., A. Saxena, and A. Ng, "High Speed Obstacle Avoidance Using Monocular Vision and Reinforcement Learning", *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pp. 593–600, ACM, New York, NY, USA, 2005.

8. Harris, C. and D. Charnley, "Intelligent Autonomous Vehicles: Recent Progress and Central Research Issues", *Computing & Control Engineering Journal*, Vol. 3, No. 4, pp. 164–171, July 1992.

9. Durrant-Whyte, H. F., "Sensor Models and Multisensor Integration", *International Journal of Robotics Research*, Vol. 7, No. 6, pp. 97–113, 1988.

10. Brady, M., "Special Issue on Sensor Data Fusion", *International Journal of Robotics Research*, Vol. 7, No. 6, December 1988.

11. Sharir, M., "Algorithmic Motion Planning in Robotics", *Computer*, Vol. 22, No. 3, pp. 9–19, March 1989.

12. Leonard, J., H. Durrant-Whyte, and I. J. Cox, "Dynamic Map Building for Autonomous Mobile Robot", *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems 'Towards a New Frontier of Applications'*, Vol. 1, pp. 89–96, July 1990.

13. Elfes, A., *Sonar-based Real-world Mapping and Navigation*, pp. 233–249, Springer-Verlag Inc., New York, NY, USA, 1990.

14. Cox, I. J., *Autonomous Robot Vehicles*, Springer-Verlag Inc., New York, NY, USA, 1990.

15. Dickmanns, E. and A. Zapp, "Guiding Land Vehicles along Roadways by Computer Vision", *AFCET Conference*, pp. 233–244, October 1985.

16. Moore, C. and C. Harris, "Intelligent Identification and Control for AGV's Using Adaptive Fuzzy Based Algorithms", *International Journal of Engineering Applications of AI*, Vol. 2, pp. 267–285, 1992, `http://eprints.ecs.soton.ac.uk/273/`.

17. Brooks, R. A., "A Robust Layered Control System For a Mobile Robot", Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

18. Corfield, S., R. Fraser, and C. Harris, "Architecture for Real-time Intelligent Control of Autonomous Vehicles", *Computing & Control Engineering Journal*, Vol. 2, No. 6, pp. 254–262, November 1991.

19. Bertozzi, M., A. Broggi, and A. Fascioli, "Vision-based Intelligent Vehicles: State of the Art and Perspectives", *Robotics and Automation Systems*, Vol. 32, pp. 1–16, June 2000.

20. Wall, R., J. Bennett, and G. Eis, "Creating a Low-cost Autonomous Vehicle", *28th Annual Conference of IEEE Industrial Electronics Society*, Vol. 4, pp. 3112–3116, November 2002.

21. Rabbit Semiconductors, "Rabbit®2000 Microprocessors", 2009, `http://www.rabbit.com/`.

22. Chen, N., "A Vision-guided Autonomous Vehicle: an Alternative Micromouse Competition", *IEEE Transactions on Education*, Vol. 40, No. 4, pp. 253–258, November 1997.

23. Capozzo, L., G. Attolico, and G. Cicirelli, "Building Low Cost Vehicles for Simple Reactive Behaviors", *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 6, pp. 675–680 vol.6, 1999.

24. Holden, M. E., "Low-Cost Autonomous Vehicles Using Just GPS", *Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition*, 2004.

25. Goel, P. and G. Sukhatme, "Sonar-based Feature Recognition and Robot Navigation Using a Neural Network", *Proceedings of the International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 109–114, 2000.

26. Borenstein, J. and Y. Koren, "Real-Time Obstacle Avoidance for Fast Mobile Robots", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 5, pp. 1179–1187, 1989.

27. Gini, G. C. and A. Marchi, "Indoor Robot Navigation With Single Camera Vision", *PRIS*, pp. 67–76, 2002.

28. Kudo, H., S. M., T. Yamamura, and N. Ohnishi, "Measurement of the Ability in Monocular Depth Perception During Gazing at Near Visual Target-effect of the Ocular Parallax Cue", *IEEE International Conf. Systems, Man & Cybernetics*,

Vol. 2, pp. 34 – 37, 1999.

29. Abut, H., H. Erdogan, A. Ercil, *et al.*, "Data Collection with "UYANIK": Too Much Pain; But Gains Are Coming", *DSPincars*, June 2007.

30. Bertozzi, M. and A. Broggi, "Vision-Based Vehicle Guidance", *Computer*, Vol. 30, No. 7, pp. 49–55, 1997.

31. Whittaker, W., "Red Team Technical Paper - DARPA Grand Challenge 2004", Technical report, Carnegie Mellon University, Robotics Institute, 2004.

32. Urmson, C., J. Anhalt, M. Clark, *et al.*, "High Speed Navigation of Unrehearsed Terrain: Red Team Technology for Grand Challenge 2004", Technical Report CMU-RI-TR-04-37, Robotics Institute, Pittsburgh, PA, June 2004.

33. Thrun, S., M. Montemerlo, H. Dahlkamp, *et al.*, "Stanley: The Robot That Won the DARPA Grand Challenge: Research Articles", *Journal of Robotic Systems*, Vol. 23, No. 9, pp. 661–692, 2006.

34. Trepagnier, P. G., J. Nagel, P. M. Kinney, C. Koutsougeras, and M. Dooner, *KAT-5: Robust Systems for Autonomous Vehicle Navigation in Challenging and Unknown Terrain*, Vol. Volume 36 of *Springer Tracts in Advanced Robotics: The 2005 DARPA Grand Challenge*, pp. 103–128, Springer Berlin / Heidelberg, 2007.

35. Braid, D., A. Broggi, and G. Schmiedel, *The TerraMax Autonomous Vehicle*, Vol. 36 of *Springer Tracts in Advanced Robotics: The 2005 DARPA Grand Challenge*, pp. 130–153, Springer Berlin / Heidelberg, 2007.

36. Urmson, C., J. Anhalt, D. Bartz, *et al.*, "A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain", *Journal of Field Robotics*, Vol. 23, No. 8, pp. 467–508, August 2006.

37. Whittaker, W., "Red Team Technical Paper - DARPA Grand Challenge 2005", Technical report, Carnegie Mellon University, Robotics Institute, 2005.

38. Peterson, K., "Red Team Too Technical Paper - DARPA Grand Challenge 2005", Technical report, Carnegie Mellon University, Robotics Institute, 2005.

39. Wicks, A., B. Leedy, A. Blumer, R. Wilhelm, C. Bauman, S. Cacciola, and M. Webster, "Virginia Tech Team Rocky - DARPA Grand Challenge 2005", Technical report, Virginia Polytechnic Institute and State University, 2005.

40. Team Terramax, "TEAM TERRAMAX - DARPA GRAND CHALLENGE 2005", Technical report, Oshkosh Truck Corp, Oshkosh, WI and Rockwell Collins, Cedar Rapids, IA and Rockwell Scientific, Thousand Oaks, CA and University of Parma, Parma, Italy, 2005.

41. Carr, G. A. and B. Mee, "DARPA Grand Challenge 2005 - Team ENSCO's DEXTER", Technical report, ENSCO, 2005.

42. Stanford Racing Team, "Stanford Racing Team's Entry In The 2005 DARPA Grand Challenge", Technical report, Stanford Artificial Intelligence Laboratory, 2005.

43. Trepagnier, P. G., P. M. Kinney, J. Nagel, M. T. Dooner, and J. S. Pearce, "Team Gray Technical Paper - DARPA Grand Challenge 2005", Technical report, Tulane University, Gray & Company, Inc, 2005.

44. Urmson, C. and W. Whittaker, "Self-Driving Cars and the Urban Challenge", *IEEE Intelligent Systems*, Vol. 23, No. 2, pp. 66–68, 2008.

45. Urmson, C., J. Anhalt, J. D. Bagnell, *et al.*, "Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge", Technical report, Carnegie Mellon University, General Motors, Caterpillar, Continental AG, 2007.

46. Urmson, C., J. Anhalt, H. Bae, *et al.*, "Autonomous Driving In Urban Environments: Boss and the Urban Challenge", *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, Vol. 25, No. 1, pp. 425–466, June 2008.

47. Stanford Racing Team, "Stanford's Robotic Vehicle "Junior:" Interim Report", Technical report, Stanford University, 2007.

48. Kammel, S., B. Pitzer, S. Vacek, J. Schroder, C. Frese, M. Werling, and M. Goebl, "DARPA Urban Challenge Team AnnieWAY Technical System Description", Technical report, Karlsruhe University, 2007.

49. Reinholtz, C., T. Alberi, S. Frash, *et al.*, "Victor Tango DARPA Urban Challenge Technical Paper", Technical report, Virginia Polytechnic Institute and State University, 2007.

50. Montemerlo, M., J. Becker, S. Bhat, *et al.*, "Junior: The Stanford Entry in the Urban Challenge", *Journal of Field Robotics*, Vol. 25, No. 9, pp. 569–597, 2008.

51. Campbell, M., E. Garcia, D. Huttenlocher, *et al.*, "Team Cornell: Technical Review of the DARPA Urban Challenge Vehicle", Technical report, Cornell University, 2007.

52. Defence Advanced Research Projects Agency (DARPA), "DARPA Grand Challenge 2005 Rules", October 2004.

53. Porter, J., R. Behringer, R. Elsley, *et al.*, "The Autonomous Ground Vehicle RASCAL: Team SciAutonics/Auburn Engineering in the DARPA Grand Challenge 2005", Technical report, SciAutonics LLC, 2004.

54. Team Juggernaut, "Team Juggernaut DARPA Grand Challenge 2005 Technical Paper", Technical report, DesignJug, LLC, 2005.

55. Defence Advanced Research Projects Agency (DARPA), "Urban Challenge Rules Rules", October 2007.

56. Team MIT, "Technical Report - DARPA Urban Challenge", Technical report, MIT, 2007.

57. Trepagnier, P. G., J. Nagel, P. M. Kinney, M. T. Dooner, S. V. Drakunov, A. S. Lee, M. T. Dewenter, M. Hardey, and E. V. Gray, "Team Gray's 2007 Urban Challenge Vehicle", Technical report, GrayMatter, Inc., 2007.

58. The Ben Franklin Racing Team, "2007 DARPA Urban Challenge The Ben Franklin Racing Team Team B156 Technical Paper", Technical report, University of Pennsylvania, Lehigh University and Lockheed Martin Advanced Technology Laboratories, 2007.

59. Bohren, J., T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little Ben: The Ben Franklin Racing Team's Entry in the 2007 DARPA Urban Challenge", *Journal of Field Robotics*, Vol. 25, No. 9, pp. 598–614, 2008.

60. Todt, E. and C. Torras, "Color Constancy for Landmark Detection in Outdoor Environments", *In 4th European Workshop on Advanced Mobile Robots (Eurobot'01)*, pp. 75–82, 2001.

61. Davies, E. R., *Machine Vision : Theory, Algorithms, Practicalities*, Morgan Kaufmann, December 2004.

62. Cantzler, H., "An Overview of Image Texture", 2004, `http://homepages.inf.ed.ac.uk/`.

63. Tuceryan, M. and A. K. Jain, *Texture Analysis*, chapter 2.1, pp. 207–248, World Scientific Publishing Co., 1998.

64. Laws, K., *Textured Image Segmentation*, Ph.D. thesis, Department of Electrical Engineering, University of Southern California, Los Angeles, Calif, USA, 1980.

65. Laws, K., "Rapid Texture Identification", *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Vol. 238 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 376–380, 1980.

66. Saxena, A., S. H. Chung, and A. Ng, "Learning Depth from Single Monocular Images", Weiss, Y., B. Scholkopf, and J. Platt (editors), *Advances in Neural Information Processing Systems 18*, pp. 1161–1168, MIT Press, Cambridge, MA, 2006.

67. Naito, T., T. Ito, and Y. Kaneda, "The Obstacle Detection Method Using Optical Flow Estimation at the Edge Image", *IEEE Intelligent Vehicles Symposium*, pp. 817–822, June 2007.

68. Wikman, T., M. Branicky, and W. Newman, "Reflexive Collision Avoidance: A Generalized Approach", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 31–36, May 1993.

69. Khatib, O., *Real-time Obstacle Avoidance for Manipulators and Mobile Robots*, pp. 396–404, Springer-Verlag Inc., New York, NY, USA, 1990.

70. Quinlan, S. and O. Khatib, "Elastic Bands: Connecting Path Planning and Control", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 802–807, May 1993.

71. Hilgert, J., K. Hirsch, T. Bertram, and M. Hiller, "Emergency Path Planning for Autonomous Vehicles Using Elastic Band Theory", *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Vol. 2, pp. 1390–1395, July 2003.

72. Borenstein, J. and Y. Koren, "The Vector Field Histogram-fast Obstacle Avoidance for Mobile Robots", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, pp. 278–288, June 1991.

73. Kodagoda, K., W. Wijesoma, and E. Teoh, "Fuzzy Speed and Steering Control of an AGV", *IEEE Transactions on Control Systems Technology*, Vol. 10, No. 1, pp. 112–120, January 2002.

74. Akce, A., "Development of a Mobile Autonomous Outdoor Robotics Platform", June 2007, senior Project at Department of Computer Engineering, Bogazici University, Istanbul.

75. Tamiya INC., *Blackfoot Xtreme*, 2003, `http://www.tamiya.com`.

76. Atmel Corporation, *Pulse Width Modulation*, 2001, `http://www.atmel.com`.

77. Atmel Corporation, *ATMEGA16: 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash*, 2008, `http://www.atmel.com`.

78. Logitech Inc, *Logitech QuickCam Pro 5000*, 2008, `http://www.logitech.com`.

79. Pinchart, L., "Linux Driver and User-space Tools for USB Video Class Devices", 2008, `http://linux-uvc.berlios.de/`.

80. Kavaklioglu, C., *Developing a Probabilistic Post Perception Module for Mobile Robotics*, Master's thesis, Boğaziçi University, 2009.

81. Xhaard, M., "Small Video Capture Program Ideal for Webcam Testing and Problem Debugging", 2005-2008, `http://www.quickcamteam.net/software/linux/v4l2-software/luvcview`.

82. Hokuyo Automatic Co. LTD, "Scanning Range Finder URG", 2008, `http://www.hokuyo-aut.jp/02sensor/07scanner/urg.html`.

83. Pavlik, V., A. Fuerst, P. Machek, J. Erdfelt, O. Neukum, and D. Kubicek, "USB Abstract Control Model Driver for USB Modems and ISDN Adapters", 1999-2005, `http://www.kernel.org/doc/`.

84. Gerkey, B., "The Player Robot Device Interface", 1999-2007, `http://playerstage.sourceforge.net/`.

85. Gumstix, Inc., "Gumstix - Dream, Design, Deliver", 2008,

http://www.gumstix.com/.

86. Fujitsu Siemens Computers, "Pocket Loox C Series", 2007, http://www.pocketloox-choice.com/.

87. Vermeulen, S., G. Goodyear, R. Marples, *et al.*, *Gentoo Linux x86 Handbook*, September 2008, http://www.gentoo.org/doc/en/handbook/handbook-x86.xml.

88. Bluetooth SIG, Inc., "Bluetooth Network Encapsulation Protocol (BNEP) Specification", 2001, http://grouper.ieee.org/groups/802/15/Bluetooth/BNEP.pdf.

89. CollabNet Inc., "Subversion, an Opensource Version Control System", 2001-2008, http://subversion.tigris.org/.

90. Langinga, S., "Simple Directmedia Layer Library", 2008, http://www.libsdl.org/.

91. von Hundelshausen, F., M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with Tentacles: Integral Structures for Sensing and Motion", *Journal of Field Robotics*, Vol. 25, No. 9, pp. 640–673, 2008.

92. The National Institute for Rehabilitation Engineering, "Visual Distance Perception and Depth Perception", 2004, http://www.abledata.com/abledata_docs/Distance-Perception.pdf.

93. Alpaydin, E., *Introduction to Machine Learning*, The MIT Press, 2004.

94. Miller, A., *Subset Selection in Regression*, Chapman & Hall/CRC, 1990.

95. Foroutan, I. and J. Sklansky, "Feature Selection for Automatic Classification of Non-Gaussian Data", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 17, No. 2, pp. 187–198, 1987.

96. Yang, J. and V. Honavar, "Feature Subset Selection Using a Genetic Algorithm", *IEEE Intelligent Systems and their Applications*, Vol. 13, No. 2, pp. 44–49, May 1998.

97. Wisstein, E. W., "Moore-Penrose Matrix Inverse", 1999 – 2009, `http://mathworld.wolfram.com/`.

98. Penrose, R., "A Generalized Inverse for Matrices", *Proceedings of the Cambridge Philosophical Society*, Vol. 51 of *Proceedings of the Cambridge Philosophical Society*, pp. 406–413, July 1955.

99. Google Inc., "Google Earth", 2009, `http://earth.google.com/`.