UNIVERSITY MOHAMED KIDHER -BISKRA

Obstacle avoidance of an autonomous car using Raspberry Pi

by

BAIDJI El-hadj Ahmed

A thesis submitted in fulfillment for the degree of
MASTER

in the
Faculty of Exact Sciences and Sciences of Nature and Life
Department of Computer Science

July 2019

Declaration of Authorship

I, BAIDJI El-hadj Ahmed, declare that this thesis titled, 'Obstacle avoidance of an autonomous car using Raspberry Pi' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"People if they return to their minds, each of them mind."

- Ibn Taymiyya

UNIVERSITY MOHAMED KIDHER -BISKRA

Abstract

Faculty of Exact Sciences and Sciences of Nature and Life
Department of Computer Science

MASTER

by BAIDJI El-hadj Ahmed

The project aims to build a monocular vision autonomous car prototype using Raspberry Pi as a processing chip. An HD camera along with an ultrasonic sensor is used to provide necessary data from the real world to the car. The car is capable of reaching the given destination safely and intelligently thus avoiding the risk of human errors. Many existing algorithms like lane detection, obstacle detection are combined together to provide the necessary control to the car.
Keywords: vision , autonomous car ,Raspberry Pi,sensor, avoiding , obstacle .

Resumé

Le projet vise à construire un prototype de voiture autonome à vision monoculaire utilisant le Raspberry Pi comme puce de traitement. Une caméra HD ainsi qu'un capteur à ultrasons sont utilisés pour fournir les données nécessaires du monde réel à la voiture. La voiture est capable d'atteindre la destination donnée en toute sécurité et intelligemment, évitant ainsi le risque d'erreurs humaines. De nombreux algorithmes existants, tels que la détection de voie, la détection d'obstacle, sont combinés pour fournir le contrôle nécessaire à la voiture.
mots clé: voiture autonome , vision ,Raspberry Pi, capteur ,évitant , d'obstacle .

Acknowledgements

I would like to express my special thanks of gratitude to my teacher **CHERIF FOUDIL** as well as our principal **BABA HENINI MOHAMED CHOUKI** who gave me the golden opportunity to do this wonderful project on the topic **"Obstacle avoidance of an autonomous car using Raspberry Pi"**, which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them. Secondly i would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **iv**

List of Figures **ix**

List of Tables **xi**

List of programs **xii**

Abbreviations **xiii**

Symbols **xiv**

Introduction **1**
- 1. Preface . . . 1
- 2. Purposes and goals of the thesis . . . 1
- 3. Structure of the thesis . . . 2

1 Robots and Their Applications **3**
- 1.1 Definition of robot . . . 5
- 1.2 Classification of Robots . . . 5
- 1.3 Framework of roadmap of robotics . . . 7
- 1.4 Robotics Applications . . . 8
  - 1.4.1 Industrial Robots . . . 8
  - 1.4.2 Autonomous Mobile Robots . . . 9
  - 1.4.3 Educational Robots . . . 11
- 1.5 Welfare benefits and advantages of robotics . . . 12
- 1.6 Biggest Challenges in Robotics . . . 13
  - 1.6.1 New materials, fabrication methods . . . 13
  - 1.6.2 Creating bio-inspired robots . . . 13
  - 1.6.3 Better power sources . . . 14

1.6.4 Navigating unmapped environments 15
1.6.5 Ethics 16

2 Sensing and Navigation 18
2.1 Sensor Technology 19
2.1.1 Sensor concept 19
2.1.2 Physics of Sensors 20
2.1.3 Sensor Characteristics 20
2.1.4 Classification of Sensors 22
2.1.5 Sensor Function 22
2.1.6 Sensor Types 23
2.1.6.1 Chemical sensors 25
2.1.6.2 Electrical sensors 26
2.1.6.3 Optical sensors 27
2.1.6.4 Camera 28
2.2 Real-Time Obstacle Avoidance Algorithm for Mobile Robots 29
2.2.1 Brief Overview of the Existing Obstacle Avoidance Algorithms 29
2.2.1.1 The Bug Algorithms 29
2.2.1.2 The Potential Field Algorithm 30
2.2.1.3 The Vector Field Histogram (VFH) Algorithm . . . 31
2.2.1.4 The Bubble Band Technique 32
2.2.1.5 Other Obstacle Avoidance Algorithms 32
2.2.2 Description of the Proposed Algorithms 33
2.2.2.1 Detection of Obstacles 33
2.2.2.2 Description of the Algorithm 34

3 Computer Vision and Autonomous Vehicles 37
3.1 Computer vision 38
3.1.1 Object detection 38
3.1.2 Simple detection by colour 38
3.1.3 Introduction of Haar-like features 39
3.1.4 Cascade classifier 40
3.2 Related work 41
3.2.1 PureTech Systems Car Counting 41
3.2.2 Autonomous Real-time Vehicle Detection 41
3.2.3 Monocular Vehicle Detection and Tracking 42
3.3 Object detection using Haar-cascade classifier 43
3.3.1 Hardware 43
3.3.2 Software 43
3.3.2.1 Recorder 44
3.3.2.2 Detector 44
3.3.2.2.1 Background subtraction 44
3.3.2.2.2 Background subtraction augmentation . . 45
3.3.2.3 Regional merging 47

3.3.3 Training cascade . . . 48
3.4 Color detection using HSV method . . . 50
3.4.1 Hue . . . 51
3.4.2 Saturation . . . 51
3.4.3 Value (or Brightness) . . . 51

4 Tools for dynamics simulation of robots 53
4.1 Definition of simulator . . . 55
4.2 The role of simulation . . . 55
4.3 Challenges in simulation . . . 56
4.4 Overiew of simulators . . . 57
4.4.1 Important features for simulation . . . 57
4.4.2 Criteria for choosing a simulator . . . 58
4.4.3 Currently used tools . . . 58
4.4.4 Tools for robots . . . 62
4.5 SOFTWARE INFORMATION FICHES . . . 62
4.5.1 Gazebo . . . 62
4.5.2 ARGoS . . . 63
4.5.3 ODE . . . 64
4.5.4 Bullet . . . 64
4.5.5 V-Rep . . . 65
4.5.6 Webots . . . 65
4.5.7 OpenRave . . . 66
4.5.8 Robotran . . . 66
4.5.9 XDE . . . 67

5 Conception ,Implementation and Development 69
5.1 Environment description . . . 71
5.2 Global architecture . . . 72
5.3 Detailed architecture . . . 73
5.3.1 Use case diagram for autonomous-system of smartCar . . . 73
5.3.2 Sequence Diagrams for autonomous-system of smartCar . . . 75
5.3.2.1 Sequence Diagram : connection scenario . . . 75
5.3.2.2 Sequence Diagram : initialization hardware scenario 76
5.3.2.3 Sequence Diagrams : avoid obstacle scenario . . . 77
5.4 Protocol used . . . 78
5.4.1 I2C Protocols . . . 79
5.4.1.1 Principal . . . 79
5.4.1.2 How I2C works . . . 79
5.4.2 RTSP Protocol . . . 81
5.5 Technologies and structures used . . . 82
5.5.1 Web API . . . 82
5.5.1.1 Definition . . . 82
5.5.1.2 Why is a web API not a web service? . . . 82

5.5.2 Cloud . . . 83
5.6 Setup software and hardware . . . 84
5.6.1 Driver motor . . . 84
5.6.1.1 Driver motor L298N . . . 84
5.6.1.2 Features . . . 85
5.6.1.3 Motor speed control . . . 86
5.6.1.4 Motor direction control . . . 87
5.6.2 Ultrasonic sensor . . . 88
5.6.2.1 Ultrasonic sensor HC-SR04 . . . 88
5.6.2.2 Features . . . 88
5.6.2.3 How Does it Work? . . . 89
5.6.3 LCD . . . 91
5.6.3.1 How connect LCD? . . . 91
5.6.4 Camera . . . 92
5.6.4.1 How can camera detect non-material obstacle? . . . 92
5.7 Global pseudo Algorithm . . . 94
5.8 Global circuit . . . 96
5.9 Environment for Development . . . 97
5.9.1 Environment of Hardware and Software . . . 97
5.9.2 Libraries and languages of programmation . . . 98
5.10 Result . . . 99

Conclusion **104**
1. Contributions . . . 104
2. Perspective . . . 104

Bibliography **106**

List of Figures

1.1 Classification of robots by environment and mechanism of interaction 6
1.2 Classification of robots by application field 7
1.3 Evolution of robotics 8
1.4 Robots on an assembly line in a car factory 9
1.5 Autonomous mobile robot weeding a field 10
1.6 Educational Robots 12
1.7 Convergence of conditions accelerating opportunities for design of bioinspired and biohybrid robots 14
1.8 A summary of different energy sources for robotics. 15
1.9 Ethical and security risks of robotics and AI developments. 16

2.1 Principle of a sensor 20
2.2 Block diagram of a sensor and the graphical representation of its function 21
2.3 Calibration of a sensor and the definition of the accuracy class of a sensor 21
2.4 Classification of sensors 22
2.5 Functional relationships for the static and dynamic behavior of sensors 23
2.6 Classification scheme of sensor types according to their input 24
2.7 Ions, electrons and electric potentials. 27
2.8 Photoelectric effect. 27
2.9 Detect object by camera. 28
2.10 Illustration of the Bug algorithm 29
2.11 Illustration of the Bug2 algorithm 30
2.12 Illustration of the potential field algorithm 30
2.13 The polar histogram used in VFH algorithm 31
2.14 Illustration of the bubble band concept 32
2.15 Robot, ultrasonic sensors, and sensitivity bubble 33
2.16 Polar diagram of the sonar readings 34
2.17 An illustration of the rebound process 35
2.18 Flowchart for the bubble rebound algorithm 35

3.1 total set of features used by the Opencv detection. 39
3.2 Stages of the cascade classifier. 40
3.3 Car candidates detected by Haar-cascade. 42
3.4 Output using MOG with default parameters. 45

3.5 MOG with dilation and erosion. 46
3.6 Regional merge in action. 47
3.7 Depiction of the results of undertrained classifier. 49
3.8 Best solution obtained by the author. 50
3.9 Tracing road by HSV method. 52

4.1 The simulation tools currently used. 59

5.1 Environment of smart car. 72
5.2 Global architecture. 73
5.3 Use case diagram for autonomous-system of smartCar. 74
5.4 Sequence diagram for tasks of connection scenario. 75
5.5 Sequence diagram for tasks of initialization hardware scenario. . . . 76
5.6 Sequence diagram for tasks of avoid obstacle scenario. 78
5.7 I2C wires. 79
5.8 I2C data frame. 80
5.9 I2C communication. 80
5.10 RTSP Protocol. 81
5.11 Web API. 82
5.12 Cloud and Autonomous car. 84
5.13 Driver motor L298N. 85
5.14 Pulse Width Modulation technique. 86
5.15 Work principle of ultrasonic. 89
5.16 Connect the Raspberry Pi with LCD using I2C Adapter 91
5.17 Detect object with using haarcascade. 93
5.18 Detect object with using haarcascade. 93
5.19 Global pseudo Algorithm. 94
5.20 Electronic circuits. 96
5.21 Environment of Hardware and Software. 98
5.22 Prototype of autonomous car. 99
5.23 Environment of autonomous car. 100
5.24 weather notification. 100
5.25 Autonomous car overshoot a car. 101
5.26 Autonomous car detect stop sign. 101
5.27 Result of detection. 102
5.28 Autonomous car try detect sign Maximum speed 80 km/h. 102
5.29 Result of failed detection 103

List of Tables

2.1 Sensor selection criteria 25

3.1 Color degrees 51

4.1 Most important features for a simulator. 58
4.2 Most important criteria for choosing a simulator. 58
4.3 Most diffused tools for a selection of the research areas. 60
4.4 Ratings for the level of user satisfaction of the most diffused tools. . 61

5.1 Motor direction control 87

List of programs

5.1 motor speed python 87
5.2 Motor direction python 87
5.3 Ultrasonic measure python 90
5.4 Ultrasonic measure python 91
5.5 Detect object with using haar-cascade python 92
5.6 Detect color with using HSV python 93

Abbreviations

ISO	**I**nternational **O**rganization for **S**tandardization
STEM	**S**cience, **T**echnology, **E**ngineering, and **M**athematics
LIDAR	**Li**ght **D**etection **A**nd **R**anging
I2C	**I**nter-**I**ntegrated **C**ircuit
EEPROM	**E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory
RTSP	**R**eal **T**ime **S**treaming **P**rotocol
API	**A**pplication **P**rogramming **I**nterface
XML	E**x**tensible **M**arkup **L**anguage
RPC	**R**emote **P**rocedure **C**all
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
CoAP	**Co**ntrained **A**pplication **P**rotocol
RESTful	**R**epresentational **S**tate **T**ransfer
DIY	**D**o **I**t **Y**ourself
LCD	**L**iquid **C**rystal **D**isplay
MOG	**M**ixture **O**f **G**aussians
GPS	**G**lobal **P**ositioning **S**ystem
A/D	**A**nalog/**D**igital
ONVIF	**O**pen **N**etwork **V**ideo **I**nterface **F**orum
PWM	**P**ulse-**W**idth **M**odulation
SNS	**S**imple **N**otification **S**ervice

Symbols

x	input (stimulus).
y	output (response).
Δ	the width of the strip of the calibration diagram is the range of the maximum permissible errors.
ε	the sensitivity of the sensor.
p	The accuracy of a sensor.

Intoduction

1. Preface

By definition, “self-driving robots are those in which operation of the vehicle occurs without direct driver input to control the steering, acceleration, and braking and are designed so that the driver is not expected to constantly monitor the roadway while operating in self-driving mode” . The technology has been advertised and experimented since the 1920s . However, it was no sooner than in the 2010s that autonomous cars were officially introduced to the market . Since then, several major automotive manufacturers have been testing driverless car systems.
Nowadays, robots are designed to perform complicated tasks, such as grasping and manipulating objects, navigating in outdoor environments, and driving in urban roads. Traditionally, robots had to be tediously hand programmed for every task they performed, which is a laborious and time-intensive engineering process. Rather than manually pre-programming each desired behavior, an appropriate robot controller is always preferable that can be derived from observations of a human's own performance.

2. Purposes and goals of the thesis

The overall purpose of the thesis was to identify how autonomous vehicles technology could bring improvement , specifically in navigation on roads.In this context, learning mechanisms are needed to acquire knowledge from such an interaction process.
The main research questions that the thesis aimed to answer were:

- **How are autonomous vehicles more efficient than traditional vehicles ?**
- **How can safety aspect of drivers and vehicles be improved with the advanced technologies of autonomous vehicles?**
- **How are autonomous vehicles learn for avoid matieral obstacles and matieral obstacles?**
- **How to make a self-driving car in a real environment?**

3. Structure of the thesis

for answer on this problematic we organize five chapter are as follows:
chapter I: **Robots and Their Applications**: In this chapter, we introduced and categorized robots, their field in human life, and different challenges waiting for them.
chapter II: **Sensing and Navigation** : In this chapter, we presented various sensors and the principle of their work, and their using for navigation. We also displayed a set of navigation algorithms.
chapter III: **Computer Vision and Autonomous Vehicles:** In the chapter, we presented methods of computer vision , as well as various algorithms in which it operates. And how detect the objects in the picture, and how to learn the machine.
chapter IV: **Tools for dynamics simulation of robots** : In the chapter, we presented tools of simulations and criteria to choose ,and displayed role and importance of simulators.
chapter V: **Conception ,Implementation and Development**:In the chapter, we presented how development a system for autonomous car by the basis of our understanding of the subject while using a various way to build a research project.

Chapter 1

Robots and Their Applications

Introduction
Although everyone seems to know what a robot is, it is hard to give a precise definition. The Oxford English Dictionary gives the following definition:"**A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer"**.This definition includes some interesting elements:

- **"Carrying out actions automatically "**.This is a key element in robotics, but also in many other simpler machines called automata. The difference between a robot and a simple automaton like a dishwasher is in the definition of what a "complex series of actions" is. Is washing clothes composed of a complex series of actions or not? Is flying a plane on autopilot a complex action? Is cooking bread complex? For all these tasks there are machines that are at the boundary between automata and robots.
- is another key element of a robot, because some automata **"Programmable by a computer"** are programmed mechanically and are not very flexible. On the other hand computers are found everywhere, so it is hard to use this criterion to distinguish a robot from another machine.

Question: What is a robot? Types and applications? And its challenges?

1.1 Definition of robot

A standard definition of a robot is that a robot is a mechanical or virtual agent, usually an electromechanical machine guided by a computer program or electronic circuitry. Man has to define robots in many different ways. Another definition, from 1993 according to Standard EN 775, defines the robot as "automatically controlled, re-programmable, multi-purpose processing device, which has several degrees of freedom, and which may be either stationary or mounted to be movable, for the use in industrial automation systems". The International Robot Association, in turn, defines the robot, in accordance with ISO 8373: 1994, as an "automatically controlled reprogrammable, multifunctional manipulator with at least three programmable axes." The term "robot" refers to forced labour, to a human unilaterally controlled to work according to rules [1]. it usually means a mechanical device or a machine that knows how to act in certain ways in the physical world. Thus, humans have initially forced robots to forced labour, to slavery. Man has forced the robots also to thinking, to the use of artificial intelligence.

1.2 Classification of Robots

Robots can be classified according to the environment in which they operate (fig.1.1). The most common distinction is between fixed and mobile robots. These two types of robots have very different working environments and therefore require very different capabilities. Fixed robots are mostly industrial robotic manipulators that work in well defined environments adapted for robots. Industrial robots perform specific repetitive tasks such soldering or painting parts in car manufacturing plants. With the improvement of sensors and devices for human-robot interaction, robotic manipulators are increasingly used in less controlled environment such as high-precision surgery[2].

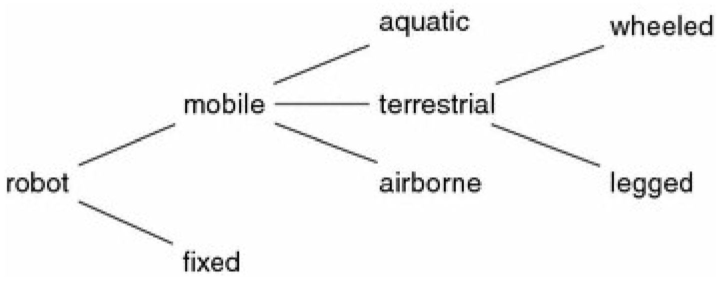


Figure 1.1: Classification of robots by environment and mechanism of interaction [2]

By contrast, mobile robots are expected to move around and perform tasks in large, ill-defined and uncertain environments that are not designed specifically for robots. They need to deal with situations that are not precisely known in advance and that change over time. Such environments can include unpredictable entities like humans and animals. Examples of mobile robots are robotic vacuum cleaners and self-driving cars.

There is no clear dividing line between the tasks carried out by fixed robots and mobile robots—humans may interact with industrial robots and mobile robots can be constrained to move on tracks—but it is convenient to consider the two classes as fundamentally different. In particular, fixed robots are attached to a stable mount on the ground, so they can compute their position based on their internal state, while mobile robots need to rely on their perception of the environment in order to compute their location[2].

There are three main environments for mobile robots that require significantly different design principles because they differ in the mechanism of motion: aquatic (underwater exploration), terrestrial (cars) and aerial (drones). Again, the classification is not strict, for example, there are amphibious robots that move in both water and on the ground. Robots for these three environments can be further divided into subclasses: terrestrial robots can have legs or wheels or tracks, and aerial robots can be lighter-than-air balloons or heavier-than-air aircraft, which are in turn divided into fixed-wing and rotary-wing (helicopters).

Robots can be classified by intended application field and the tasks they perform (fig 1.2) mentioned industrial robots which work in well-defined environments.

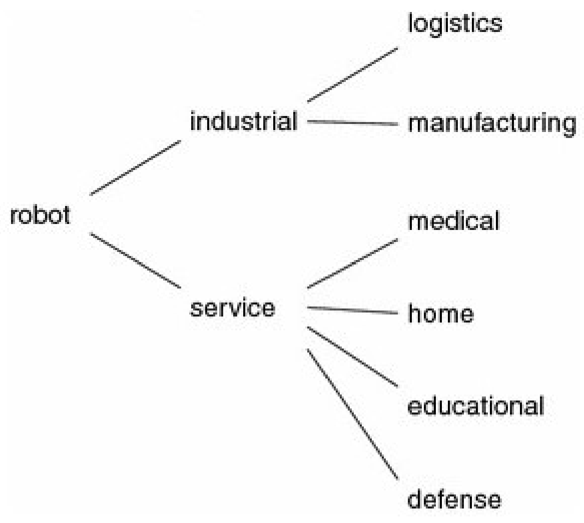


FIGURE 1.2: Classification of robots by application field [2]

1.3 Framework of roadmap of robotics

Between the 1960s and the 1990s, most robots and robotics in general were related to industrial applications. The key challenge was to rationalize production at manufacturing sites. Now robots are becoming ubiquitous, reaching exceptional capabilities and robustness. Service robots support, accompany and nurse humans. Robots will be helpers in healthcare and personal life. Service industries develop many new service robot applications. Service robots share the human environment and exhibit basic intelligent behavior to accomplish assigned tasks. We can expect that degree of autonomy and system complexity along with human-centered applications [3].

In the future service robotics will play a bigger role. Service robotics is an emerging application for human-centered technologies and the service economy. Recent studies imply that the rise of household and personal assistance robots forecast a human-robot collaborative society. In(fig.1.3), a 3-phase model of robotics is presented. This model illustrates key phases of robotics. In (fig1.3) Current trends are leading towards more complex, more personalized and autonomous systems and robot services. This implies flexible systems that are able to perform tasks in an unconstrained, human centered environment. [4]

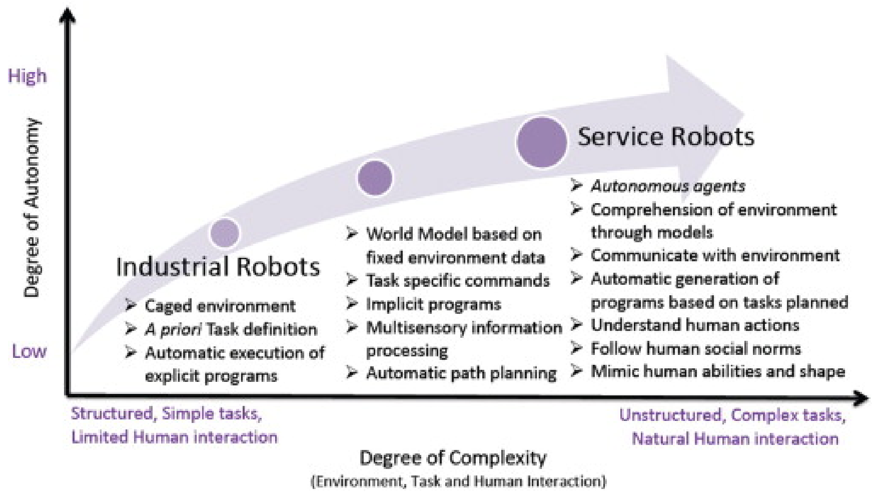

High
Low
Degree of Autonomy
Industrial Robots
➢ Caged environment
➢ *A priori* Task definition
➢ Automatic execution of explicit programs
➢ World Model based on fixed environment data
➢ Task specific commands
➢ Implicit programs
➢ Multisensory information processing
➢ Automatic path planning
Service Robots
➢ *Autonomous agents*
➢ Comprehension of environment through models
➢ Communicate with environment
➢ Automatic generation of programs based on tasks planned
➢ Understand human actions
➢ Follow human social norms
➢ Mimic human abilities and shape
Structured, Simple tasks, Limited Human interaction
Unstructured, Complex tasks, Natural Human interaction
Degree of Complexity
(Environment, Task and Human Interaction)

FIGURE 1.3: Evolution of robotics [1]

1.4 Robotics Applications

currently, robots perform a number of different jobs in numerous fields and the amount of tasks delegated to robots is rising progressively. The best way to split robots into types is a partition by their application.

1.4.1 Industrial Robots

The first robots were industrial robots which replaced human workers performing simple repetitive tasks. Factory assembly lines can operate without the presence of humans, in a well-defined environment where the robot has to perform tasks in a specified order, acting on objects precisely placed in front of it(fig1.4).
One could argue that these are really automata and not robots. However, today's automata often rely on sensors to the extent that they can be considered as robots. However, their design is simplified because they work in a customized environment which humans are not allowed to access while the robot is working.
However, today's robots need more flexibility, for example, the ability to manipulate objects in different orientations or to recognize different objects that need to be packaged in the right order. The robot can be required to transport goods to and from warehouses. This brings additional autonomy, but the basic characteristic remains: the environment is more-or-less constrained and can be adapted to the robot[2].
Additional flexibility is required when industrial robots interact with humans and

this introduces strong safety requirements, both for robotic arms and for mobile robots. In particular, the speed of the robot must be reduced and the mechanical design must ensure that moving parts are not a danger to the user. The advantage of humans working with robots is that each can perform what they do best: the robots perform repetitive or dangerous tasks, while humans perform more complex steps and define the overall tasks of the robot, since they are quick to recognize errors and opportunities for optimization.

FIGURE 1.4: Robots on an assembly line in a car factory

1.4.2 Autonomous Mobile Robots

Many mobile robots are remotely controlled, performing tasks such as pipe inspection, aerial photography and bomb disposal that rely on an operator controlling the device. These robots are not autonomous; they use their sensors to give their operator remote access to dangerous, distant or inaccessible places. Some of them can be semi-autonomous, performing subtasks automatically. The autopilot of a drone stabilizes the flight while the human chooses the flight path. A robot in a pipe can control its movement inside the pipe while the human searches for defects that need to be repaired. Fully autonomous mobile robots do not rely on an operator, but instead they make decisions on their own and perform tasks, such as transporting material while navigating in uncertain terrain (walls and doors within buildings, intersections on streets) and in a constantly changing environment (people walking around, cars moving on the streets)[2].
The first mobile robots were designed for simple environments, for example, robots that cleaned swimming pools or robotic lawn mowers. Currently, robotic vacuum cleaners are widely available, because it has proved feasible to build reasonably priced robots that can navigate an indoor environment cluttered with obstacles.

Many autonomous mobile robots are designed to support professionals working in structured environments such as warehouses. An interesting example is a robot for weeding fields (fig.1.5) . This environment is partially structured, but advanced sensing is required to perform the tasks of identifying and removing weeds. Even in very structured factories, robot share the environment with humans and therefore their sensing must be extremely reliable.
Perhaps the autonomous mobile robot getting the most publicity these days is the self-driving car. These are extremely difficult to develop because of the highly complex uncertain environment of motorized traffic and the strict safety requirements.

An even more difficult and dangerous environment is space. The Sojourner and

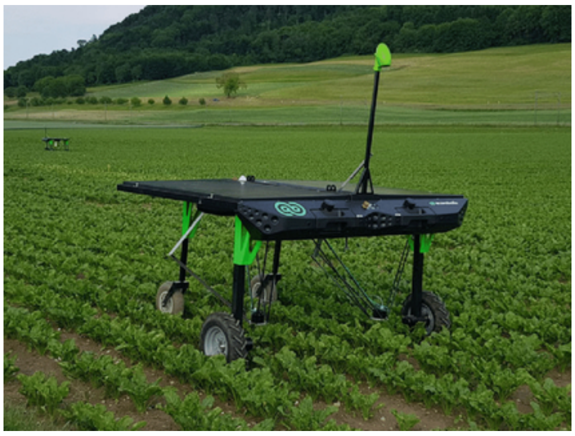


FIGURE 1.5: Autonomous mobile robot weeding a field [2]

Curiosity Mars rovers are semi-autonomous mobile robots. The Sojourner was active for three months in 1997. The Curiosity has been active since landing on Mars in 2012! While a human driver on Earth controls the missions (the routes to drive and the scientific experiments to be conducted), the rovers do have the capability of autonomous hazard avoidance.
Much of the research and development in robotics today is focused on making robots more autonomous by improving sensors and enabling more intelligent control of the robot.
Better sensors can perceive the details of more complex situations, but to handle these situations, control of the behavior of the robot must be very flexible and adaptable. Vision, in particular, is a very active field of research because cameras are cheap and the information they can acquire is very rich. Efforts are being made to make systems more flexible, so that they can learn from a human

or adapt to new situations. Another active field of research addresses the interaction between humans and robots. This involves both sensing and intelligence, but it must also take into account the psychology and sociology of the interactions.

1.4.3 Educational Robots

Advances in the electronics and mechanics have made it possible to construct robots that are relatively inexpensive. Educational robots are used extensively in schools, both in classrooms and in extracurricular activities. The large number of educational robots makes it impossible to give a complete overview. Here we give few examples that are representative of robots commonly used in education.

- **Pre-Assembled Mobile Robots**

Many educational robots are designed as pre-assembled mobile robots. fig.1.6a shows the Thymio robot from Mobsya and fig.1.6b shows the Dash robot from-Wonder Workshop. These robots are relatively inexpensive, robust and contain a large number of sensors and output components such as lights. An important advantage of these robots is that you can implement robotic algorithms "out of the box," without investing hours in mechanical design and construction. However, pre-assembled robots cannot be modified, though many do support building extensions using, for example, LEGO® components .

you can design and build a robot to perform a specific task, limited only by your imagination. A robotics kit can also be used to teach students mechanical design. The disadvantages of robotics kits are that they are more expensive than simple pre-assembled robots and that exploration of robotics algorithms depends on the one's ability to successfully implement a robust mechanical design.

A recent trend is to replace fixed collections of bricks by parts constructed by 3D printers. An example is the Poppy Ergo Jr robotic arm (fig.1.6b). The use of 3D printed parts allows more flexibility in the creation of the mechanical structure and greater robustness, but does require access to a 3D printer.

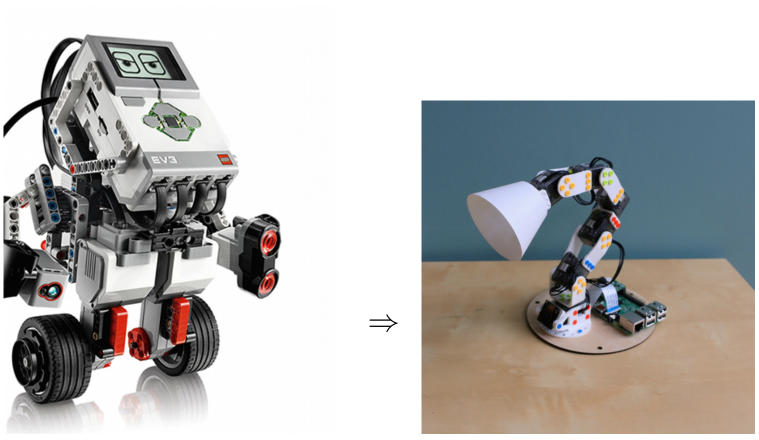

(A) LEGO ® Mindstorms EV3 (Courtesy of Adi Shmorak, Intelitek) ⇒ (B) Poppy Ergo Jr robotic arms (Courtesy of the Poppy Project)

FIGURE 1.6: Educational Robots

1.5 Welfare benefits and advantages of robotics

Robots can free people to focus on the creative process by taking care of unpleasant physical and mechanical work. The greatest benefits of robotics should be meant for people working in unhealthy environments, such as mines and deep waters. Using robots, industrial production could be maintained in countries with high labour costs – especially for small scale batch production. The third domain for robots would be confined to productive activities and tasks that a man cannot perform. Robots planned to analyse, audit and edit massive data are in the business interest of companies and private experts [1].
The majority of data collected in the world has been gathered in recent years; approximately 90 per cent of it all in the past two years. Thus, the "Big Data" are real-time data and up-to-date; in their analysis, attention is paid to the volume of data stream, variances, and to the velocity of data. When big data is analyzed on the basis of these criteria, businesses and other stakeholders will identify new opportunities in crowds, markets and networks. Crowdsourcing, the practice of obtaining needed ideas, services, or strategic content by soliciting contributions from a large group of people, and especially from an online community, rather than from traditional employees or suppliers, is gaining importance in various business fields. Applying crowdsourcing techniques of BigData will be a key issue for companies and corporations.

1.6 Biggest Challenges in Robotics

There are a number of challenges we will need to overcome first, technically, socially and ethically.
five of these challenges are technologies that will have an impact on all applications of robotics – including creating new materials and manufacturing methods, developing cost-effective and long-lasting power sources for mobile robots, and programming artificial intelligence to perform deep moral and social reasoning about real-world problems.

1.6.1 New materials, fabrication methods

Gears, motors, and actuators are fundamental to today's robots. However, tremendous work is already being done with artificial muscles, soft robotics, and assembly strategies that will help develop the next generation of autonomous robots that are multifunctional and power-efficient[5].

1.6.2 Creating bio-inspired robots

Robots inspired by nature are becoming more common in robotics labs. The main idea is to create robots that perform more like the efficient systems found in nature. But the study says the major challenges involved with this area have remained largely unchanged for 30 years – a battery to match metabolic conversion, muscle-like actuators, self-healing material, autonomy in any environment, human-like perception, and computation and reasoning[6].
Materials that couple sensing, actuation, computation, and communication must be developed and shared before this segment takes off. These advances could lead to robots with features such as body support, weight reduction, impact protection, morphological computation, and mobility.

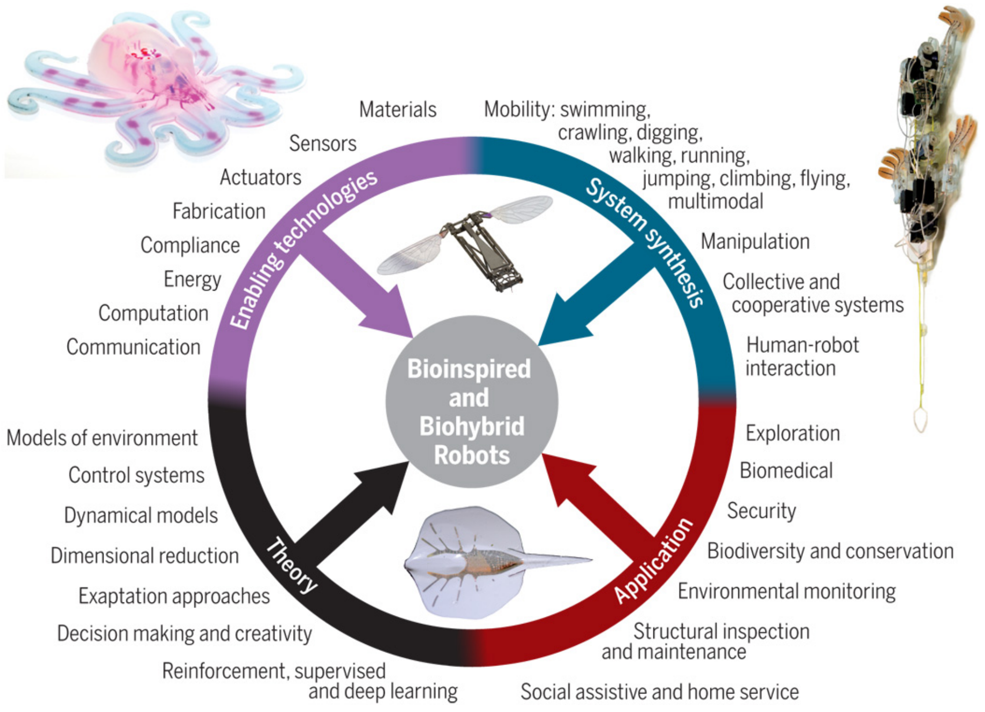

Bioinspired and Biohybrid Robots
Enabling technologies
Materials
Sensors
Actuators
Fabrication
Compliance
Energy
Computation
Communication
System synthesis
Mobility: swimming, crawling, digging, walking, running, jumping, climbing, flying, multimodal
Manipulation
Collective and cooperative systems
Human-robot interaction
Application
Exploration
Biomedical
Security
Biodiversity and conservation
Environmental monitoring
Structural inspection and maintenance
Social assistive and home service
Theory
Models of environment
Control systems
Dynamical models
Dimensional reduction
Exaptation approaches
Decision making and creativity
Reinforcement, supervised and deep learning

FIGURE 1.7: Convergence of conditions accelerating opportunities for design of bioinspired and biohybrid robots

1.6.3 Better power sources

Robots, typically, are energy-inefficient. Improving the battery life is a major issue, especially for drones and mobile robots. Thankfully, increased adoption of these systems is leading to new battery technologies that are affordable, safe and longer-lasting[7].

Work is certainly being done to make the components of a robot more power efficient. But the study mentions robots that need to operate wirelessly in unstructured environments will eventually extract energy from light, vibrations, and mechanical movement.

Research is also being done to improve battery technology beyond the nickel-metal hydride and lithium ion options currently available.

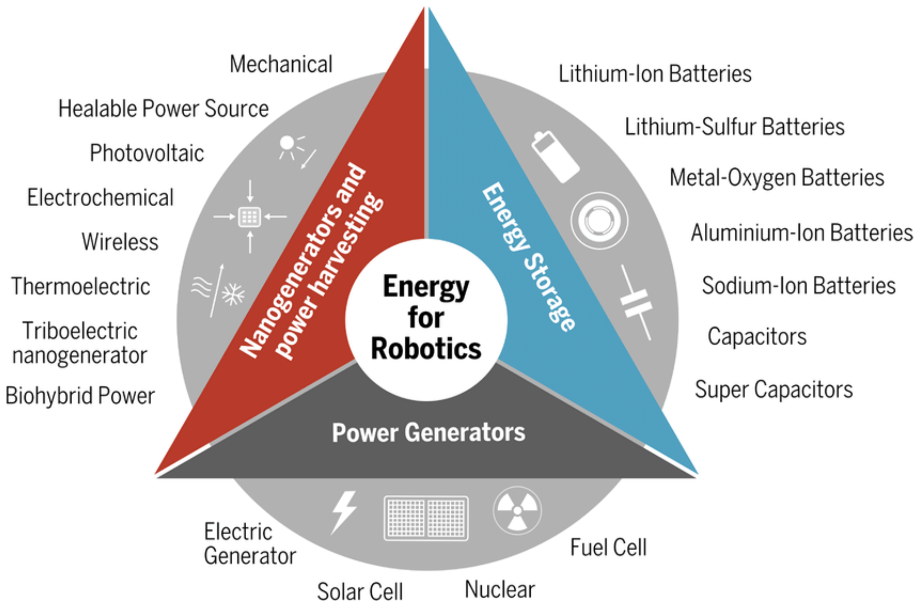


FIGURE 1.8: A summary of different energy sources for robotics.[5]

1.6.4 Navigating unmapped environments

There has been significant progress made when it comes to robots perceiving and navigating their environments. Just look at self-driving cars, for example. Mapping and navigation techniques will continue to evolve, but future robots need to be able to operate in environments that are unmapped and poorly understood. Some of the improvement that need to be made include[8]:

- How to learn, forget, and associate memories of scenes both qualitatively and semantically.
- How to surpass purely geometric maps to have semantic understanding of the scene.
- How to reason about new concepts and their semantic representations and discover new objects or classes in the environment through learning and active interactions.

"For navigation, the grand challenge is to handle failures and being able to adapt, learn, and recover. For exploration, it is developing the innate abilities to make and recognize new discoveries", the study says. "From a system perspective, this requires the physical robustness to withstand harsh, changeable environments, rough handling, and complex manipulation[5]. The robots need to have significant levels of autonomy leading to complex self-monitoring, self-reconfiguration, and repair such that there is no single point of complete failure but rather graceful system degradation. When possible, solutions need to involve control of multiple heterogeneous robots; adaptively coordinate, interface, and use multiple assets; and share information from multiple data sources of variable reliability and accuracy".

1.6.5 Ethics

Stop me if you've heard concerns about robot ethics before. All kidding aside, it's a major challenge, and the robotics industry is well aware of. The study breaks down ethical problems into five topics[9]:

1. Sensitive tasks that should require human supervision could be delegated entirely to robots[10].
2. Humans will no longer take responsibility for failures.
3. Unemployment and de-skilling of the workforce.
4. AI could erode human freedom[11].
5. Using AI in unethical ways.

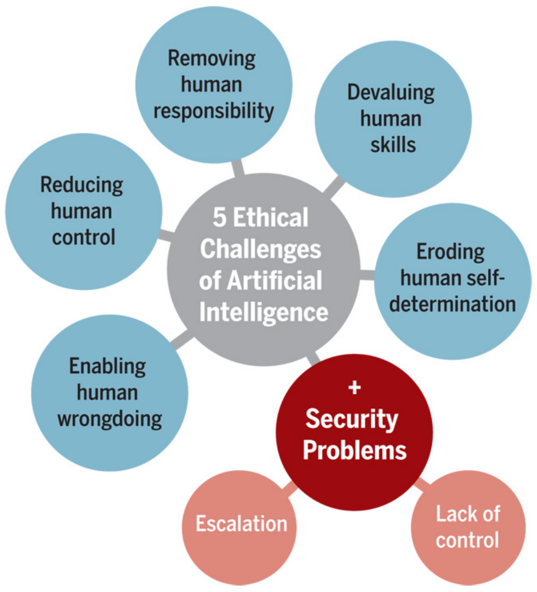

Removing human responsibility
Devaluing human skills
Reducing human control
5 Ethical Challenges of Artificial Intelligence
Eroding human self-determination
Enabling human wrongdoing
+ Security Problems
Escalation
Lack of control

FIGURE 1.9: Ethical and security risks of robotics and AI developments.[5]

Conclusion

Robots are found everywhere: in factories, homes and hospitals, and even in outer space. Much research and development is being invested in developing robots that interact with humans directly .Robots are used in schools in order to increase students' motivation to study STEM (science, technology, engineering, and mathematics) and as a pedagogical tool to teach STEM in a concrete environment. The focus of this chapter is the definition of robots, their classification and theirs applications.

Chapter 2

Sensing and Navigation

Introduction

In general a fully autonomous robot should be able to gain information about the surrounding environment, work for an extended period of time without human intervention and possibly avoid situations that are harmful to people, property, or itself. An autonomous robot should also be able to learn or gain new knowledge to adapt to surroundings changes. To fulfill all these capabilities, the robot requires several proprioceptive and exteroceptive sensors.

Proprioception, or sensing one's own internal status, is required for robots to work autonomously near people and in harsh environments. Common proprioceptive sensors include thermal, optical, electrical and haptic sensing.

Exteroception is sensing things about the environment. Autonomous robots must have a range of environmental sensors to perform their task while keeping an adequate level of knowledge and safety. Common exteroceptive sensors include the electromagnetic, sound, touch, chemical, temperature, range, pressure, and altitude sensors.

All these sensors cannot work indipendently but their measures should be fused to obtain good estimation of environmental and internal variables.

2.1 Sensor Technology

Sensors are measurement transducers. They transform parameters of interest into electrical signals that can be displayed, stored, or further processed for applications in science and technology. Based on a view on the physics of sensors, the characteristics of sensors are described, including the sensing measuring chain, signal processing, and data acquisition[12].

This section gives a survey of sensor types for dimensional metrology, kinematics, kinetics, and temperature. It describes also microsensors and embedded sensors.

2.1.1 Sensor concept

One can define a sensor as a device/component capable of quantifying a physical quantity. However, one can still state that this is a barely rudimentary definition of a sensor, without any technical insight. Thus, another possible definition would be to consider a sensor as a component capable of detecting a given variation in input energy and converting it to another or the same type of energy as output [13] . This definition matches the transducer definition as a component capable of translating a form of energy into another form. It is with this scenario that both concepts become blurry.

2.1.2 Physics of Sensors

The design of sensors is based on physical effects for the transformation of parameters of interest into electrical signals [14] . The principle of a sensor is depicted in fig.2.1

There is a broad variety of physical principles that can be used for sensing parameters of interest. The following brief overview denotes examples of physical effect for sensors to detect and to measure dimensional, kinematic, dynamic, and thermal parameters.

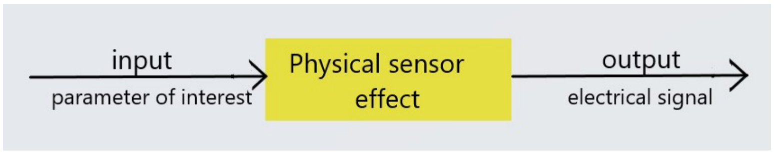


FIGURE 2.1: Principle of a sensor [14]

2.1.3 Sensor Characteristics

A sensor can be illustrated by a block diagram and described by the sensor function $y = f(x)$ which relates the output y to the input x.see fig.2.2.If there is a linear input–output relation, the sensitivity ε of the sensor is defined as $\varepsilon = \Delta y/\Delta x$ and its reciprocal is the input/output coefficient $c = \Delta x/\Delta y$. From the measurement of y, the parameter of interest x can be determined through the relation $x = c * y$ [12] .

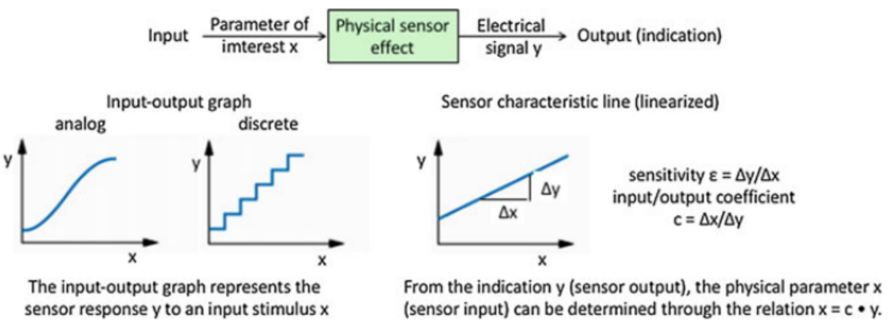

Input
Parameter of imterest x
Physical sensor effect
Electrical signal y
Output (indication)
Input-output graph
analog
discrete
Sensor characteristic line (linearized)
sensitivity ε = Δy/Δx
input/output coefficient
c = Δx/Δy
The input-output graph represents the sensor response y to an input stimulus x
From the indication y (sensor output), the physical parameter x (sensor input) can be determined through the relation x = c • y.

FIGURE 2.2: Block diagram of a sensor and the graphical representation of its function[14]

The sensor line must be determined by the calibration of the sensor(see fig.2.3). A calibration diagram represents the relation between indications of a sensor and

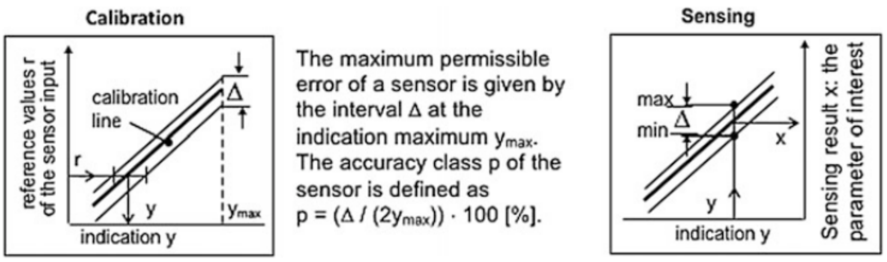

Calibration
reference values r of the sensor input
calibration line
indication y
The maximum permissible error of a sensor is given by the interval Δ at the indication maximum y_max.
The accuracy class p of the sensor is defined as
p = (Δ / (2y_max)) · 100 [%].
Sensing
max
min
indication y
Sensing result x: the parameter of interest

FIGURE 2.3: Calibration of a sensor and the definition of the accuracy class of a sensor[14]

a set of reference values of the measurand. At the maximum indication y_{max} , the width Δ of the strip of the calibration diagram is the range of the maximum permissible errors. The accuracy class p of a sensor is defined as:

$$p = (\Delta/(2ymax)) * 100\% \tag{2.1}$$

In summary, the basic technological features of sensors are:

- sensor principle.
- input x (stimulus), measurement range, accuracy class of sensor.
- output y (response), analog display, digital display.
- sensor line, i.e., the $x - y$ calibration graph.
- signal transmission (stationary, dynamic).
- data acquisition architecture.

- sensor data communication.

In addition, sensors have to comply with the general requirements for all engineered products including quality, safety, reliability, environmental compatibility, and recyclability. For the application of sensors, the production technology and costs are also important economic factors.

2.1.4 Classification of Sensors

Sensors are classified as proprioceptive or exteroceptive, and exteroceptive sensors are further classified as active or passive (Fig.2.4).Aproprioceptive sensor measures something internal to the robot itself. An exteroceptive sensor measures something external to the robot such as the distance to an object. An active sensor affects the environment usually by emitting energy: a sonar range finder on a submarine emits sound waves and uses the reflected sound to determine range[2]. A passive sensor does not affect the environment: a camera simply records the light reflected off an object. Robots invariably use some exteroceptive sensors to correct for errors that might arise from proprioceptive sensors or to take changes of the environment into account.

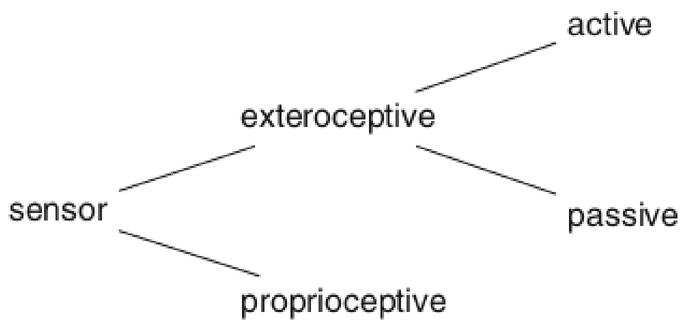


FIGURE 2.4: Classification of sensors[2]

2.1.5 Sensor Function

The function of a sensor is to transform a parameter of interest into an electrical signal[12]. For every sensor exists an input–output (stimulus–response) relationship. The basic relationships for the static and dynamic behavior of sensors are depicted in fig.2.5.

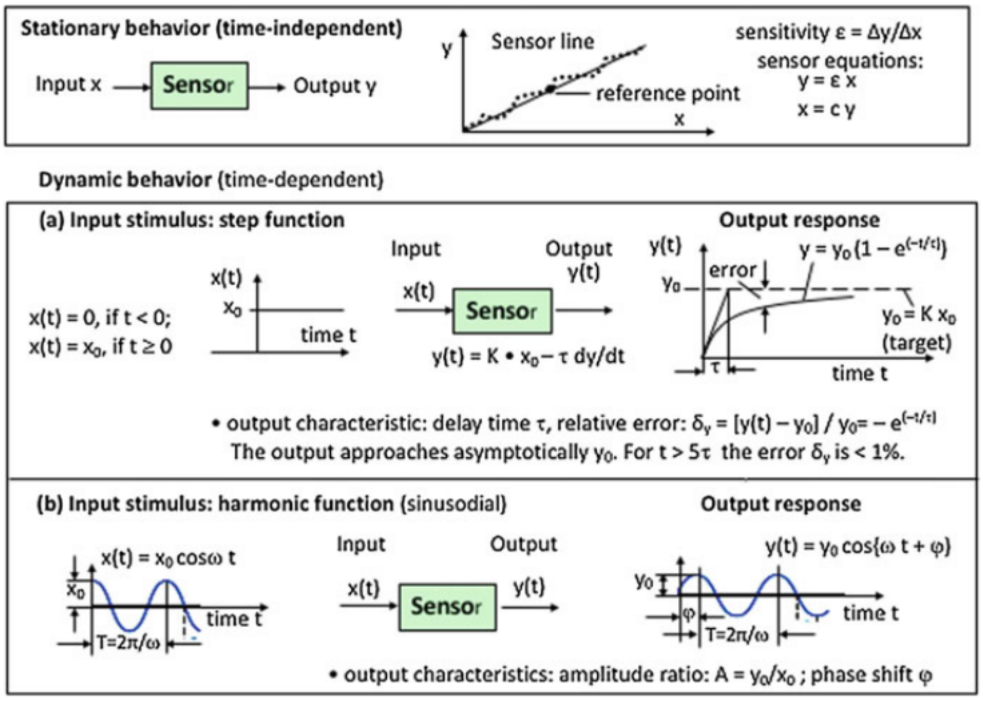

Stationary behavior (time-independent)
Input x
Sensor
Output y
y
Sensor line
reference point
x
sensitivity ε = Δy/Δx
sensor equations:
y = ε x
x = c y
Dynamic behavior (time-dependent)
(a) Input stimulus: step function
x(t) = 0, if t < 0;
x(t) = x₀, if t ≥ 0
x(t)
x₀
time t
Input
x(t)
Sensor
Output
y(t)
y(t) = K • x₀ − τ dy/dt
Output response
y(t)
y₀
error
y = y₀(1 − e^(−t/τ))
y₀ = K x₀
(target)
τ
time t
• output characteristic: delay time τ, relative error: δy = [y(t) − y₀] / y₀ = − e^(−t/τ)
The output approaches asymptotically y₀. For t > 5τ the error δy is < 1%.
(b) Input stimulus: harmonic function (sinusodial)
x(t) = x₀ cosω t
x₀
time t
T=2π/ω
Input
x(t)
Sensor
Output
y(t)
Output response
y(t) = y₀ cos(ω t + φ)
y₀
φ
T=2π/ω
time t
• output characteristics: amplitude ratio: A = y₀/x₀ ; phase shift φ

FIGURE 2.5: Functional relationships for the static and dynamic behavior of sensors[12]

The stationary (time-independent) behavior is described by a linear input–output function and graphically expressed as sensor line. Such sensor is called in the terminology of control theory a "zero-order sensor".
The dynamic (time-dependent) behavior of a sensor is conventionally characterized by two different input test functions:

(a) a step function input x gives an output response y in the form of a first-order differential equation. Such sensor is called in the terminology of control theory a "first-order sensor". It is characterized by a delay time constant s which is a measure of the sensor's inertia.

(b) harmonic undamped sinusoidal input x gives a sinusoidal output y with an amplitude ratio y_0/x_0 and a phase shift of the output signal. Such sensor is called in the terminology of control theory a "second-order sensor"[12].

2.1.6 Sensor Types

The broad variety of sensors, which have been developed for the transformation of non-electrical parameters of interest into electrical signals, can be classified according to the physical nature of their input parameters. An overview of sensor categories is given in fig.2.6. The first step in selecting a sensor type for a particular

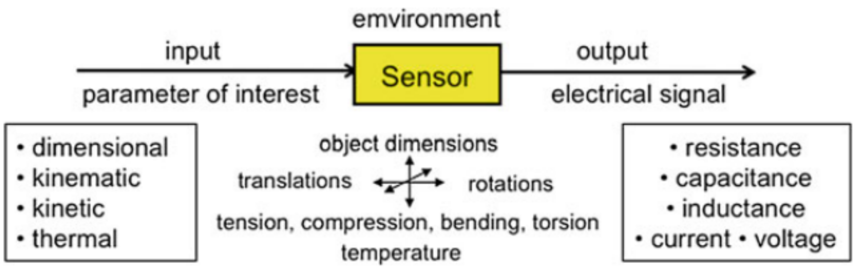


FIGURE 2.6: Classification scheme of sensor types according to their input[12]

application is to specify the parameter of interest or measurand, i.e., the input of the sensor and the environment in which they will operate. Sensor selection has to consider[12]:

- The physical quantities (measurands) to be sensed.
- Nominal values and the expected ranges of the measurands.
- Spatial and temporal properties of the measurands.
- Accuracy required for each measurement.
- Duration of sensing of the expected sensor life time.
- Environmental conditions in which the measurements will be made.

After the measurands and the environment have been characterized, a collection of possible sensors can be assorted. The criteria to find the optimal sensors can be assigned to three primary categories:

- Sensor performance characteristics.
- Environmental constraints.
- Economic considerations.

Some important aspects of sensor application are listed in table 2.1 . A validation of the individual components and the whole sensing module under controlled and field conditions are required before the system goes under operation[2].

Removal of undesired features from data, such as trends (e.g., drift of a sensor) and noise, is necessary. Calibration of the individual sensor and the data acquisition components has to be performed; see Fig.2.6. The response of the integrated sensory data acquisition system should also be calibrated. To increase the reliability of sensing modules, redundant instrumentation by sensors of different physical conversion principle may be considered for long-term surveillance.

Sensor performance characteristics	**Environmental constraints**	**Economic considerations**
Range	Temperature range	Cost
Linearity	Humidity range	Availability
Sensitivity	Thermal effects	Reliability
Resolution	Size	Ease of installation
Response time	Packaging	Data acquisition needs
Precision	Need of isolation from	
Hysteresis	disturbances	
Stability		

Table 2.1: Sensor selection criteria[12]

2.1.6.1 Chemical sensors

Chemical sensors respond to measurands through various chemicals and chemical reactions. They have the ability to identify and quantify liquid or gaseous chemical species. This class of sensors is currently used in many chemical analysis techniques, such as mass spectroscopy, chromatography, infrared technology and others [15], due to the miniaturization of sensors. There are a variety of chemical sensors with various methods of transduction. The transduction methods of chemical sensors can be organized into three classes based on their modes of measurement:

(i) electrical and electrochemical properties.

(ii) changes in the physical properties.

(iii) optical absorption of the chemical analytes to be measured.

Electrochemical sensors are the most common approach, and operate by reacting with the chemical solutions and producing an electrical signal that is proportional to the analyte concentration [16]. These types of sensors can be classified into three categories according to their operation mode [15].

- Potentiometric, which relies on the measurement of voltage. To measure the concentration of analytes it is necessary for a combination of electric and ionic current to flow in a closed circuit. A practical application of this sensing principle will be given in the biosensors section.
- Amperometric, which relies on the measurement of current. These sensors have been shown to be effective in a broad range of applications, such as volatile organic compounds detection in soils and groundwater, detection of mines and analytes detection in blood [17].
- Conductometric relies on the measurement of either conductivity or resistivity. This type of sensor comprises a capacitor that changes its capacitance when exposed to the desired analyte. The capacitance of the sensor changes due to a selectively absorbing material such as polymers or other insulators. These absorbing materials serve as the dielectric layer of the capacitor and their permittivity changes with exposure to the analyte. These sensors are commonly used to detect humidity as well as carbon dioxide and volatile organic compounds. In the humidity sensor case, the dielectric layer comprises a water-sensitive polymer.

2.1.6.2 Electrical sensors

Electrical sensors have the ability to transform electric potentials, fields and charges into electrical signals. Since the source measurand is already of electrical nature, these sensors use the direct sensing principle[12].
A commonly used example is that of the physiological electrodes. These devices can convert electrical fields and potentials generated by the movement and concentration of ions in biological systems into electric signals that can be acquired by electronic instrumentation. The physical principle behind the functioning of such electrodes is the electric potential generated by a point charge. A charged particle creates an electric field, and therefore an electric potential in space.
Consider the example of a group of ions with a positive net charge moving in the vicinity of a metallic electrode. The movement of the ions will change the electric field and therefore, the electric potential felt by the nearby electrode. If there is a second electrode with a reference potential, the ions will cause a variation in the potential of the first electrode, generating an electrical signal that can be measured by external electronics (fig.2.7).

This kind of electrodes is used to measure many sources of electrical signals in

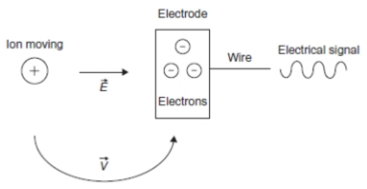


FIGURE 2.7: Ions, electrons and electric potentials.

the body, such as the heart electrical signal. In this example the myocardial depolarization and repolarization of heart cells can be measured through a system called electrocardiogram (ECG) with electrodes placed on the patient's skin [18]

2.1.6.3 Optical sensors

Optical sensors have the capacity to detect and quantify various properties of light such as intensity, frequency, wavelength, and polarization. These sensors rely on light detectors that have the ability of converting light into electrical signals. These detectors are able to sense electromagnetic radiation from the infrared up to the ultraviolet wavelengths.
Many light detectors such as the photovoltaic and photoconductive, depend on the interaction between photons and the crystalline lattice of semiconductors. This interaction is known as the photoelectric effect, and was firstly proposed by Einstein in 1905. The photoelectric effect explains that when a photon hits the surface of a conductor it generates a free electron (Fig.2.8). These free electrons will affect the material's electrical properties, therefore allowing the measurement of the properties of the incident light.
A widely used tool for optical sensors is the photodiode. This device is capable

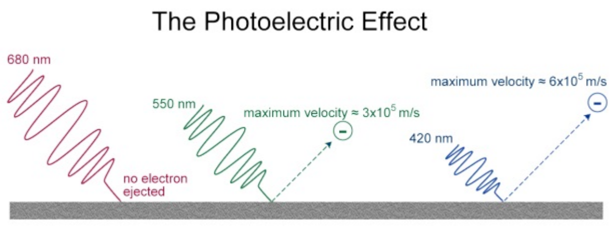

The Photoelectric Effect
680 nm
no electron ejected
550 nm
maximum velocity ≈ 3x10^5 m/s
420 nm
maximum velocity ≈ 6x10^5 m/s

FIGURE 2.8: Photoelectric effect.

of converting light into electric current and thus being able to sense light intensity. This device is composed of a p- and n-type semiconductor bonded together forming a diode. These devices are similar to semiconductor diodes, except that

their packaging allows light to hit the pn-junction. When reversely biased, as light hits the semiconductor, free electrons will be generated and the current flowing in the circuit will increase quite noticeably. Even when there is no light, the circuit will generate a small current known as dark current. This current is independent of applied voltage and varies only with temperature. On the other hand, when light is shining over the semiconductor a significantly higher current will flow through the circuit and is generated by the photocurrent.
One example of an optical sensor used for medical applications is the pulse oximeter. This device is placed in a thin part of the patient's body such as a fingertip or earlobe. Two lights with different wavelengths are passed through the patient's body and absorbed in a photodetector on the other side. This system is able to detect the saturation of oxygen in the blood[19].

2.1.6.4 Camera

The fundamental issue with cameras in robotics is that we are not interested in an array of "raw" pixels, but in identifying the objects that are in the image. The human eye and brain instantly perform recognition tasks: when driving a car we identify other cars, pedestrians, traffic lights and obstacles in the road automatically, and take the appropriate actions. Image processing by a computer requires sophisticated algorithms and significant processing power. For that reason, robots with cameras are much more complex and expensive than educational robots that use proximity sensors.[2]

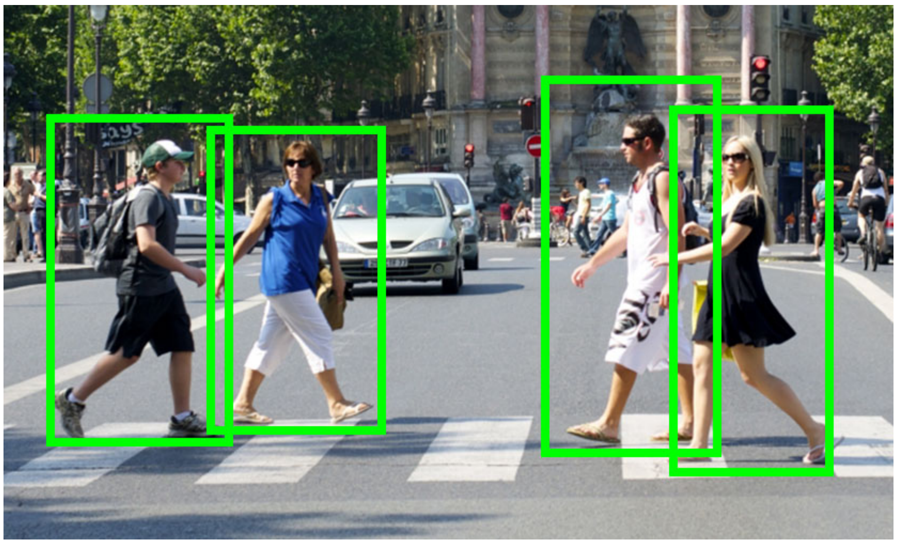


FIGURE 2.9: Detect object by camera.

2.2 Real-Time Obstacle Avoidance Algorithm for Mobile Robots

The ability to detect and avoid obstacles in real time is an important design requirement for any practical application of autonomous vehicles. Therefore, a significant number of solutions have been proposed for this problem. Unfortunately, most of these solutions demand a heavy computational load, which makes them difficult, if not impossible, to implement on low cost, microcontroller based, control structures[20].
This part presents the results of a research aimed to develop a new algorithm for obstacle avoidance relying on low cost ultrasonic or infrared sensors, and involving a reasonable level of calculations, so that it can be easily used in real time control applications with microcontrollers.

2.2.1 Brief Overview of the Existing Obstacle Avoidance Algorithms

We will explain different algorithms for avoid obstacle in different cases:

2.2.1.1 The Bug Algorithms

The simplest obstacle avoidance algorithm ever described is called "the bug algorithm"[21]. According to it, when an obstacle is encountered, the robot fully circles the object in order to find the point with the shortest distance to the goal, then leaves the boundary of the obstacle from this point (see fig 2.10).
This algorithm is obviously very inefficient, and therefore several improvements

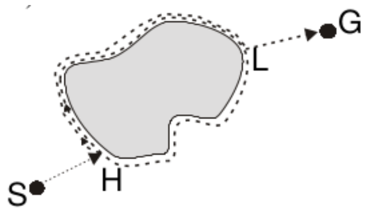


FIGURE 2.10: Illustration of the Bug algorithm

have been proposed. In the 'bug2" algorithm (see fig 2.11), the robot starts following the boundary of the obstacle, but leaves it as soon as it intersects the line segment that connects the start point and the goal. Although their simplicity is a

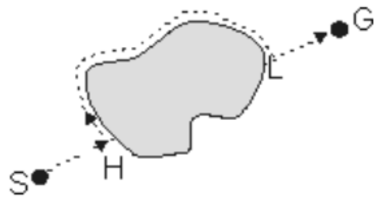


FIGURE 2.11: Illustration of the Bug2 algorithm

major advantage, the bug-type algorithms have some significant shortcomings:

- they do not consider the actual kynematics of the robot, which is important with non-holonomic robots.
- they consider only the most recent sensor readings, and therefore sensor noise seriously affects the overall performance of the robot, they are slow.

2.2.1.2 The Potential Field Algorithm

While the bug-type algorithms are based on a purely reactive approach, the following algorithms tend to view the obstacle avoidance as a sub-task of the path planning, in a deliberative approach [20].
The potential field algorithm , assumes that the robot is driven by virtual forces that attract it towards the goal, or reject it away from the obstacles. The actual path is determined by the resultant of these virtual forces (see fig.2.12).

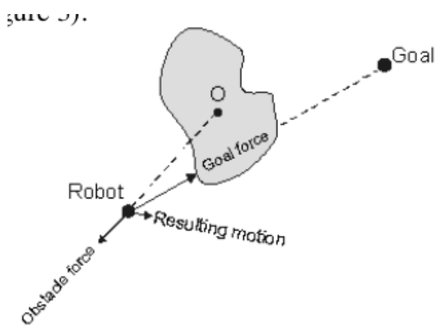


FIGURE 2.12: Illustration of the potential field algorithm [20].

Despite its elegance, this algorithm still does not solve all the drawbacks of the bug algorithms, performs poorly on narrow passages, and is more difficult to use in real time applications.

2.2.1.3 The Vector Field Histogram (VFH) Algorithm

The Vector Field Histogram, or VFH algorithm overcomes the problem of the sensors noise by creating a polar histogram of several recent sensor readings, like the one depicted in fig.2.13.
In fig.2.13, the x-axis represents the angles associated with sonar readings, and the y-axis represents the probability P that there really is an obstacle in that direction.
The probabilities are computed by creating a local occupancy grid map of the environment around the robot.
The polar histogram is used to identify all the passages large enough to allow the

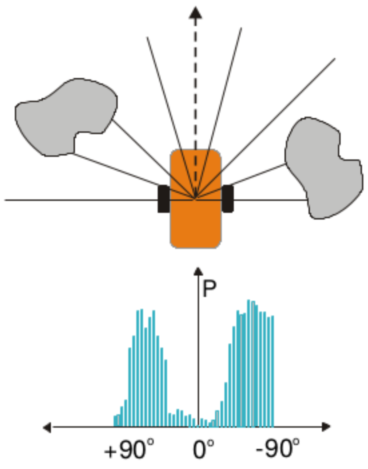


FIGURE 2.13: The polar histogram used in VFH algorithm[20].

robot to pass-through. The selection of the particular path to be followed by the robot is based on the evaluation of a cost function, defined for each passage. This depends on the alignment of the robot's path with the goal, and on the difference between the current wheel orientation and the new direction. The passage with the minimum cost is selected [22].
This algorithm offers better robustness to sensor noise, and takes into account the kinematics of the robot, but still involves a considerable computation load, which makes it difficult to implement on embedded systems.

2.2.1.4 The Bubble Band Technique

This method defines a “bubble” containing the maximum available free space around the robot, which can be traveled in any direction without collision [23]. The shape and size of the bubble are determined by a simplified model of the robot’s geometry, and by the range information provided by the sensors (see Fig.2.14). With this concept, a band of such bubbles can be used to plan a path between a starting point and a goal. Obviously, this technique is more a problem of offline path planning than one of obstacle avoidance, but we have included it in this brief presentation, because the idea of a bubble, seen as a subset of free space around the robot has some similarity with the solution proposed in this chapter [20].

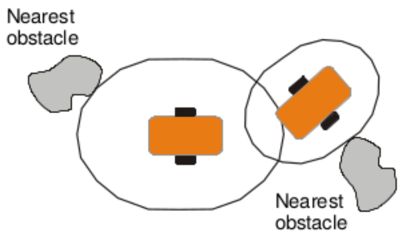


FIGURE 2.14: Illustration of the bubble band concept[20].

2.2.1.5 Other Obstacle Avoidance Algorithms

There are, of course, several other interesting algorithms for obstacle avoidance. However, relatively few of them are suitable for real-time, embedded applications, and will not be discussed here.

2.2.2 Description of the Proposed Algorithms

2.2.2.1 Detection of Obstacles

Consider a vehicle, having a ring of equidistant ultrasonic sensors, covering an angle of 180 degrees, as shown in fig.2.15.

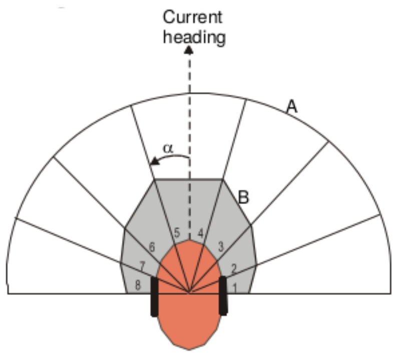


FIGURE 2.15: Robot, ultrasonic sensors, and sensitivity bubble[20].

If N is the number of sonar sensors, the following code defines the sensitivity bubble:

Algorithm 1: Algorithm Detection Obstacle

```
Result: defines the sensitivity bubble
initialization;
unsigned int $sonar\_readings[N]$;
unsigned int $bubble\_boundary[N]$;
$bubble\_boundary[i] = Ki * V * delta\_t$;
for $(i = 0; i < N; i++)$ do
    if $sonar\_readings[i] <= bubble\_boundary[i]$ then
        return(1);
    else
        return(0);
    end
end
```

Where, V is the translation velocity of the robot, *delta_t* is the time interval between successive evaluations of sensor data, and Ki are scaling constants, used for tuning. In our experiment,$k = \in [2 - 3]$.

Note that the shape and size of the sensitivity bubble (curve B in fig.2.15) defined

like this is dynamically adjusted, depending on the distance that can be traveled through by the robot, during the time interval *delta_t*, provided that the bubble does not exceed the range of the sonar sensors (curve A in fig.2.15). Also note, that the readings of the sensors represent the distance between the actual position of the sensor (which is on the boundary of the vehicle) and the obstacle. Therefore, the above definition of the sensitivity bubble indirectly includes information on the geometric shape of the robot[20].
Since the ultrasonic sensors are uniformly distributed, covering an arc of 180 degrees, the sonar readings can be represented in a polar diagram, as shown in fig.2.16.

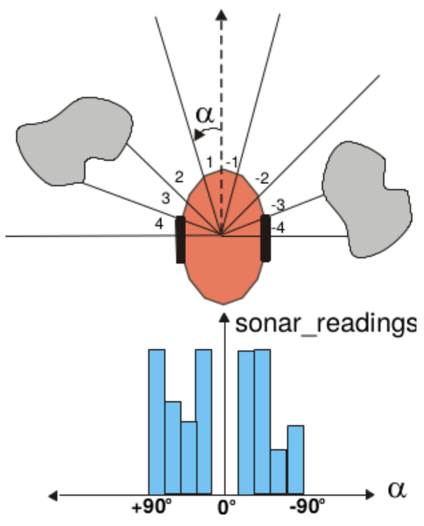


FIGURE 2.16: Polar diagram of the sonar readings[20].

2.2.2.2 Description of the Algorithm

Initially, the robot moves straight towards the goal. If an obstacle is detected within the sensitivity bubble, the robots "rebounds" in a direction found as having the lowest density of obstacles, and continues its motion in this new direction until the goal becomes visible (i.e. no obstacle within the visibility range of the sonar in that direction), or until a new obstacle is encountered[20].
Figure 2.17 presents an illustration of the rebound mechanism. In this image, H is the "hit-point" – the location of the robot at the moment of the detection of an obstacle, and V is the point where the robot regains visibility of the goal. The whole process is summarized in the flowchart presented in fig.2.18.

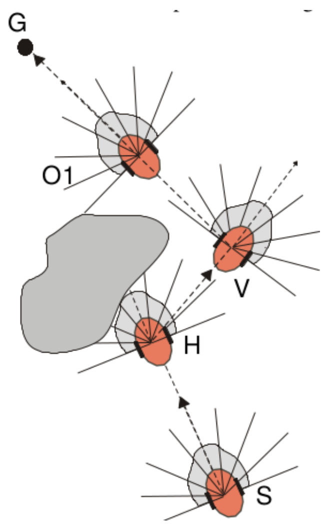


FIGURE 2.17: An illustration of the rebound process[20].

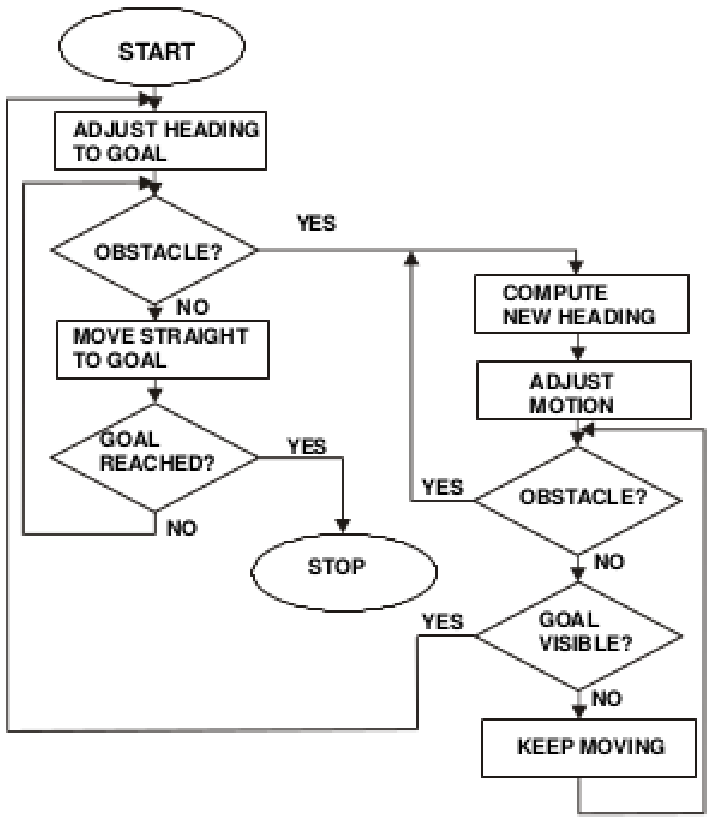


FIGURE 2.18: Flowchart for the bubble rebound algorithm[20].

Conclusion

A model of mobile robot navigation is considered in which the robot is a point automaton operating in an environment with unknown obstacles of arbitrary shapes. The robot's input information includes its own and the target-points coordinates as well as local sensing information such as that from stereo vision or a range finder. These algorithmic issues are addressed:

1. Is it possible to combine sensing and planning functions to produce 'active sensing' guided by the needs of planning? (The answer is **yes**).
2. Can richer sensing (e.g., stereo vision versus tactile) guarantee better performance, that is, resulting in shorter paths? (The general answer is **no**).

A paradigm for combining range data with motion planning is presented. It turns out that extensive modifications of simpler tactile algorithms are needed to take full advantage of additional sensing capabilities. algorithms that guarantee convergence and exhibit different styles of behavior are described, and their performance is demonstrated .

Chapter 3

Computer Vision and Autonomous Vehicles

Introduction
researchers are involved in the work that will someday make self-driving vehicles not just a reality, but commonplace. Working as part of a project known as CANVAS – Connected and Autonomous Networked Vehicles for Active Safety – the scientists are focusing much of their energy on key areas, including recognition and tracking objects such as pedestrians or other vehicles, fusion of data captured by radars and cameras, localization, mapping and advanced artificial intelligence algorithms that allow an autonomous vehicle to maneuver in its environment,and computer software to control the vehicle.

3.1 Computer vision

Computer vision is a field of informatics, which teaches computers to see. It is a way computers gather and interpret visual information from the surrounding environment [24]. Usually the image is first processed on a lower level to enhance picture quality, for example remove noise. Then the picture is processed on a higher level, for example detecting patterns and shapes, and thereby trying to determine, what is in the picture [25].

3.1.1 Object detection

Object detection is commonly referred to as a method that is responsible for discovering and identifying the existence of objects of a certain class. An extension of this can be considered as a method of image processing to identify objects from digital images[24].

3.1.2 Simple detection by colour

One way to do so, it to simply classify objects in images according to colour. This is the main variant used in, for example, robotic soccer, where different teams have assembled their robots and go head to head with other teams. However, this color-coded approach has its downsides. Experiments in the international RoboCup competition have shown that the lighting conditions are extremely detrimental to the outcome of the game and even the slightest ambient light change can prove fatal to the success of one or the other team. Participants need to recalibrate their systems multiple times even on the same field, because of the minor ambient light change that occurs with the time of day[26]. Of course, this type of detection is not suitable for most real world applications, just because of the constant need for recalibration and maintenance.

3.1.3 Introduction of Haar-like features

A more sophisticated method is therefore required. One such method would be the detection of objects from images using features or specific structures of the object in question.
However, there was a problem. Working with only image intensities, meaning the RGB pixel values in every single pixel in the image, made feature calculation rather computationally expensive and therefore slow on most platforms. This problem was addressed by the so-called Haar-like features, developed by Viola and Jones on the basis of the proposal by Papageorgiou et. al in 1998.
A Haar-like feature considers neighbouring rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums(see fig.3.1). This difference is then used to categorize subsections of an image.
An example of this would be the detection of human faces. Commonly, the areas around the eyes are darker than the areas on the cheeks. One example of a Haar-like feature for face detection is therefore a set of two neighbouring rectangular areas above the eye and cheek regions[27].

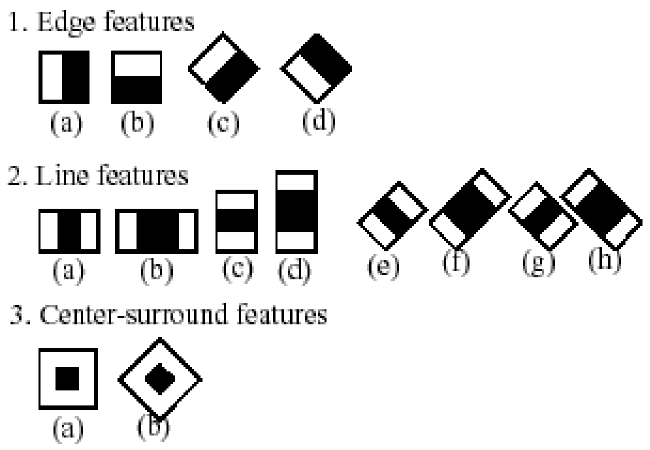


FIGURE 3.1: total set of features used by the Opencv detection see:$https : //docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html$

3.1.4 Cascade classifier

The cascade classifier consists of a list of stages, where each stage consists of a list of weak learners.
The system detects objects in question by moving a window over the image. Each stage of the classifier labels the specific region defined by the current location of the window as either positive or negative – positive meaning that an object was found or negative means that the specified object was not found in the image.
If the labelling yields a negative result, then the classification of this specific region is hereby complete and the location of the window is moved to the next location. If the labelling gives a positive result, then the region moves of to the next stage of classification. The classifier yields a final verdict of positive, when all the stages, including the last one, yield a result, saying that the object is found in the image. A true positive means that the object in question is indeed in the image and the classifier labels it as such – a positive result. A false positive means that the labelling process falsely determines, that the object is located in the image, although it is not. A false negative occurs when the classifier is unable to detect the actual object from the image and a true negative means that a non-object was correctly classifier as not being the object in question.
In order to work well, each stage of the cascade must have a low false negative rate, because if the actual object is classified as a non-object, then the classification of that branch stops, with no way to correct the mistake made. However, each stage can have a relatively high false positive rate, because even if the n-th stage classifies the non-object as actually being the object, then this mistake can be fixed in n+1-th and subsequent stages of the classifier [24].

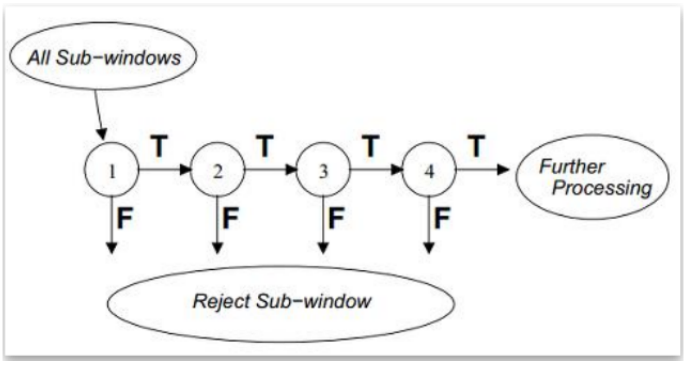


FIGURE 3.2: Stages of the cascade classifier.

3.2 Related work

3.2.1 PureTech Systems Car Counting

The system uses images received from IP and analog video cameras it to detect and count vehicles. Advanced background algorithms then filter any unnecessary and probable interference, such as shadows or lighting changes. When an object in detected, special filters make sure to minimize the chance to count nonvehicle items, for example humans and luggage. The finalized count is then outputted, based on the initial configuration - by floor or special zone.
The video is processed at a central monitoring location, which means there is no need to make cuts into pavement or similarly preinfluence the environment in such a way, which is commonly needed for inductive loop traffic detectors, where the detectors need to be placed inside the pavement, which is a fairly common method of detecting vehicles behind traffic lights.
One problem with this system is, however, that it only detects cars as they enter or leave the parking lot. This in turn means that there needs to be such a camera or counter in front of every entrance and exit point, in order to ensure that none of the vehicles are missed, which means that all these cameras need to be bought, installed and serviced, which means that the specific client will face a possibly substantial expenditure in order to keep all the devices up and running [24].

3.2.2 Autonomous Real-time Vehicle Detection

The primary means for vehicle detection make use of multiple cascaded Haar classifiers. For the application, four separate Haar-cascade classifiers were trained based on sample vehicles categorized into four different positional orientation to the horizontal line, resulting in the clockwise offsets of 0, 45, 90, 135 degrees. The training set was thereby divided into disjoint subsets based on the nearest perceived angle of the vehicle wheelbase. A separate cascaded Haar-classifier was trained for each of the subsets.
The results are subsequently passed through a secondary disjunctive verification process, which means that a vehicle may exist in one or more of the input images, if one or more of the different orientation specific classifiers yields a positive result. The returned set of regions are then merged to resolve overlaps and output a

singular set of detections from the inputs. Each region is also tested by various logical tests of width and height – is it actually probable for the image of this size to be a vehicle[24].

3.2.3 Monocular Vehicle Detection and Tracking

The system developed involves three main steps. Firstly, the preliminary car targets are generated with Haar-cascade classifier.
Only the candidates that pass through all the stages are classified as positive and the ones that are rejected at any stage are classified as negative. The advantage is that the majority of the initial candidates are actually negative images, which usually cannot pass the first few stages. This early-stage rejection thus greatly reduces the overall computing time. fter the first stage is complete, the target

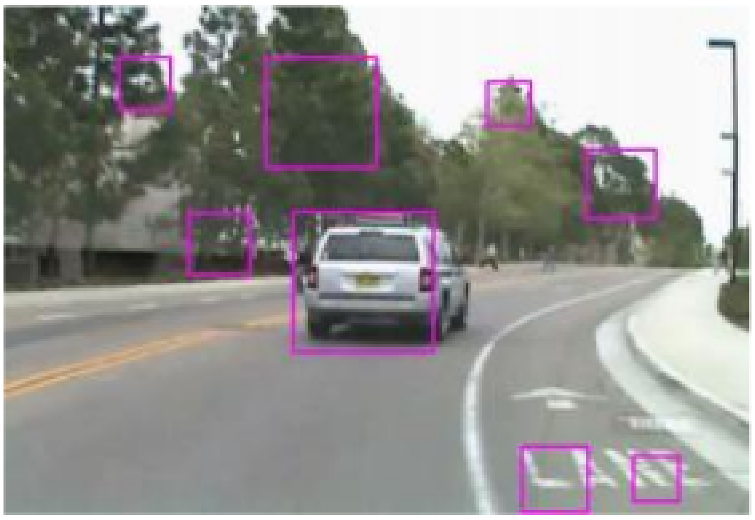


FIGURE 3.3: Car candidates detected by Haar-cascade.[24]

validation is used, which is based on car-light feature. Since there can be several false positives within the images output by the first stage of the system, which comes from the limitation of the training set. To reduce the false alarm rate, the algorithm uses the before mentioned car-light feature. Regardless of the shape, texture or colour of the vehicle, they all share a common feature – they all have red lights in the rear. Thereby assuming that most of the false positives detected by the first stage do not possess such a feature, the result can thus be refined[24].
Lastly, the results are further refined by Kalman tracking of the objects. The main idea of this step is based on a three-stage hypothesis tracking. Firstly, if a newly detected area appears and lasts for more than a certain number of steps, a

hypothesis of the object is generated. Then it is predicted where the next location of the object should be.

3.3 Object detection using Haar-cascade classifier

This section will highlight on the work conducted in the field of object detection using Haar-cascade classifier. The experiments were conducted mainly on the parking The location was chosen mainly for the ease of access and security for the hardware required to gather information[24].

3.3.1 Hardware

Initial testing was conducted with Qoltec WebCam15QT NightVision 30Mpx USB camera. The device was chosen due to its alleged high capabilities, especially the 30MP camera.
As it later turned out, the information provided was faulty, and the quality of the camera was in fact much poorer. For this reason, the device was replaced with Logitech HD Webcam C525 which offered much clearer images than its predecessor.
The camera was programmed to take pictures every five minutes, to minimize the impact on the storage capacity and duplicate images, since the changes during five minutes in the parking lot were observed to be minimal[28].
If the object is not detected for a certain number of frames, the hypothesis is discarded. This method can thusly eliminate false positives that do not last long enough and still keep track of objects that are missing for only a short period in a detection step.

3.3.2 Software

Several programs were developed in the course of this paper, ranging from a simple convert to grayscale and get size of picture to recorder, detector and PosCreator.

3.3.2.1 Recorder

Recorder application was a simple application which after every 5 minutes tries to take a picture. If it can, then a picture is saved to a folder of the corresponding date with the filename of the corresponding time. If it cannot, then it simply cuts the connection within 30 seconds and will simply wait for the next 5 minutes. This ensures that if there is a problem with taking a picture, which would cause the program to "freeze", then it is simply stops the program and tries again later, instead of potentially waiting until the power runs out someone manually stops the program. This is a must-have feature is such an application, due to the fact that several hours' worth of image gathering would be wasted due to any simple problem that halts the execution of the recorder thread[24].

3.3.2.2 Detector

The initial design of the detector application was quite simplistic. Firstly, the detector would load the classifier and determine it is not empty. If it is, then it simply exits with an error message. Then the image in question is loaded and same procedure is followed. Then classifier is applied to the image, which outputs an array of rectangles, which correspond to the detected positions of the objects, in this case automobiles. The program would then draw bright red rectangles in the locations of the detections and also add a text to the image, which could for example identify the classifier used, since one classifier would usually detect one thing[24].

3.3.2.2.1 Background subtraction However, as shown by the testing process and the literature, the classifiers trained can produce errors – either false positives or false negatives, as described above.

In order to minimise the false positive rate originating from the imperfections of the classifier, an additional layer was added to the algorithm, before the classifier is applied to the image. This layer has additional knowledge of the complete background. In this case it would be an image of only the parking lot and everything that would normally be in the parking lot, except for the cars themselves. This knowledge can be applied to attempt the filtering of the background from the image from which we would like to detect vehicles. The background subtraction type used was MOG.

MOG (abbr. from Mixture of Gaussians) is a technique used to model the background pixels of an image like a mixture of Gaussians of different weight that represent the pixel intensity. If a match is found for the pixel of the new image with one of the Gaussians then the point is classified as a background pixel. If no match is found, then the pixel is classified as the foreground pixel [29].
Other algorithms, such as MOG2 were considered, but MOG was finally chosen due to the simple fact that clearer results were obtained by using MOG. MOG gives us the background mask, so in order to apply it to the original picture, one would simply need to compute the bitwise and between the original image and the mask provided.
MOG is, however, not perfect. If we were to just take the mask provided by the default MOG background extractor, then the output for one image of the parking lot would be rather low quality, as illustrated on fig.3.4. Although a person may differentiate the regions of cars in the image, a cascade classifier proved unable to properly comprehend the regions of cars on a similar image.

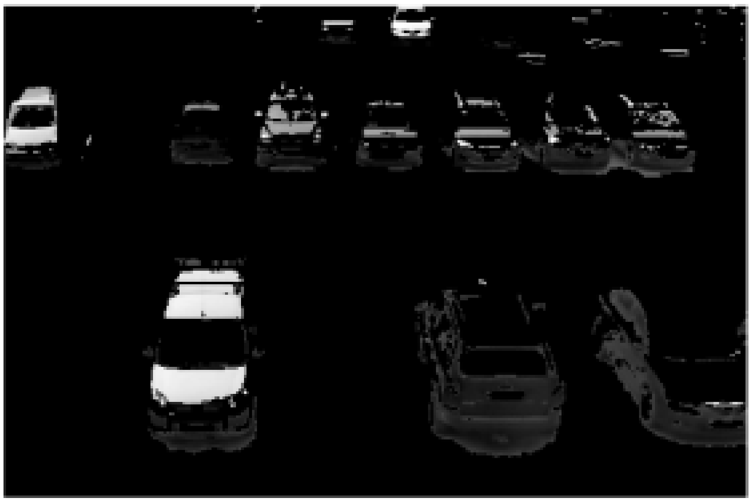


FIGURE 3.4: Output using MOG with default parameters.[24]

3.3.2.2.2 Background subtraction augmentation In order to amend this issue, different augmenting features had to be used. The ones chosen were eroding and dilating.
Dilation is a way to make the bright regions of an image to "grow". As the kernel (small matrix used for image processing) is scanned over the image, the maximal pixel value overlapped by the kernel is calculated and the image pixel in the anchor

point of the kernel (usually at the centre of the image) is replaced by the maximal value.
Erosion works similarly to dilation, but instead of the maximal, it computes the local minimum over the area of the kernel, thus making the dark areas of the image larger. If one were to apply dilation to the mask provided by MOG, then the areas of the mask which are not zeros would get larger, thus improving the overall quality of the image[24].
This can however raise a new issue, namely the fact that the small noisy areas present in the original mask could grow larger and have a negative effect on the provided mask.
For this reason, the once dilated mask is eroded with a kernel with a smaller size, so that it would not nullify the result provided by the dilating but still reducing the amount of noise produced by the dilation process, thus providing a symbiotic relation between the two operations.

The results provided by this sort of background filtering were improved. Since

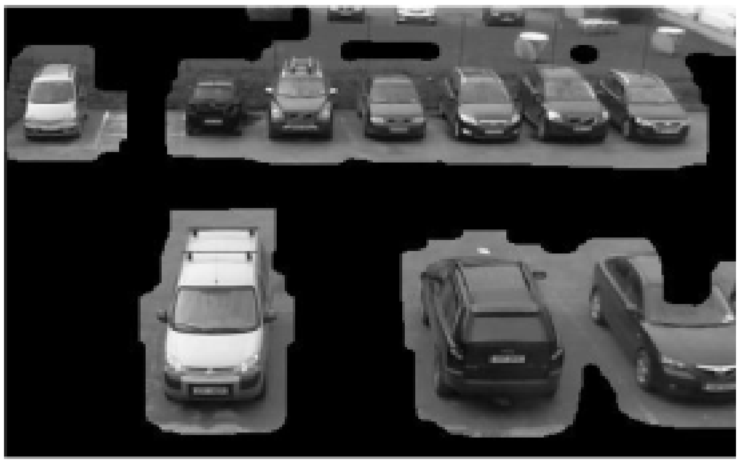


Figure 3.5: MOG with dilation anderosion.[24]

a lot of the false positives provided by the original detections were in fact on the background part, such as the trees, pavement etc., which is always there, then the algorithm discarded these areas before the Haar-cascade classifier would be applied. However, the regions created by the background removal created additional problems, such as the classifier mistaking the grey to black regions as the positive image.

3.3.2.3 Regional merging

In addition, yet another layer of filtering was added, inspired by the “Autonomous Real-time Vehicle Detection from a Medium-Level UAV” article. This layer was added after the Haar-cascade has done its work. Since it is possible for the classifier to detect many parts of a car as several distinct cars, the results of the classifier regions must be merged. The merge cannot however just detect whether the two regions have an overlap of any size, because the overlap can simply be the result of two adjacent cars, which would thereby be merged into one result. To avoid this unwanted behaviour, the two regions are merged only if the overlapped area is at least a certain percentage of one of the regions. This constant was observed to yield the best results if the value was approximately 40-60%, meaning that if the overlap of the two regions made up at least 40-60% of the overall area of one of the regions, then the two regions were merged and considered as one region from there on. An example of this procedure is shown on image 5, where green lines denote the original detection and the red line the final output of the detector[24].

Due to the probabilistic nature of the Haar-cascade, the false positives may also

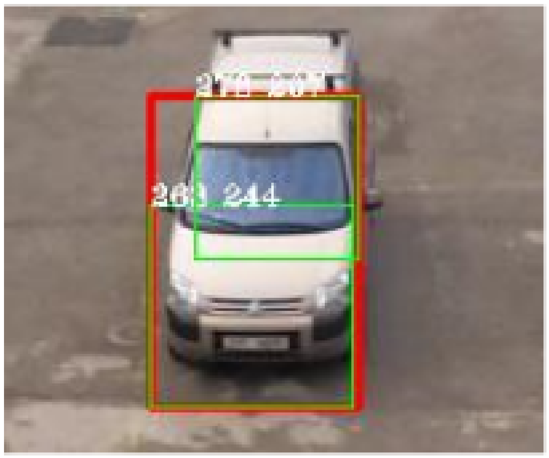


FIGURE 3.6: Regional merge in action.

include detection, which logically cannot contain an automobile – namely when the rectangle identified as a positive result is either too large or too small. Such a result would seriously compromise the result set, because the algorithm described for the merging of areas would simply merge all these areas into one large area. In order to avoid this, the results of the Haar-cascade are first prefiltered, so that areas that are too large to contain just a single car are removed from the results.

This size constant was found by observing the largest truck that was closest to the camera and adding 20% to the size.

3.3.3 Training cascade

The training of the cascade proved to be no easy task. The first necessary bit was to gather the images, then create samples based on them and finally starting the training process. The opencv traincascade utility is an improvement over its predecessor in several aspects, one of them being that traincascade allows the training process to be multithreaded, which reduces the time it takes to finish the training of the classifier. This multithreaded approach is only applied during the precalculation step however, so the overall time to train is still quite significant, resulting in hours,days and weeks of training time [30].
Since the training process needs a lot of positive and negative input images, which may not always be present, then a way to circumvent this is to use a tool for the creation of such positive images.
OpenCV built in mode allows to create more positive images with distorting the original positive image and applying a background image. However, it does not allow to do this for multiple images.
By using the Perl script createsamples to apply distortions in batch and the mergevec tool , it is possible to create such necessary files for each positive input file and then merging the outputted files together into one input file that OpenCV can understand.
Another important aspect to consider is the number of positives and negatives. When executing the command to start training, it is required to enter the number of positive and negative images that will be used. Special care should be taken with these variables, since the number of positive images here denotes the number of positive images to be used on each step of the classifier training, which means that if one were to specify to use all images on every step, then at one point the training process would end in an error. This is due to the way the training process is set up. The process needs to use many different images on every stage of the classification and if one were to give all to the first stage, then there would be no images left over for the second stage, thus resulting in an error message [24].
he training can result in many types of unwanted behaviour. Most common of these is either overtraining or undertraining of the classifier. An undertrained

classifier will most likely output too many false positives, since the training process has not had time to properly determine which actually is positive and which is not. An output may look similar to fig.3.7. The opposite effect may be observed

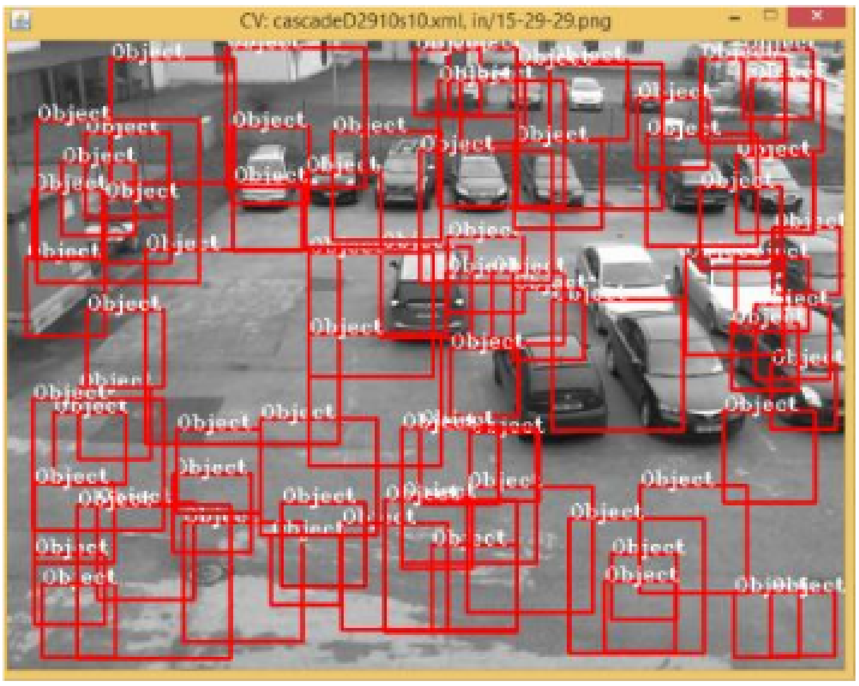


FIGURE 3.7: Depiction of the results of undertrained classifier.

if too many stages are trained, which could mean that the classification process may determine that even the positive objects in the picture are actually negative ones, resulting in an empty result set.
Fairly undefined behaviour can occur if the number of input images are too low, since the training program cannot get enough information on the actual object to be able to classify it correctly.
The opposite effect may be observed if too many stages are trained, which could mean that the classification process may determine that even the positive objects in the picture are actually negative ones, resulting in an empty result set. Fairly undefined behaviour can occur if the number of input images are too low, since the training program cannot get enough information on the actual object to be able to classify it correctly[24]. One of the best results obtained in the course of this work is depicted on fig.3.8. As one can observe, the classifier does detect some vehicles without any problems, but unfortunately also some areas of the pavement and some parts of grass are also classified as a car. Also some cars are not detected as standalone cars. The time taken to train the classifier to detect at this level can be measured in days and weeks, rather than hours.

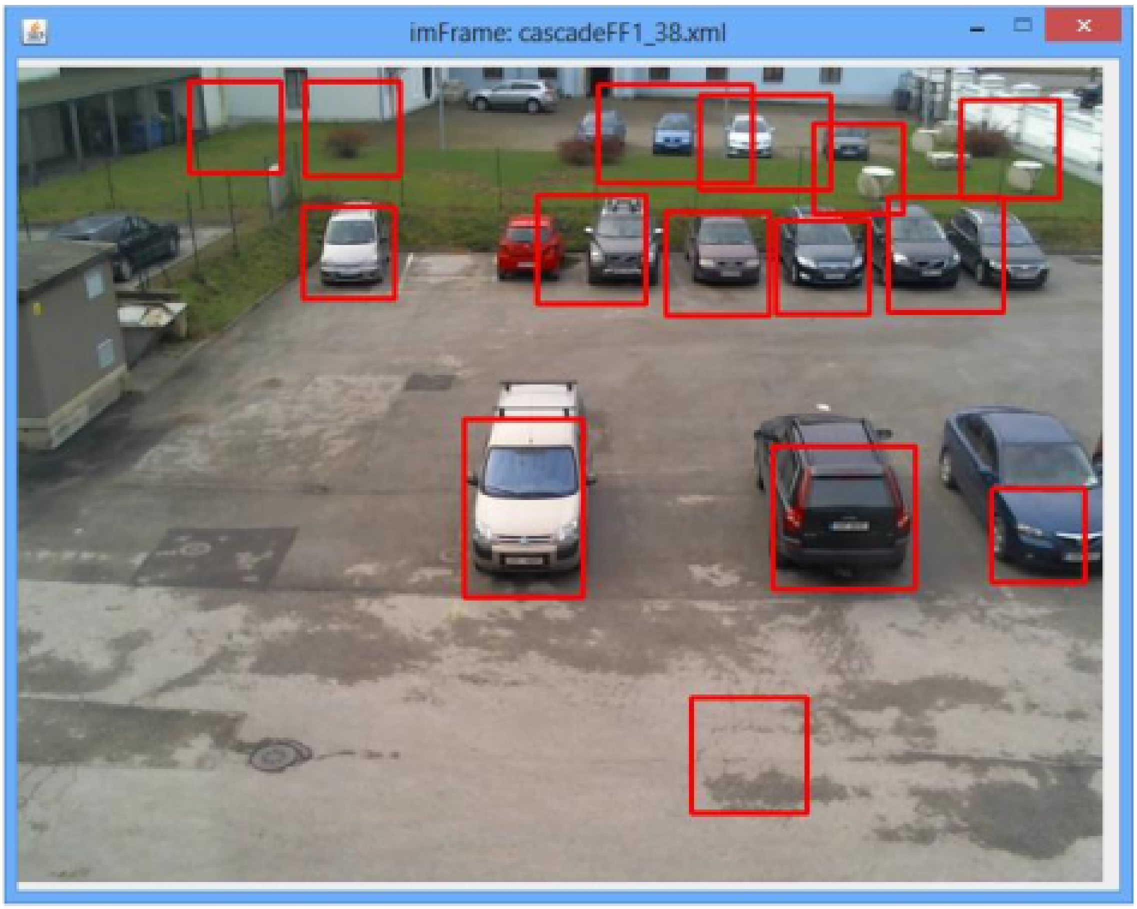


FIGURE 3.8: Best solution obtained by the author.

Since the training process is fairly probabilistic, then a lot of work did also go into testing the various parameters used in this work, from the number of input images to the subtle changes in the structuring element on the background removal, and verifying whether the output improved, decreased or remained unchanged[24].

3.4 Color detection using HSV method

HSV (Hue-Saturation-Value) color space is more often used in machine vision owing to its superior performance compared to RGB color space in varying illumination levels [31]. Often thresholding and masking is done in HSV color space. So it is very important to know the HSV values of the color which we want to filter out. HSV color space of OpenCV is a bit complicated than other software programmes like Gimp, Photoshop etc.

3.4.1 Hue

Hue is the color portion of the color model, expressed as a number from 0 to 360 degrees:

Color	**Angle**
Red	0–60
Yellow	60–20
Green	120–180
Cyan	180–240
Blue	240–300
Magenta	300–360

TABLE 3.1: Color degrees

3.4.2 Saturation

Saturation is the amount of gray in the color, from 0 to 100 percent. Reducing the saturation toward zero to introduce more gray produces a faded effect. Sometimes, saturation is expressed in a range from just 0-1, where 0 is gray and 1 is a primary color [31].

3.4.3 Value (or Brightness)

Value works in conjunction with saturation and describes the brightness or intensity of the color, from 0-100 percent, where 0 is completely black, and 100 is the brightest and reveals the most color.
The HSV color model is used when selecting colors for paint or ink because HSV better represents how people relate to colors than does the RGB color model. The HSV color wheel is also used to generate high-quality graphics. Although less well known than its RGB and CMYK cousins, the HSV approach is available in many high-end image editing software programs[31].
Selecting an HSV color begins with picking one of the available hues, which is how most humans relate to color and then adjusting the shade and brightness values.
-we can use HSV mask for filter image (see fig.3.9) .

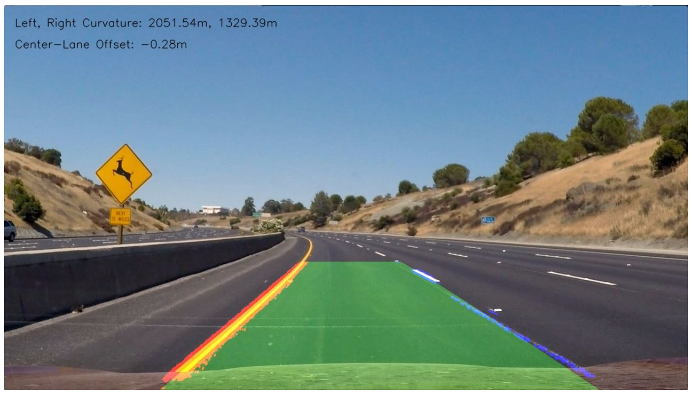

Left, Right Curvature: 2051.54m, 1329.39m
Center–Lane Offset: −0.28m

FIGURE 3.9: Tracing road by HSV method.

Conclusion

Recent years have witnessed amazing progress in AI related fields such as computer vision, machine learning and autonomous vehicles. As with any rapidly growing field, however, it becomes increasingly difficult to stay up-to-date or enter the field as a beginner. While several topic specific survey papers have been written, to date no general survey on problems, datasets and methods in computer vision for autonomous vehicles exists. This chapter attempts to narrow this gap by providing a state-of-the-art survey on this topic. Our survey includes recognition, reconstruction, motion estimation, tracking, scene understanding and end-to-end learning.

Chapter 4

Tools for dynamics simulation of robots

Introduction

Simulation has been recognized as an important research tool since the beginning of the 20th century. In the beginning, simulation was first of all an academic research tool. The “good times” for simulation started with the development of computers. First the analog computers and later the digital computers have boosted simulation to new levels. So, the simulation is now a powerful tool supporting the design, planning, analysis, and decisions in different areas of research and development. Simulation has become a strategic tool in many fields, used by many researchers, developers and by many manufacturers. Of course, robotics as a modern technological branch is no exception. Actually, in robotics, simulation plays a very important role, perhaps more important than in many other fields. This chapter presents robotics from the simulation point of view.

Topics like how simulation makes things easier, advantages and backdrops of the simulation in robotics, virtual and real world, are pointed out. This chapter presents typical simulation examples in different fields of robotics. The simulation of kinematics and dynamics of robotic manipulators, and integrated environments for dynamic simulation of robot manipulators are also discussed. It compares some tools for the of robot manipulators. An overview of simulation and visualization tools suitable for the simulation of robot systems using general dynamic engines and graphic languages is given.

4.1 Definition of simulator

A device or software that enables the operator to reproduce or represent under test conditions phenomena likely to occur in actual performance.

4.2 The role of simulation

Simulation is the process of designing a model of an actual or theoretical physical system, executing the model, and analysing the execution output. Actually, the simulation is very common in one's life; it helps to understand reality and all of its complexity[32].
Artificial objects are built and an act role dynamically with them is done. It is very common in one's life that they come across simulation at early stages of their life. Namely, playing a game in our youth, e.g. being a parent, a fireman, etc., is just a simulation where a child is playing a role in a virtual environment with toys. The principle "learning" is essential in simulation. Using simulation one can learn about something in a very effective way and while modifying "rules" the effects of our interaction can observed. "Seeing is believing" to every individual so the visualization is the other paramount in simulation. Being able to simulate opens a wide range of options for solving many problems creatively[33].
Investigation, design, visualization, and testing of an object or even if it does not exist (virtual) can be done. The results of a system yet to be built can also be seen. It is possible that solutions may fail or even blow up, but only in simulation. So, using the simulation tools one can avoid injuries and damages, unnecessary changes in design after the production of parts has already started, to long cycle times in manufacturing process, and even unnecessary paper work. Simulation enables us to work even in four dimensions. For example, one can observe within a few minutes how a planned production will be realized in next month, or a fast process can be slowed down to observe all details in "slow motion". All these make things easier and cheaper.
Simulation is a highly interdisciplinary field since it is widely used in all fields of research from engineering and computer science to economics and social science, and at different levels from academic research to manufactures[34].
Simulation has been also recognized as an important tool in robotics: in designing new products, investigating its performances and in designing applications of

these products. Simulation also helps studying the structure, characteristics and the function of a robot system at different levels of details each posing different requirements for the simulation tools. As the complexity of the system under investigation increases the role of the simulation becomes more and more important. Advanced robotic systems are quite complex systems[35].
Hence, the simulation tools can certainly enhance the design, development, and even the operation of robotic systems. Augmenting the simulation with visualization tools and interfaces, one can simulate the operation of the robotic systems in a very realistic way. Depending on the particular application different structural attributes and functional parameters have to be modeled. Therefore, a variety of simulation tools has been developed for the robotic systems which are used in mechanical design of robotic manipulators, design of control systems, off-line programming systems, to design and test the robot cells, etc.

4.3 Challenges in simulation

Dynamics simulators for robotics have more strict requirements than the ones used for animating virtual characters, where time, computational burden and physical reality can be less constraining. In entertainment (e.g. video-games), unfeasible forces may not be a problem since the law of physics can be violated. In bio/mechanical studies, simulators can be used offline to analyze or synthesize behaviors. Although the field of dynamics modeling and simulation has matured over the last decades , the growing need to control wholebody movements of complex structures, such as humanoids, poses additional challenges to simulators for robotics[36]:

1. **numerical stability:** which poses strong limitations on the use of simulations in real-time control settings.
2. **the capability to be used as predictive engines in real-time control loops:** which requires the ability to be extremely fast in computing the dynamics and the guarantee for the solvers to converge to physically feasible solutions upon a certain time .
3. **the simulation of rigid and soft bodies in contact with rigid and compliant environments :** the inaccurate computation of contact forces between bodies may result in unrealistic contacts or physically unfeasible

contact forces (this issue has been particularly evident in the virtual phase of the Darpa Robotics Challenge - DRC).

4. **the capability to model and simulate new types of actuation systems:** such as variable impedance or soft actuators , and different types of contacts, for example with deformable materials, compliant and soft surfaces.

Finally, the robotics community urges for standardized software tools and particularly open source software. The benefit of open-source is not only in the community that can grow around the software, developing new tools, improving its quality and avoiding to "re-invent the wheel" at each time, but also in checking its efficiency and robustness on real platforms (which is expensive).

4.4 Overiew of simulators

In this section will analyse different softwares of simulations starting survey done by researchers Serena Ivaldiy and other in [36]in different countries , The survey was filled by 119 participants.

4.4.1 Important features for simulation

The main purposes for the use of dynamics simulation in researchs:

- 66% simulating the interaction of the robot with the environment.
- 60% simulating the robot locomotion.
- 59% simulating behaviors of the robot before doing them on the real robot.
- 49% simulating the robot navigation in the environment.
- 48% simulating collisions and interactions between bodies (not specifically robots).
- 41% testing low-level controllers for robots.
- 22% simulating multi-fingered grasp.

- 21% simulating human movements.
- 8% animating virtual characters.

The most important features for a good simulation (they could evaluate the importance of each element from "not important at all" - 1 to "very important, crucial" - 5). Their ranking of important features is reported in Table 4.1. The stability of simulation is the only element that was evaluated as "very important", whereas speed, precision and accuracy of contact resolution were marked important. Remarkably, the same API between real and simulated robot is also signed as important.

Rank	**Feature**	**Overall Evaluation**	**Rating**	**Median rating**
1	Stability of simulation	Very important	4.50 ± 0.58	5
2	Speed	Important	4.05 ± 0.75	4
3	Precision of simulation	Important	4.02 ± 0.71	4
4	Accuracy of contact resolution	Important	3.91 ± 0.92	4
5	Same interface between real & simulated system	Important	3.67 ± 1.26	4
6	Computational load (CPU)	Neutral	3.53 ± 0.85	3
7	Computational load (memory)	Neutral	3.22 ± 0.90	3
8	Visual rendering	Neutral	3.02 ± 1.02	3

TABLE 4.1: Most important features for a simulator.[36]

4.4.2 Criteria for choosing a simulator

The first most important criteria:

Rank	**Most important criteria**
1	Simulation very close to reality
2	Open-source
3	Same code for both real and simulated robot
4	Light and fast
5	Customization
6	No interpenetration between bodies

TABLE 4.2: Most important criteria for choosing a simulator.[36]

4.4.3 Currently used tools

The most current simulation tool are using. Results are shown in Fig.4.1. The most diffused software among the users are:13% Gazebo, 9% ARGoS, 8% ODE, 7% Bullet, 6% V-Rep, 6% Webots, 5% OpenRave, 4% Robotran, 4% XDE. All the other tools have less than 4% of user share.

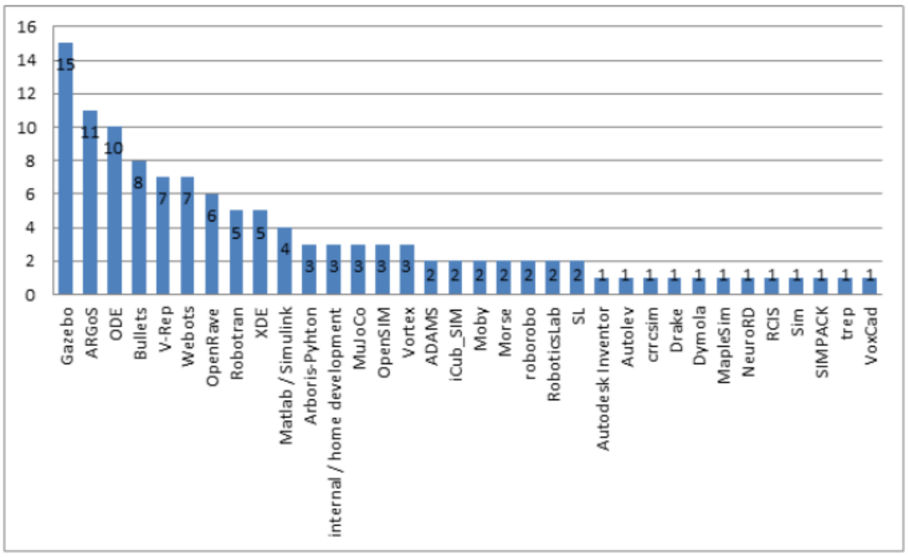


FIGURE 4.1: The simulation tools currently used.[36].

These tools are the ones we are focusing on in our following analysis. Some technical information about the selected tools can be indicative of the user needs and use:

- **Primary OS:** 66% GNU/Linux, 30%Windows, 4% MAC OSX.
- **Primary API language:** 52% C++, 18% python, 13% Matlab, 8%C, 3% LUA, 2% Java; 3% of participants do not use an API
- **License:** 67% of the tools are open-source (GPL, Apache, BSD and analogous/derivatives licenses), only 17% of the tools have a commercial license, 16% have an academic license (i.e., they are free but not open-source).
- **Hardware:** 39% a powerful desktop (i.e., multi-core, 8/16GB RAM), 35% everyday laptop, 18% powerful desktop with powerful GPU card, 5% multi-core cluster.
- **Middleware:** 52% is not using the tool with a middleware, the remainder is using ROS (25%), YARP (6%), OROCOS (4%).

The research areas being different, we extracted the most used tools for a selection of research areas: results are shown in Table 4.3. The most relevant results are for

humanoid robotics (31 users, that is 26% of the participants to the survey) and mobile robotics (25 users, that is 21% if the participants).
For humanoid robotics, the most diffused tools are ODE and Gazebo, and there is a variety of several custom-made simulators. It is interesting to notice that Gazebo supports ODE and Bullet as physical engines, hence it is probable that the quota of ODE for humanoid robotics is higher. For mobile robotics, the most diffused tools among the survey participants are Gazebo, ARGoS and Webots.

Research area	**Users**	**Most used software**	**Other used software**
Humanoid Robotics	32	(4) ODE, (3) Gazebo, Robotran, OpenRave, Arboris-Python, (2) XDE, iCub SIM	(1) Drake, MapleSim, MuJoCo, OpenSIM, RoboticsLab,SL, Vortex, V-Rep, Webots, own code
Mobile Robotics	25	(5) Gazebo, ARGoS, (3) Webots, (2) VRep,Vortex	(1) ADAMS, Autodesk Inventor, Bullet, ODE,Morse, roborobo, Sim, own code
Multi-legged robotics	13	(3) Webots, (2) ODE	(1) Gazebo, ADAMS, Autolev, Bullet, Moby, RoboticsLab, SIMPACK, VoxCad
Service robotics	12	(4) Gazebo, (3) OpenRave	(1) OpenSIM, V-Rep, Morse, RCIS, SL
Numerical simulation of physical systems	8	(2) Bullet	(1) MuJoCo, ODE, OpenSIM, Simulink, trep, XDE
Flying robots	6	(2) ARGoS	(1) Robotran, crrcsim, Gazebo, Simulink/Matlab
Swarm robotics	5	(4) ARGoS	(1) roborobo
Industrial manipulators	5		(1) Bullets, Dymola, Matlab, V-Rep, XDE
Mechanical design	4		(1) Moby, MuJoCo, V-Rep, own code
Human Motion analysis	3		(1) Robotran, Bullet, XDE
Snake robots	3	(2) ODE	(1) Matlab

Table 4.3: Most diffused tools for a selection of the research areas.[36]

The different concentration of tools for the different research areas reveals that some tools are more appropriate than others for simulating robotic systems in different contexts or applications. A researcher may therefore let his choice about

Tool	**Documentation**	**Support**	**Installation**	**Tutorials**	**Advanced use**	**Active project & community**	**API**	**Global**
Gazebo	3.47 ± 0.99	4.00 ± 1.07	3.93 ± 1.03	3.53 ± 1.12	3.80 ± 0.86	4.73 ± 0.45	3.67 ± 0.82	3.88 ± 0.91
ARGoS	3.40 ± 0.70	3.90 ± 0.99	4.70 ± 0.48	4.20 ± 0.63	4.60 ± 0.70	4.10 ± 0.74	4.30 ± 0.67	4.17 ± 0.70
ODE	3.80 ± 0.63	3.40 ± 1.07	4.10 ± 0.46	3.20 ± 1.13	3.90 ± 1.37	3.30 ± 1.25	3.40 ± 1.26	3.59 ± 1.15
Bullets	3.37 ± 1.06	3.62 ± 0.91	4.75 ± 0.46	4.00 ± 0.76	3.75 ± 0.71	4.37 ± 0.74	3.87 ± 0.83	3.96 ± 0.78
V-Rep	4.28 ± 0.76	4.43 ± 0.79	4.71 ± 0.76	4.14 ± 0.90	4.28 ± 0.76	4.43 ± 0.53	4.14 ± 1.07	4.25 ± 0.80
Webots	3.86 ± 1.07	3.57 ± 1.13	4.43 ± 0.79	3.43 ± 1.51	4.42 ± 0.78	4.14 ± 0.69	4.57 ± 0.53	4.20 ± 0.96
OpenRave	3.50 ± 0.55	4.67 ± 0.52	4.17 ± 0.75	3.50 ± 1.22	4.33 ± 0.82	4.33 ± 0.52	4.33 ± 0.52	4.12 ± 0.70
Robotran	3.60 ± 0.55	3.80 ± 0.45	3.80 ± 0.45	3.20 ± 0.84	4.20 ± 0.84	3.20 ± 0.84	3.80 ± 0.45	3.66 ± 0.63
Vortex	3.33 ± 1.15	3.67 ± 1.53	5.00 ± 0.00	2.67 ± 0.58	3.67 ± 0.58	2.67 ± 1.15	3.33 ± 0.58	3.48 ± 0.80
OpenSIM	4.33 ± 0.58	4.67 ± 0.58	3.67 ± 0.58	3.00 ± 1.00	4.00 ± 0.00	4.67 ± 0.58	3.67 ± 0.58	4.00 ± 0.55
MuJoCo	2.33 ± 1.15	1.67 ± 0.58	4.33 ± 1.15	3.33 ± 1.15	4.67 ± 0.57	4.00 ± 0.00	5.00 ± 0.00	3.62 ± 0.66
XDE	1.40 ± 0.55	2.80 ± 1.09	3.60 ± 0.55	2.80 ± 1.09	3.40 ± 1.10	2.80 ± 0.84	3.00 ± 1.00	2.83 ± 1.07

TABLE 4.4: Ratings for the level of user satisfaction of the most diffused tools.[36]

the adoption of a simulator be guided by the custom in his field. With this in mind, we investigated what was the main reason for a researcher to pick up his current tool.

Overall, the main reasons why they chose the current tool is: 29% the best tool for their research upon evaluation, 23% "inheritance", i.e. it was "the software" (already) used in their laboratory, 8% they are the developers, 8% it was chosen by their boss/project leader, 7% it is open-source, 7% it was happily used by colleagues. Only 3% of the participants chose the tool because of a robotic challenge. Interestingly there is quite a demarcation between the first reasons and the others. There are certainly some tools that distinguish for the fact that they have been chosen as best option for research, for example V-Rep (71%), Bullet (63%) and Gazebo (53%). Some tools have instead been adopted by "inheritance", i.e., they were already used in the lab: ARGoS (45%), Robotran (40%) and XDE (40%). For the latter, it is also a choice imposed by the project leader (40%).

We asked participants to evaluate their level of satisfaction of the use of their tool, in a global way, from Very negative (1) to Very Positive (5): all software tools were evaluated "positive", whereas only MuJoCo was "very positive" (subjective evaluation by 3 users). We also asked participants to indicate their level of satisfaction with respect to some specific aspects (documentation, support, installation, tutorials, advanced use, active project and community, API), and to rate each element on a scale from 1 to 5. Table 4.4 reports the mean and standard deviation of the notes received by the users of each tool[36].

4.4.4 Tools for robots

The majority of users is using the software tool to simulate robots (91%). Users could point out the robots they are simulating (more than one in general):
the aggregated table of simulated robot is shown in fig.4.1, where the x-axis shows how many users selected the robot.
We extracted the principal tools used for simulating the main robots:

- **iCub:** 25% Arboris-Python, 17% ODE, 17% Robotran, 17% iCub SIM
- **Atlas:** 50% Gazebo, 25% MuJoCo, 12% Autolev, 12% Drake PR2: 21% OpenRave, 14% Gazebo, 14% MuJoCo, 7% Bullet, 7% V-Rep
- **Multi-legged robot:** 22% ODE, 11% SL, 11% Bullet, 11% Webots
- **Wheeled vehicle:** 14% Gazebo, 14% V-Rep, 11% ARGoS, 7% Morse, 7% Webots, 7% Vortex
- **Quadrotor:** 24% Gazebo, 24% ARGoS, 12% V-Rep

4.5 SOFTWARE INFORMATION FICHES

We report in the following some essential information for the main software tools (the most diffused) that may be of help for the interested reader.

4.5.1 Gazebo

Gazebo is a multi-robot simulator for outdoor environments, developed by Open-Source Robotics Foundation. It is the official software tool for the DRC (Darpa Robotics Challenge). It supports multiple physics engines (ODE, Bullet).

- **Web:** http://gazebosim.org/
- **License:** Apache 2
- **OS share:** 100% GNU/Linux
- **Main API:** 80% C++

- **Main reason for adoption:** 53% best tool upon evaluation, 20% software already used in the lab, 20% official tool for a challenge, 7% open-source
- **Mostly used:** in USA (33%)
- **Mainly used for:** 33% mobile robotics, 27% servicerobotics, 20% humanoid robotics
- **Main simulated robots:** 40% Atlas, 33% custom platform,27% wheeled vehicle, 27% quadrotor, 27% turtlebot, 20%PR2
- **Main middleware used with:** 93% ROS
- **Main simulated robots:** 40% Atlas, 33% custom platform,27% wheeled vehicle, 27% quadrotor, 27% turtlebot, 20%PR2

4.5.2 ARGoS

ARGoS is a multi-robot, multi-engine simulator for swarm robotics, initially developed within the Swarmanoid project.

- **Web:** http://iridia.ulb.ac.be/argos/
- **License:** GPLv3.0
- **OS share:** 91% GNU/Linux, 9% MAC OSX
- **Main API:** 73% C++
- **Main reason :** 45% software already used in the lab, 27% colleagues using it.
- **Mostly used:** in Belgium (36%) and Italy (27%)
- **used for:** 46% mobile robotics, 36% swarm robotics,18% flying robots
- **Main simulated robots:** 64% khepera/e-puck/thymio, 36% marXbot/-footbot, 27% quadrotor.

4.5.3 ODE

ODE (Open Dynamics Engine) is an open-source library for simulating rigid body dynamics, used in many computer games and simulation tools. It is used as physics engines in several robotics simulators, such as Gazebo and V-Rep.

- **Web:** http://www.ode.org/
- **License:** GNU LGPL and BSD
- **OS share:** 100% GNU/Linux
- **Main API:** 80% C++
- **Main reason :** 50% best tool upon evaluation, 20% used before, 10% boss choice, 10% open-source, 10% software already used in the lab.
- **Mostly used:** in France (20%)
- **used for:** 50% humanoid robotics, 20% multi-legged robotics, 20% snake robots, 10% numerical simulation of physical systems
- **Main simulated robots:** 40% multi-legged robot, 20% iCub

4.5.4 Bullet

Bullet is an open-source physics library, mostly used for computer graphics and animation. The latest release5 also supports Featherstone's articulated body algorithm and a Mixed Linear Complementarity Problem solver, which makes it suitable for robotics applications.

- **Web:** http://bulletphysics.org
- **License:** ZLib license, free for commercial use
- **OS share:** 50% Windows, 38% GNU/Linux, 12% MAC OSX
- **Main API:** 75% C++
- **Main reason :**63% best tool upon evaluation, 25% opensource, 12% colleagues using it

- **Mostly used:** in France (25%), Italy (25%) and Belgium (25%)
- **used for:** 25% humanoid robotics, 25% numerical simulation of physical systems, 12.5% industrial manipulators, 12.5% human motion analysis, 12.5% mobile robotics, 12.5% multi-legged robotics.
- **Main simulated robots:** 25% multi-legged robot.

4.5.5 V-Rep

V-Rep is a robot simulator software with an integrated development environment, produced by Coppelia Robotics. Like Gazebo, it supports multiple physics engines (ODE, Bullet, Vortex).

- **Web:** http://www.coppeliarobotics.com/
- **License:** Dual-licensed source code: commercial or GNUGPL
- **OS share:** 57% GNU/Linux, 43% Windows
- **Main API:** 57% C++, 29% LUA
- **Middleware :**43% ROS, 57% None
- **Main reason:**72% best tool upon evaluation, 14% colleagues using it, 14% boss choice
- **used for:** 29% mobile robotics, 14% industrial manipulators, 14% humanoid robotics, 14% mechanical design, 14% cognitive architectures, 14% service robotics.
- **Main simulated robots:** 9% Nao, 29% quadrotor, 29% wheeled vehicle, 29% Bioloid, 29% khepera/ e-puck/ thymio

4.5.6 Webots

Webots is a development environment used to model, program and simulate mobile robots developed by Cyberbotics Ltd.

- **Web:** http://www.cyberbotics.com

- **License:** Commercial or limited features free academic license
- **OS share:** 57% GNU/Linux, 29
- **Main API:** 71% C++
- **Main reason:** 29% best tool upon evaluation, 29% software already used in the lab, 14% boss choice, 14% official tool for a challenge, 14
- **Used for:** 43% mobile robotics, 43% multi-legged robotics, 14% humanoid robotics
- **Main simulated robots:** 29% KUKA LWR, 29% Lego Mindstorm, 29% wheeled vehicle.

4.5.7 OpenRave

OpenRave is an environment for simulating motion planning algorithms for robotics.

- **Web:** http://openrave.org/
- **License:** LGPL and Apache 2
- **OS share:** 100% GNU/Linux
- **Main API:** 83% python
- **Main reason:** 50% best tool upon evaluation, 33% colleagues using it, 17% boss choice.
- **Mostly used:** in USA (33%)
- **Used for:** 50% humanoid robotics, 50% service robotics
- **Main simulated robots:** 50% PR2

4.5.8 Robotran

Robotran is a software that generates symbolic models of multi-body systems, which can be analysed and simulated in Matlab and Simulink. It is developed by the Center for Research in Mechatronics(University Catholique de Louvain).

- **Web:** http://www.robotran.be/
- **License:** commercial and free non commercial license
- **OS share:** 80% Windows, 20
- **Main API:** 60% C
- **Main reason:** 40% software already used in the lab, 20% best tool upon evaluation, 20% developer, 20% opensource (free)
- **Used only:** in Belgium (40%) and Italy (60%)
- **Used for:** 60% humanoid robotics, 20% human motion analysis, 20% flying robots
- **Main simulated robots:** 60% Coman, 40% iCub

4.5.9 XDE

XDE is an interactive physics simulation software environment fully developed by CEA LIST.

- **Web:** http://www.kalisteo.fr/lsi/en/aucune/a-propos-de-xde
- **License:** Commercial and free non commercial license
- **OS share:** 60% GNU/Linux, 40% Windows
- **Main API:** 100% python
- **Middleware:** OROCOS
- **Main reason:** 40% boss choice, 40% software already used in the lab, 20% developer
- **Used only:** in France (100%)
- **Used for:** 40% humanoid robotics, 20% industrial manipulators, 20% numerical simulation of physical systems, 20% human motion analysis
- **Main simulated robots:** 40% industrial robots, 40% KUKA LWR, 20% iCub, 20% wheeled vehicle

Conclusions

With the growing interest of robotics for physical interaction, simulation is no longer a tool for offline computation and visualization, but is used in particular for rapidly prototyping controllers. That is why researchers stressed the importance of more realistic simulation, same code for both real and simulated robot, beside the availability of the source code.

To conclude, we overviewed simulation tools that are currently used in robotics. Each software inherits its specificities from the expected domains of application or the original application for which is was conceived, which results in a variety of tools with different features ranging from dynamic solver libraries to systems simulation software.

Chapter 5

Conception ,Implementation and Development

Introduction

For a vehicle to driver itself without a driver onboard, four interdependent functions are needed: navigation, situation analysis, motion planning and trajectory control.

- **Navigation:** The vehicle's ability to plan its route, nowadays achieved by using satellite navigation systems, typically GPS. In addition, the vehicle has to retrieve data related to road types, settings, terrains as well as weather conditions in order to have the most suitable route.

- **Situation Analysis:** The vehicle's ability to keep track of its surrounding environment, including all relevant objects and their movement. This function requires the use of different types of sensors, typically visual image, radar, ultrasonic sensors LIDAR (light detection and ranging), etc. The ultimate goal is to combine the data collected to make the vehicle continuously aware of its surroundings, so that it can decide what actions to conduct.

-**Motion Planning:** The vehicle's ability to determine the correct course of motion (speed, direction) within a certain pre-defined period of time, so that the vehicle keeps going its lane and its pre-set direction determined by navigation, without colliding with static and dynamic objects that are identified by situation analysis.

-**Trajectory Control:** The vehicle's ability to maintain driving stability in the events of changes in direction and speed planned by Motion Planning.

In this chapter will show differents methods for development system of autonomous car.

5.1 Environment description

The particular aim of this work is to enable the classification of the on-road obstacles according to their relative velocities, onto the categories of incoming, outgoing, and stationary, as a prerequisite for their avoidance in the context of autonomously guided cars.
The existing techniques used in the on-road obstacle detection may vary according to the definition of the obstacles. They might be classified into two categories .
The first one is related to the obstacles reduced to a specific object (vehicle, pedestrian, etc). In this case, the detection can be based on search for specific patterns, possibly supported by features such as texture, shape , symmetry, or the use of an approximate contour. The second category is used when the definition of the obstacles is more general. In this case, two methods are generally used.
The second , usage of a monocular camera based on an analysis of optical flow . This method requires rather huge calculation, and it is sensitive to vehicle movement.
Generally, a method for detecting both moving and static objects simultaneously is required because the static objects such as boxes can fall on the road in front of a car and they are dangerous too.
Actually, the algorithm of on-road obstacles detection should provide that:

1. The objects that are outside the road are eliminated.
2. Irregularities on the road surface that are not affecting the driving are not considered.
3. Static obstacles on the road are properly recognized in order to be avoided.
4. Vehicles on the road are detected in order to adjust own motion according to their relative distances and velocities.

The system of autonomous car driving may react appropriately in order to keep the motion of a car along the nominal trajectory relative to the road borders, simultaneously avoiding incoming and outgoing cars. Here, obstacles are defined as actual arbitrary objects protruding from the ground plane in the road area, both static and moving ones. Road markers in the road area (e.g., pedestrian crossings) as well as a number of objects outside the road region are considered as the obstacles of no interest.

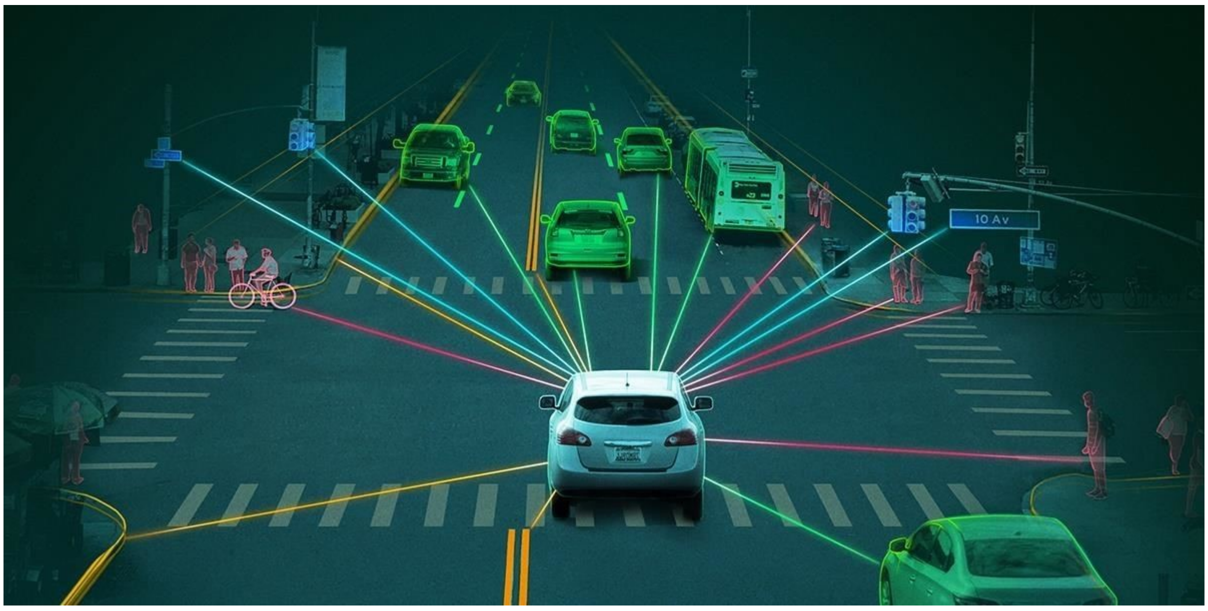


Figure 5.1: Environment of smart car.

5.2 Global architecture

we can describe global architecture with:

- Systems are made autonomous and smart by software.
- Must adaptation.
- Systems potentially operate in unpredictable and uncontrollable environments.
- Might share environment with other independent systems and even humans.
- Architecting and engineering collaborative systems is different from architecting and engineering a single system.
- Cloud solutions for autonomous driving basically consist of an information flow between the vehicle and a data processing platform within the cloud. There are also solutions in which the cars interact with each other but this would extend the scope of this blog post. Within this data flow all the sensor

information are send to the cloud platform in which two major processes occur:

  – Within the planning module, the sensory perception and localization data are mapped to plan the path for the car. An emergency module screens changes in those data to adjust the path or initiate an emergency brake.
  – The request module sends the data back to the car where the controller executes the received data.

- We can use webAPI for get open-information (e.g weather , status of road,.....).

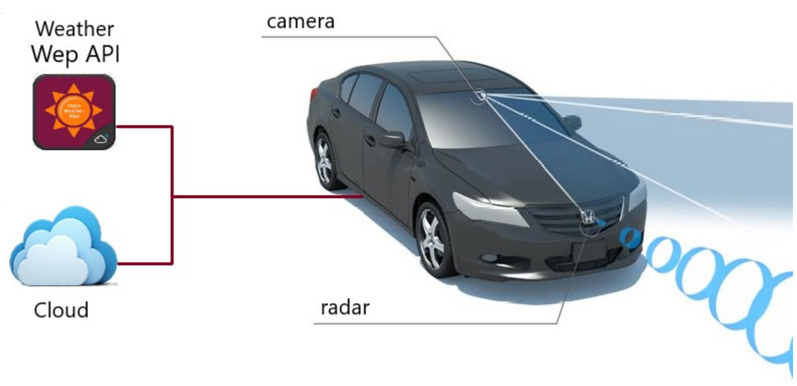


FIGURE 5.2: Global architecture.

5.3 Detailed architecture

We will now focus in this section as well as the detailed architecture of each component, and then develop a detailed delineation with 'UML' in which the overall structure of the system .

5.3.1 Use case diagram for autonomous-system of smart-Car

In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role (known in the Unified Modeling Language (UML) as an actor) and a system to achieve a goal. The actor can be a human or other external system. In systems engineering, use cases are used

at a higher level than within software engineering, often representing missions or stakeholder goals.
Use case analysis is an important and valuable requirement analysis technique that has been widely used in modern software engineering .

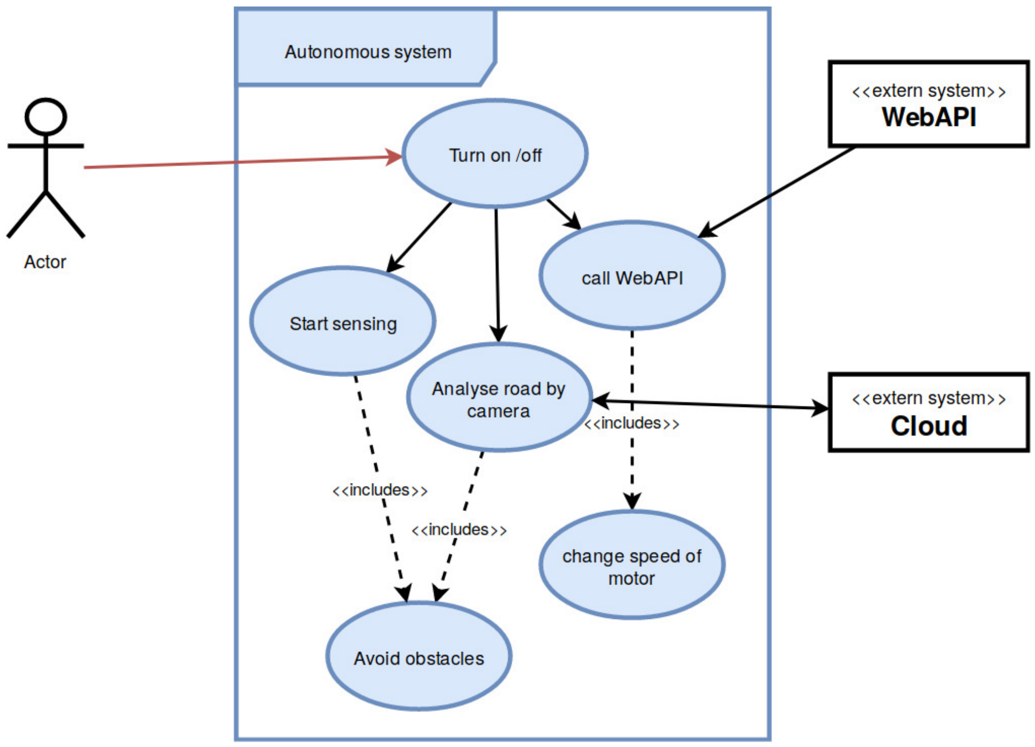


FIGURE 5.3: Use case diagram for autonomous-system of smartCar.

The presentation is illustrated with a use case model for AutonomousCar. A use case diagram for this system is shown in Fig.5.3.The AutonomousCar use a Actor(human..) and externals system (WebAPI, Cloud) for sharing different tasks. The object of the AutonomousCar is to allow:

- Actor(driver) run car.
- Call wepAPI for extraxt sensible infomation .
- AutonomousCar can sensing physical obstacle.
- AutonomousCar use camera for extract non-material obstacles with help Cloud because have high resources of calcul .
- AutonomousCar decide best option for avoid obstacle based on material obstacle and non-material obstacle.

5.3.2 Sequence Diagrams for autonomous-system of smart-Car

Are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when .

5.3.2.1 Sequence Diagram : connection scenario

The presentation is illustrated with a sequence model for connection .A sequence diagram for this scenario is shown in fig.5.4.

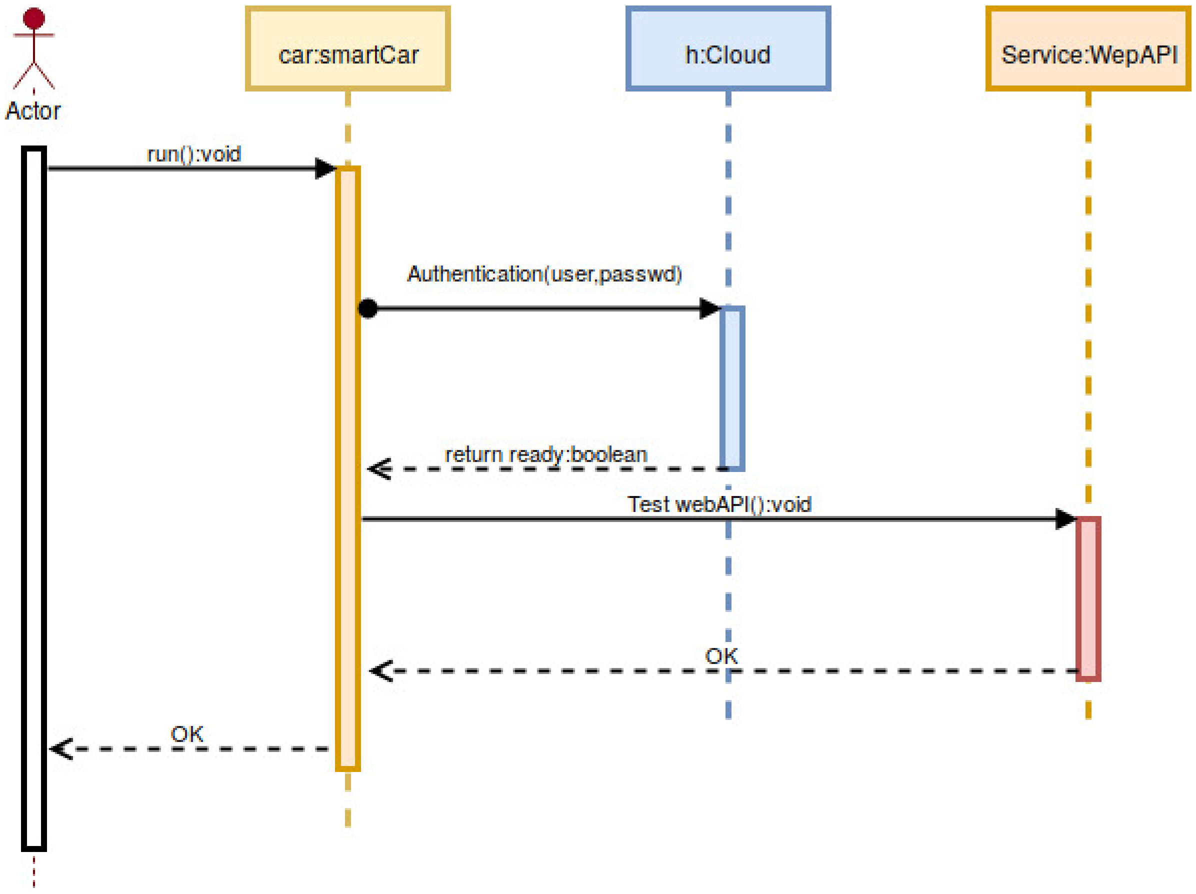


FIGURE 5.4: Sequence diagram for tasks of connection scenario.

Description: we can describe scenario as follow :

1. driver(Actor) run smartCar (Autonomous system) by method called ***run()***.
2. smartCar (Autonomous system) to connect with cloud based on two parameters user and password (by method called — ***Authentication (user ,passwd)***).
3. smartCar (Autonomous system) test of readiness of web API(by method called — ***TestwebAPI()***).

5.3.2.2 Sequence Diagram : initialization hardware scenario

The presentation is illustrated with a sequence model for initialization hardware.A sequence diagram for this scenario is shown in fig.5.5

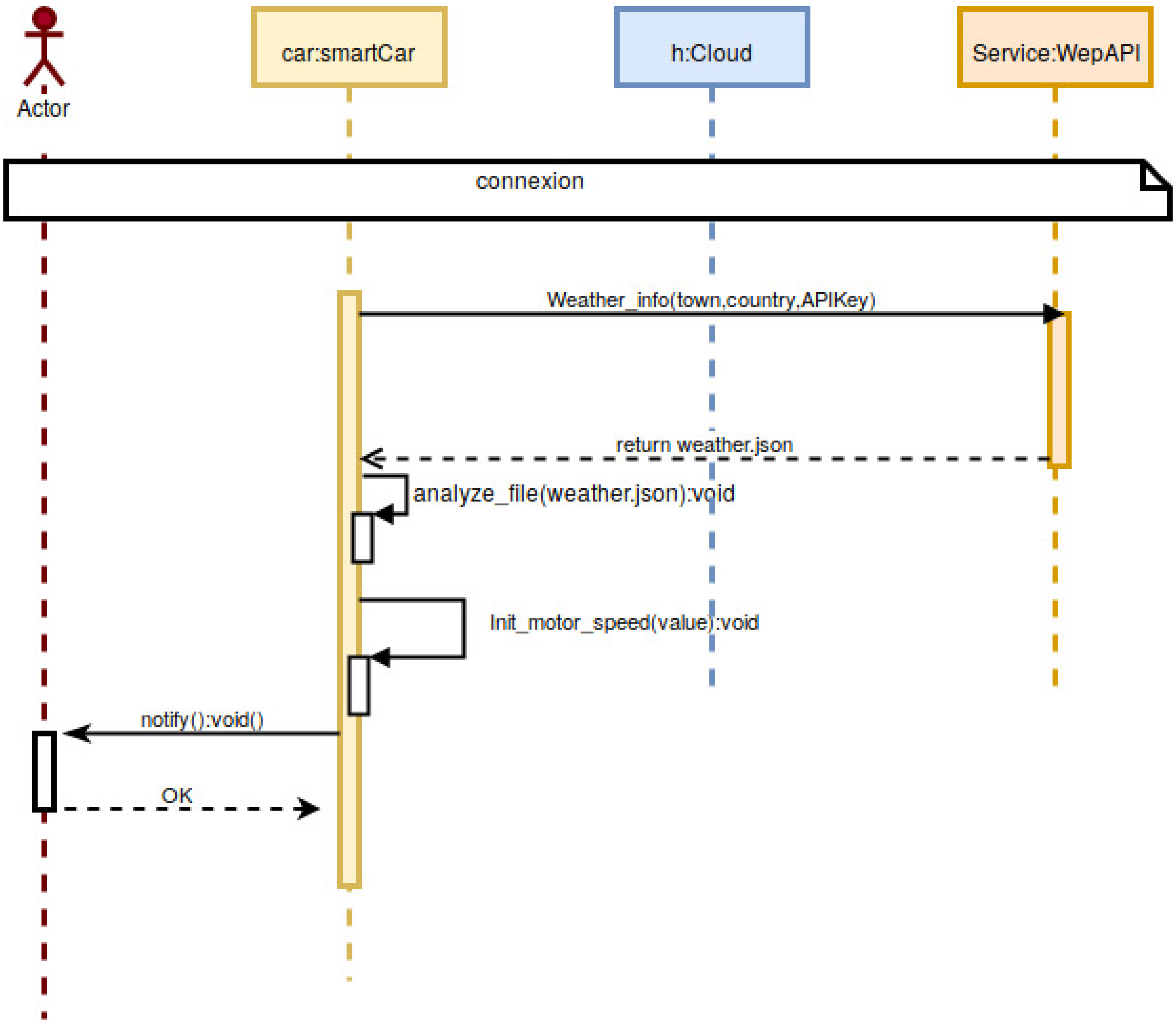

Actor
car:smartCar
h:Cloud
Service:WepAPI
connexion
Weather_info(town,country,APIKey)
return weather.json
analyze_file(weather.json):void
Init_motor_speed(value):void
notify():void()
OK

FIGURE 5.5: Sequence diagram for tasks of initialization hardware scenario.

Description: we can describe scenario as follow :

1. smartCar (Autonomous system) to require information from webAPI and receive file **weather.json** based on three parameters : **town** (e.g biskra) , **country** (e.g Algeria) , **APIKey**(given by the company) (by method called — ***Weather_info(town,country,APIKey)***).
2. smartCar (Autonomous system) analyze file **weather.json** and extract information (by method called — ***analyze_file(weather.json)***).
3. smartCar (Autonomous system) initialization hardware (based on information in file **weather.json**).
4. smartCar (Autonomous system) notify driver(Actor) in any action.

5.3.2.3 Sequence Diagrams : avoid obstacle scenario

The presentation is illustrated with a sequence model for avoid obstacle .A sequence diagram for this scenario is shown in fig.5.6.
Description: we can describe scenario as follow :

1. smartCar (Autonomous system) tracking objects by ip camera and send capture to cloud by method ***streaming_video(ip_camera)***.
2. Cloud analyze frame (image) received from smartCar and extract non-material obstacle (e.g traffic sign , passersby) by function ***analyze_image()***.
3. smartCar (Autonomous system) surveying road and extract material obstacle (e.g cars. . .) by function ***sensing_physical_obstacle()***.
4. smartCar (Autonomous system) allow to choose best decision for avoid obstacle based on result ***analyze_image()*** and ***sensing_physical_obstacle()***.

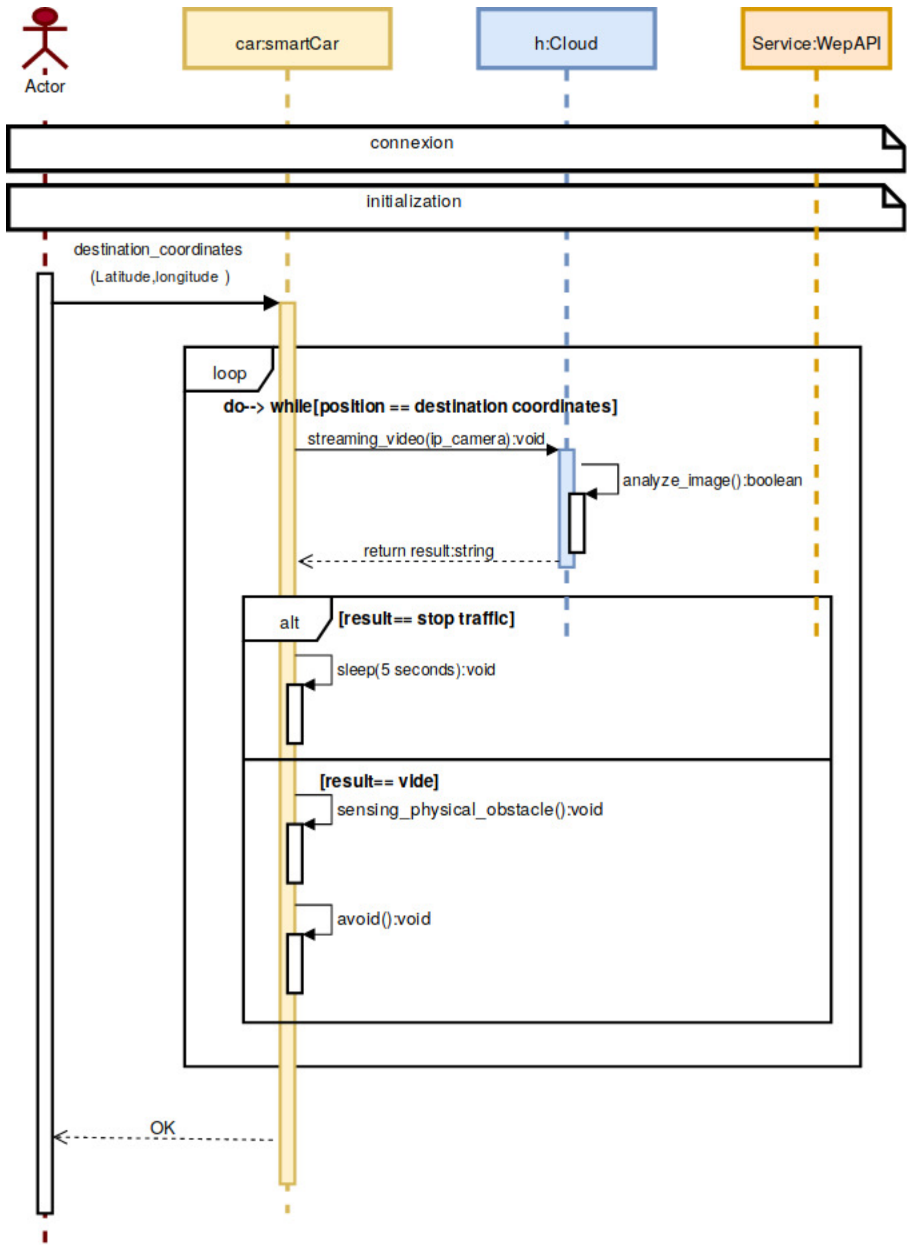


FIGURE 5.6: Sequence diagram for tasks of avoid obstacle scenario.

5.4 Protocol used

We will display and illustrate all protocol used in this project.

5.4.1 I2C Protocols

5.4.1.1 Principal

I2C (Inter-Integrated Circuit) is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs (electrically erasable programmable read-only memory), A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. It was invented by Philips and now it is used by almost all major IC manufacturers. Each I2C slave device needs an address[37].

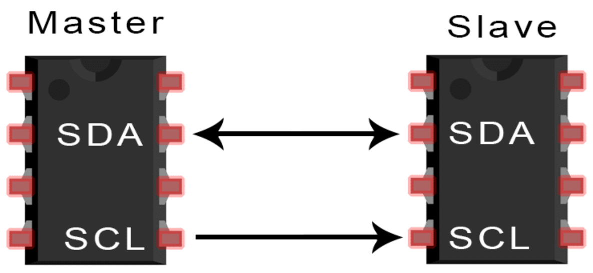


FIGURE 5.7: I2C wires.

SDA (Serial Data) – The line for the master and slave to send and receive data.

SCL (Serial Clock) – The line that carries the clock signal.

I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).
I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

5.4.1.2 How I2C works

With I2C, data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame [37]:

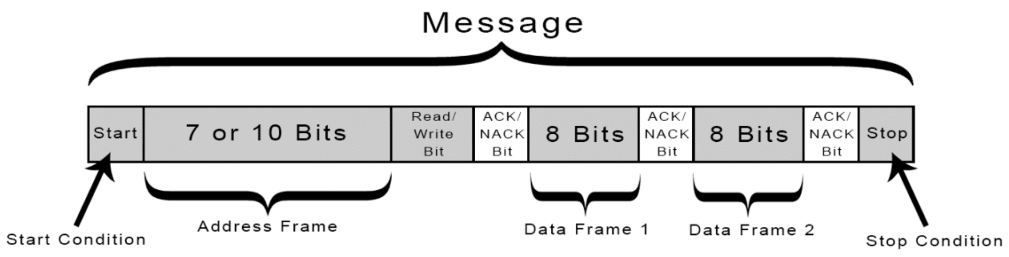


FIGURE 5.8: I2C data frame [37].

Start Condition: The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.
Stop Condition: The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.
Address Frame: A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device[37].

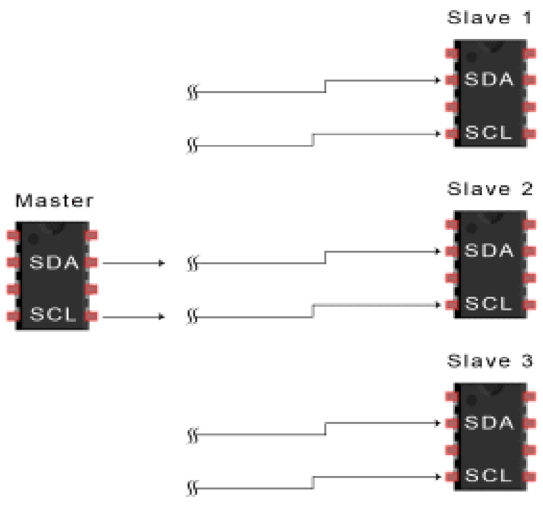


FIGURE 5.9: I2C communication.

5.4.2 RTSP Protocol

RTSP stands for Real Time Streaming Protocol, a network protocol for streaming the videos in real-time.
Figuratively, RTSP acts as a rail track for transporting videos data from point A (camera) to point B (VLC Player/RTSP viewers/RTSP Client or software).
In such sense, an RTSP IP camera is capable of streaming live videos on RTSP-compatible media players .
When IP cameras support RTSP streaming protocol and ONVIF, users could use RTSP camera viewers/software such as IP Cam Viewer, iSpy or web browsers to view live video stream easily without using its companion mobile apps or desktop software.
Meanwhile, users can use to view RTSP IP video stream on TV thanks to RTSP streaming protocol[38].

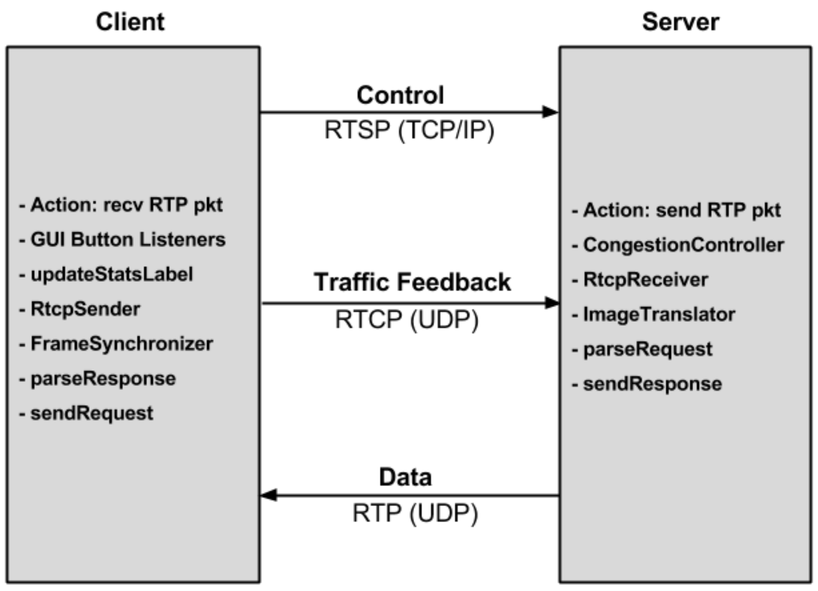


FIGURE 5.10: RTSP Protocol.

5.5 Technologies and structures used

We will display and illustrate all technologies and structures used in this project.

5.5.1 Web API

5.5.1.1 Definition

In contrast, an API specifies how software components should interact with each other. It is a set of protocols and routines, and it's responses are generally returned as JSON or XML data. APIs can use any type of communication protocol, and are not limited in the same way a web service is[39].

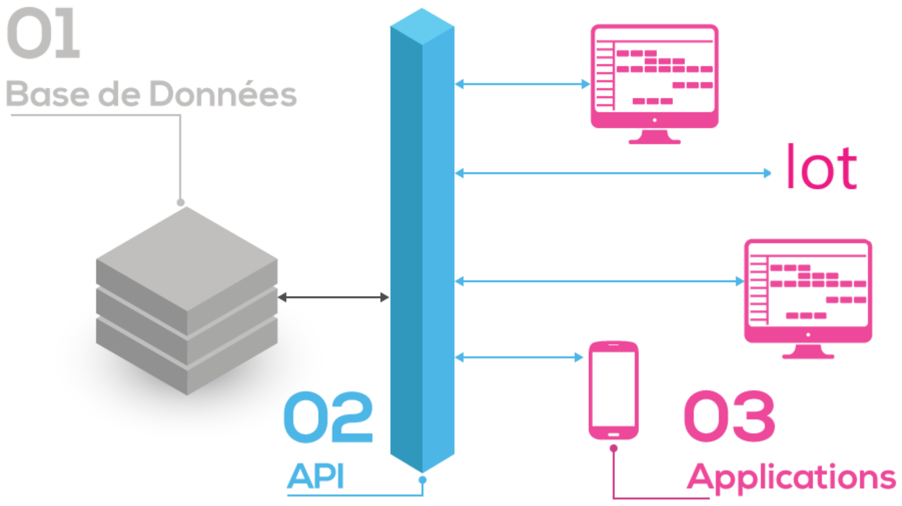


FIGURE 5.11: Web API.

5.5.1.2 Why is a web API not a web service?

In contrast, a typical Web API specifies how software components should interact with each other using the web's protocol (HTTP) as the go-between. The client doesn't need to know what procedure to call on the server. Instead, it uses a set of commands (called "verbs") that are built into HTTP and when the command arrives on the other end, it's up to the receiving system to know what to do with it. For example, the HTTP verb that's typically used to retrieve data is "GET".

When HTTP is used to abstract systems from one another, the systems are considered to be more loosely coupled (when compared to web services) and therefore the entire system is considered less brittle. Another advantage of web APIs (often referred to as RESTful APIs) is flexibility. The client system (usually called the "consumer") and the serving system (the "provider") are so independent of one another, that they can each use different languages (Java, Python, Ruby, etc.) for their part of the overall implementation. Additionally, the data payloads can be of multiple types such as JSON or XML.
RESTful APIs most typically use the web's communication protocol (again, HTTP), but are not limited in the same way a web service is. For example, CoAP, an HTTP-like protocol that's common to the Internet of Things, is also considered to be RESTful[40].

5.5.2 Cloud

The Cloud infrastructure and its extensive set of Internet-accessible resources has potential to provide significant benefits to robots and automation systems. We consider robots and automation systems that rely on data or code from a network to support their operation, i.e., where not all sensing, computation, and memory is integrated into a standalone system. This survey is organized around four potential benefits of the Cloud:

1. **Big Data:** access to libraries of images, maps, trajectories, and descriptive data.
2. **Cloud Computing:** access to parallel grid computing on demand for statistical analysis, learning, and motion planning.
3. **Collective Robot Learning:** robots sharing trajectories, control policies, and outcomes.
4. **Human Computation:** use of crowdsourcing to tap human skills for analyzing images and video, classification, learning, and error recovery.

The Cloud can also improve robots and automation systems by providing access to:

(a) datasets, publications, models, benchmarks, and simulation tools.

(b) open competitions for designs and systems.

(c) open-source software[41].

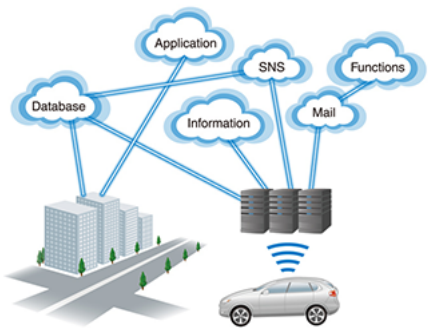


FIGURE 5.12: Cloud and Autonomous-Car.

5.6 Setup software and hardware

5.6.1 Driver motor

5.6.1.1 Driver motor L298N

The L298N is an integrated monolithic circuit in a 15- lead Multiwatt and PowerSO20 packages. It is a high voltage , high current dual full-bridge driver designed to accept standard TTL logic level sand drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the in-put signals .The emitters of the lower transistors of each bridge are connected together rand the corresponding external terminal can be used for the connection of an external sensing resistor. An additional Supply input is provided so that the logic works at a lower voltage.

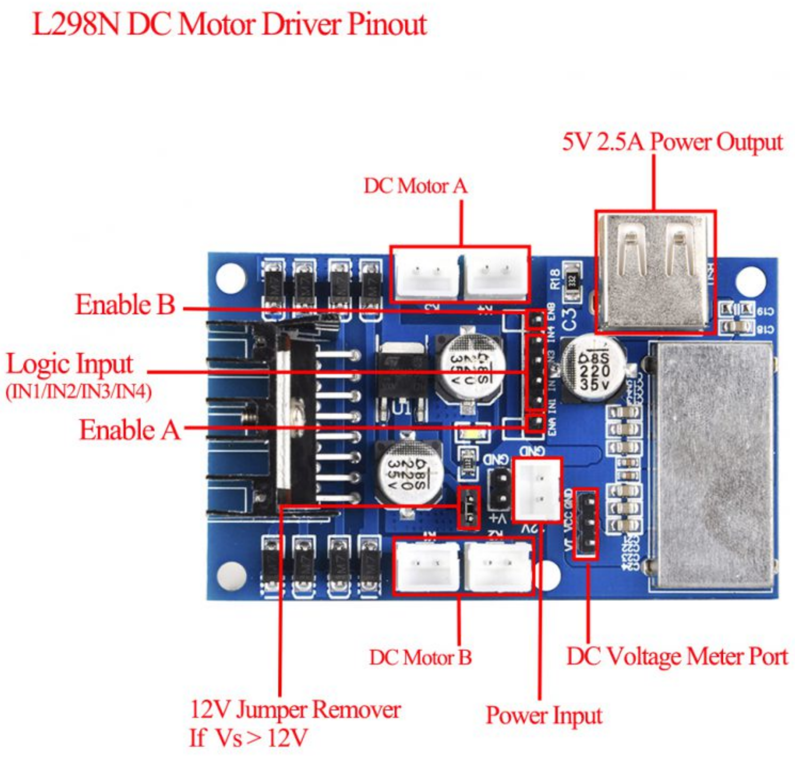


FIGURE 5.13: Driver motor L298N.

5.6.1.2 Features

1. High operating voltage, which can be up to 40 volts.
2. Large output current, the instantaneous peak current can be up to 3A.
3. With 25W rated power.
4. Two built in H-bridge, high voltage, large current, full bridge driver, which can be used to drive DC motors, stepper motors, relay coils and other inductive loads.
5. Using standard logic level signal to control.
6. Able to drive a two-phase stepper motor or four-phase stepper motor, and two-phase DC motors.
7. Adopt a high-capacity filter capacitor and a freewheeling diode that protects devices in the circuit from being damaged by the reverse current of an inductive load, enhancing reliability.

8. The module can utilize the built-in stabilivolt tube 78M05 to obtain 5v from the power supply. But to protect the chip of the 78M05 from damage, when the drive voltage is greater than 12v, an external 5v logic supply should be used.

9. Drive voltage: 5-35V; logic voltage: 5V.

10. PCB size: 4.2 x 4.2 cm.

5.6.1.3 Motor speed control

we can controle speed of motor with using PWM gpio pins.With the use of PWM, we can simulate varying levels of output energy to an electrical device. At any given time, the digital output will still be on or off. But, we can pulse the output with varying widths to control the amount of effective output. Shorter on cycles compared to the off cycle will deliver lower overall output. Longer on times relative to the off time deliver higher overall output. This relationship between the on and off duration is called "duty cycle" and is measured in percent of on time compared to the off time. Thus a 25% duty cycle will have an on time 25% of the total cycle with 75% off. A 50% duty cycle will have equal on and off times. With PWM we can vary the duty cycle from 0% to 100%.

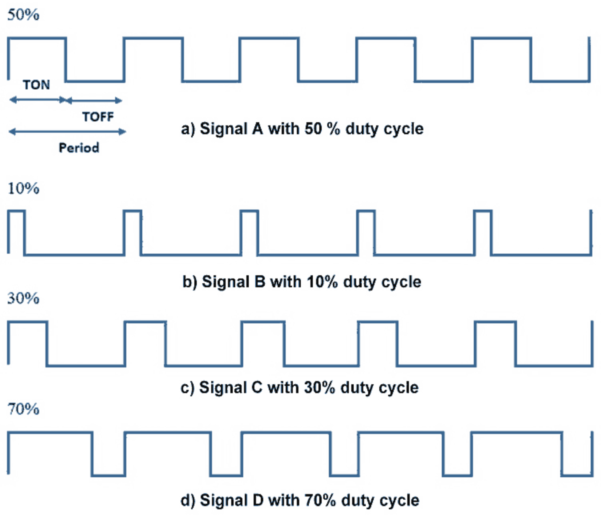

50%
TON
TOFF
Period
a) Signal A with 50 % duty cycle
10%
b) Signal B with 10% duty cycle
30%
c) Signal C with 30% duty cycle
70%
d) Signal D with 70% duty cycle

FIGURE 5.14: Pulse Width Modulation technique.

Code source

we use two pin for control motors :

Enable A : for two motor on right.(see fig.5.13)

Enable B : for two motor on left.(see fig.5.13)

```
import numpy as np
import RPi.GPIO as GPIO
.
.
.
def(val1,val2):
  enA.p.ChangeDutyCycle(val1)
  enB.p.ChangeDutyCycle(val1)
.
.
.
```

Listing 5.1: motor speed python

5.6.1.4 Motor direction control

we will use four GPIO (IN1/IN2/IN3/IN4)(see fig.5.13) as follow:

IN1	IN2	IN3	IN4	The state of DC motor
1	0	1	0	Forward
1	0	0	1	Rotate clockwise
0	1	1	0	Rotate Counterclockwise
0	0	10	0	Brake

Table 5.1: Motor direction control

```
import numpy as np
import RPi.GPIO as GPIO
.
.
.
def forward():
  GPIO.output(in1, 1)
  GPIO.output(in2, 0)
  GPIO.output(in3, 1)
  GPIO.output(in4, 0)
def brake():
  GPIO.output(in1, 0)
  GPIO.output(in2, 0)
  GPIO.output(in3, 0)
  GPIO.output(in4, 0)
def right():
  GPIO.output(in1, 1)
  GPIO.output(in2, 0)
  GPIO.output(in3, 0)
  GPIO.output(in4, 1)
def left():
  GPIO.output(in1, 0)
  GPIO.output(in2, 1)
  GPIO.output(in3, 1)
  GPIO.output(in4, 0)
.
.
.
```

Listing 5.2: Motor direction python

5.6.2 Ultrasonic sensor

5.6.2.1 Ultrasonic sensor HC-SR04

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.

5.6.2.2 Features

Here's a list of some of the HC-SR04 ultrasonic sensor features and specs:

1. Power Supply :+5V DC.
2. Quiescent Current : ¡2mA.
3. Working Current: 15mA.
4. Effectual Angle: ¡15deg.

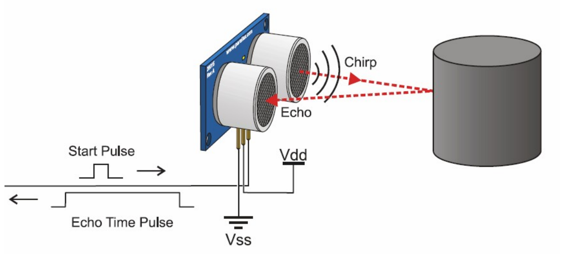
5. Ranging Distance : 2cm – 400 cm/1 – 13ft.
6. Resolution : 0.3 cm.
7. Measuring Angle: 30 degree.
8. Trigger Input Pulse width: 10uS.
9. Dimension: 45mm x 20mm x 15mm.

5.6.2.3 How Does it Work?

The ultrasonic sensor uses sonar to determine the distance to an object. Here's what happens:

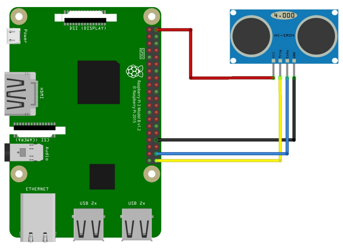
1. The transmitter (trig pin) sends a signal: a high-frequency sound.
2. When the signal finds an object, it is reflected and...
3. ... the transmitter (echo pin) receives it.

The time between the transmission and reception of the signal allows us to calculate the distance to an object. This is possible because we know the sound's velocity in the air.



(A) Ultrasonic principle

$\Rightarrow$



(B) Connect the Raspberry Pi with Ultrasonic

FIGURE 5.15: Work principle of ultrasonic.

Code source

```
import numpy as np
import RPi.GPIO as GPIO
import time
# initialization GPIO
.
.
.
.
def measure(GPIO_ECHO,GPIO_TRIGGER):
# set Trigger to HIGH
GPIO.output(GPIO_TRIGGER, True)

# set Trigger after 0.01ms to LOW
time.sleep(0.00001)
GPIO.output(GPIO_TRIGGER, False)

StartTime = time.time()
StopTime = time.time()

# save StartTime
while GPIO.input(GPIO_ECHO) == 0:
StartTime = time.time()

# save time of arrival
while GPIO.input(GPIO_ECHO) == 1:
StopTime = time.time()

# time difference between start and arrival
TimeElapsed = StopTime - StartTime
# multiply with the sonic speed (34300 cm/s)
# and divide by 2, because there and back
distance = (TimeElapsed * 34300) / 2

return distance
.
.
.
.
.
```

Listing 5.3: Ultrasonic measure python

5.6.3 LCD

An LCD is an electronic display module which uses liquid crystal to produce a visible image. The 16 × 2 LCD display is a very basic module commonly used in DIYs and circuits. The 16 × 2 translates o a display 16 characters per line in 2 such lines.

5.6.3.1 How connect LCD?

Connecting an LCD to your Raspberry Pi will spice up almost any project, but what if your pins are tied up with connections to other modules? No problem, just connect your LCD with I2C, it only uses two pins (well, four if you count the ground and power).

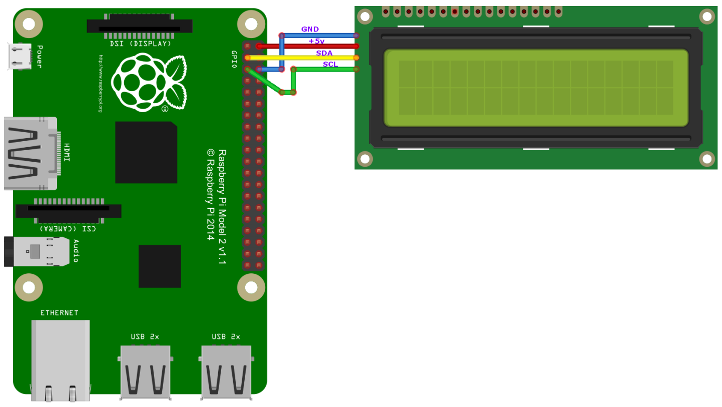


FIGURE 5.16: Connect the Raspberry Pi with LCD using I2C Adapter.

code source

```
import RPi.GPIO as GPIO
import lcddriver
# initialization GPIO
.
lcd = lcddriver.lcd()
lcd.lcd_clear()
lcd.lcd_display_string("Position.....",1)
lcd.lcd_display_string("Weather......",2)
.
```

LISTING 5.4: Ultrasonic measure python

5.6.4 Camera

Initial testing was conducted with Samsung 13Mpx ip camera. The device was chosen due to its alleged high capabilities, especially the 13MP camera. Ip camera was a every 1 second tries to take 25 a picture.

5.6.4.1 How can camera detect non-material obstacle?

Obstacle detection is the process of using camera, data structures, and algorithms to detect objects or terrain types that impede motion.
Obstacle detection and hazard detection are synonymous terms, but are sometimes applied in different domains; for example, obstacle detection is usually applied to ground vehicle navigation. Obstacle detection is a system problem that encompasses sensors that perceive the world, world models that represent the sensor data in a convenient form, mathematical models of the interaction between objects and the vehicle, and algorithms that process all of this to infer obstacle...
We will simulate to detect traffic light with using two method haar-cascade (for detect image) and HSV (for detect color) (see chapter 3).

```python
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('/home/arios/Desktop/
    Traffic_Light_file.xml')
img = cv2.imread('/home/arios/Desktop/images.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
  img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
  roi_gray = gray[y:y+h, x:x+w]
  roi_color = img[y:y+h, x:x+w]
.
.
.
```

Listing 5.5: Detect object with using haar-cascade python

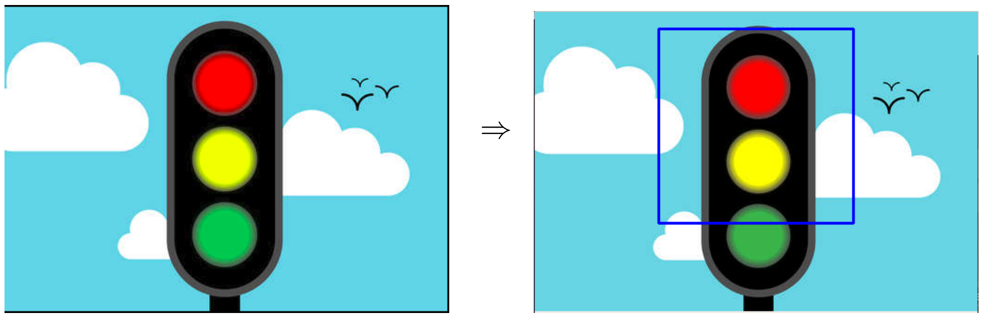

(A) Original image (B) Result of detect

FIGURE 5.17: Detect object with using haarcascade.

```
.
.
hsv_frame = cv2.cvtColor(roi_color, cv2.COLOR_BGR2HSV)
low_red = np.array([0,100,100])
high_red = np.array([20,255,255])
red_mask = cv2.inRange(hsv_frame, low_red, high_red)
red = cv2.bitwise_and(roi_color, roi_color, mask=red_mask)
cv2.imshow("Original", img)
cv2.imshow("detect", roi_color)
cv2.imshow('color',red)
cv2.waitKey(0)
cv2.destroyAllWindows()
.
.
.
```

LISTING 5.6: Detect color with using HSV python

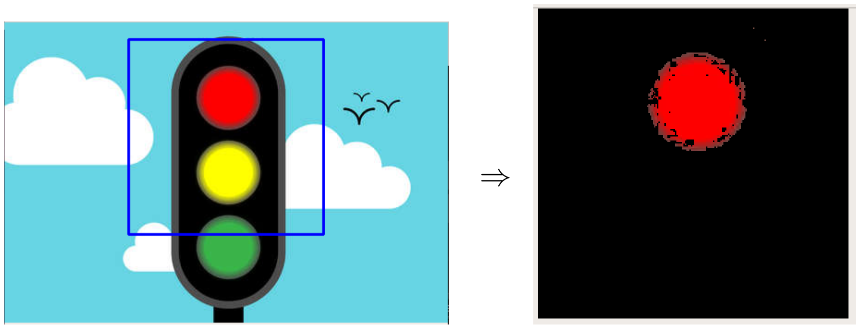

(A) Original image (B) Result of detect color red

FIGURE 5.18: Detect object with using haarcascade.

5.7 Global pseudo Algorithm

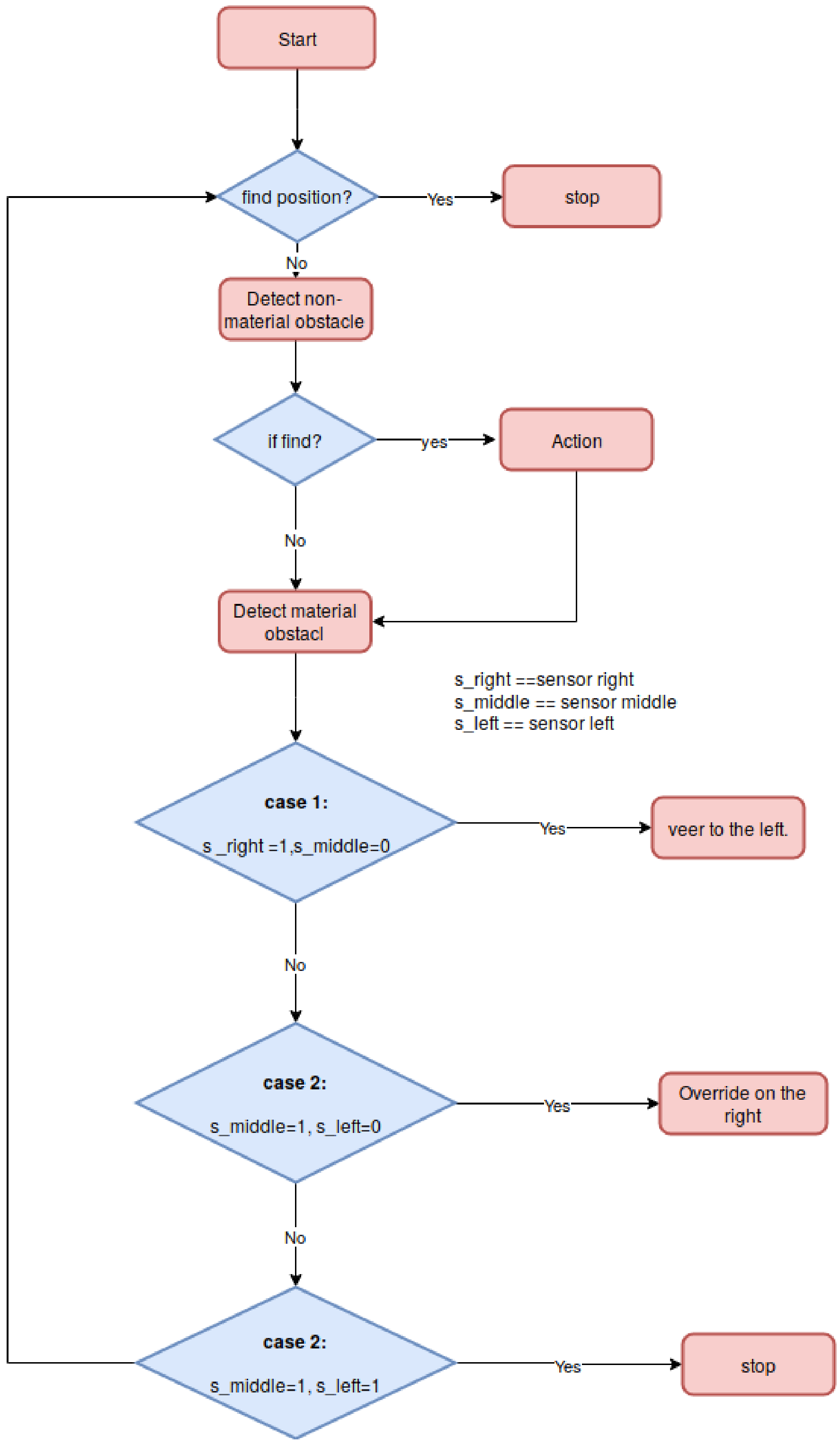


FIGURE 5.19: Global pseudo Algorithm.

Description

The fig.5.19 describes the self-driving system of the project as follows:

- Find path of navigation.
- Detect non-material obstacle (e.g traffic light , sign stop....) and decision action.
- Obstacle non-material has priority on obstacle material .
- detect material obstacle allow separates three tasks as follow:
    1. Track the road..
    2. Overtake obstacle.
    3. Stop (in case of obstruction.)
- The loop is repeated until reaching the goal.

5.8 Global circuit

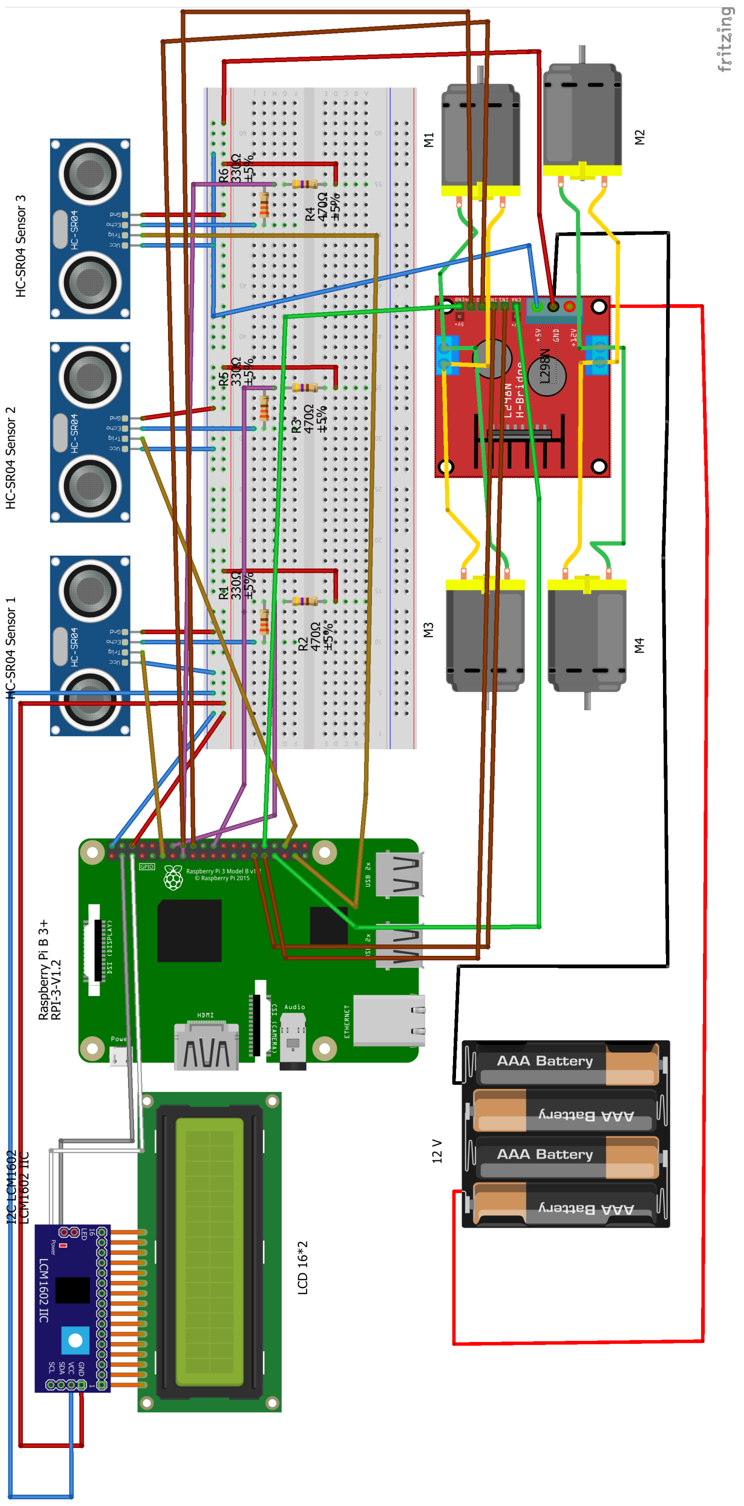


FIGURE 5.20: Electronic circuits.

We used electrical components as follow :

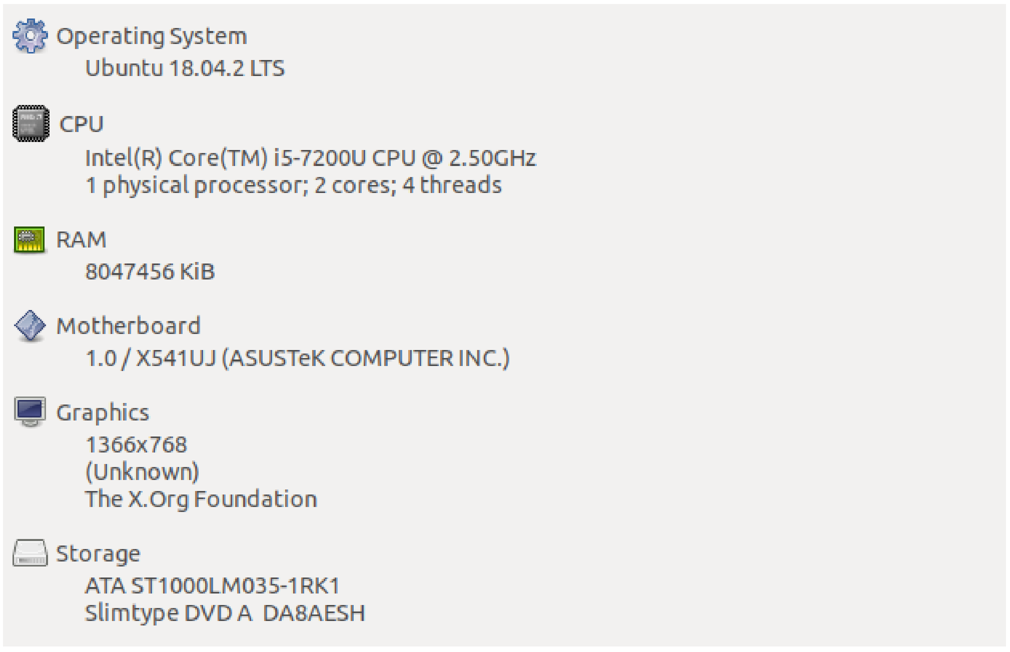
- 1 × Raspberry Pi B 3+
- 1 × driver motor L298N
- 4 × motors
- 3 × Ultrasonic HC-SR04
- 3 × Resistor 330 Ohm
- 3 × Resistor 470 Ohm
- 1 × LCD 16 × 2
- 1 × I2C Adapter
- 1 × Battery

5.9 Environment for Development

We will describe all hardwares and softwares using in developement this project.

5.9.1 Environment of Hardware and Software

for realize our system, we have a PC I3 equipped with Ubuntu 18.04.2 LTS (64bits) which is described with the following Figure:



Operating System
Ubuntu 18.04.2 LTS

CPU
Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
1 physical processor; 2 cores; 4 threads

RAM
8047456 KiB

Motherboard
1.0 / X541UJ (ASUSTeK COMPUTER INC.)

Graphics
1366x768
(Unknown)
The X.Org Foundation

Storage
ATA ST1000LM035-1RK1
Slimtype DVD A DA8AESH

FIGURE 5.21: Environment of Hardware and Software.

5.9.2 Libraries and languages of programmation

PYTHON

Python is a widely used general-purpose, high-level programming language. Its syntax allows the programmers to express concepts in fewer lines of code when compared with other languages like C, C++or java.

RPi.GPIO Python Library

The RPi.GPIO Python library allows you to easily configure and read-write the input/output pins on the Pi's GPIO header within a Python script . This package is not shipped along with Raspbian.

OpenCV

It (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision.

It has over 2500optimized algorithms, including both a set of classical algorithms and the state of the art algorithms in Computer Vision, which can be used for image processing, detection and face recognition, object identification, classification actions, traces, and other functions .

This library allows these features be implemented on computers with relative ease, provide a simple computer vision infrastructure to prototype quickly sophisticated applications.

The library is used extensively by companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, and startupsarea as Applied Minds, Video Surf and Zeitera. It is also used by many research groups and government .

5.10 Result

The figure 5.22 represents the final form of the robot model :

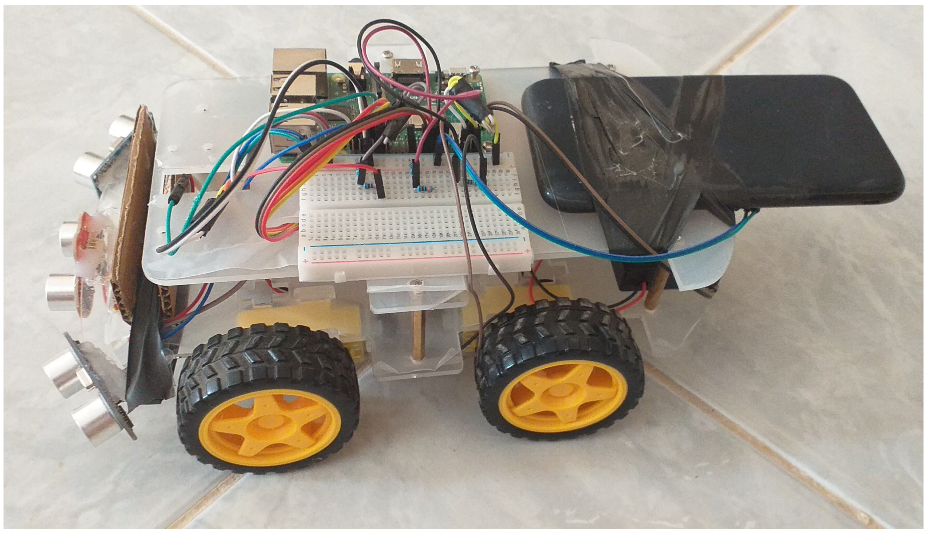


FIGURE 5.22: Prototype of autonomous car.

The figure 5.23 represents environment of work robot and different objects find in this environment (e.g : cars , signs of road...):

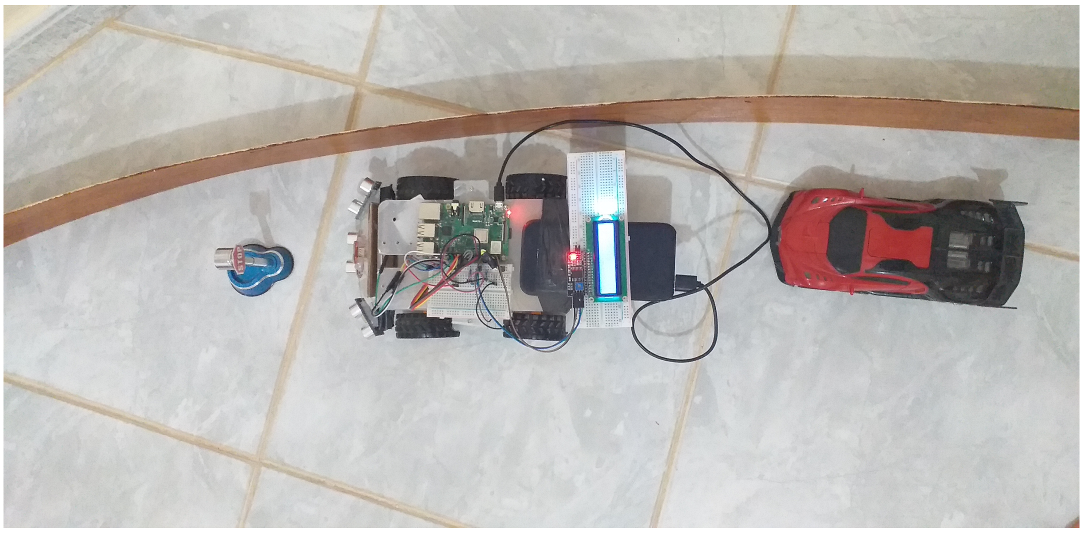


FIGURE 5.23: Environment of autonomous car.

The figure 5.24 represents show weather notification a based of information web API:

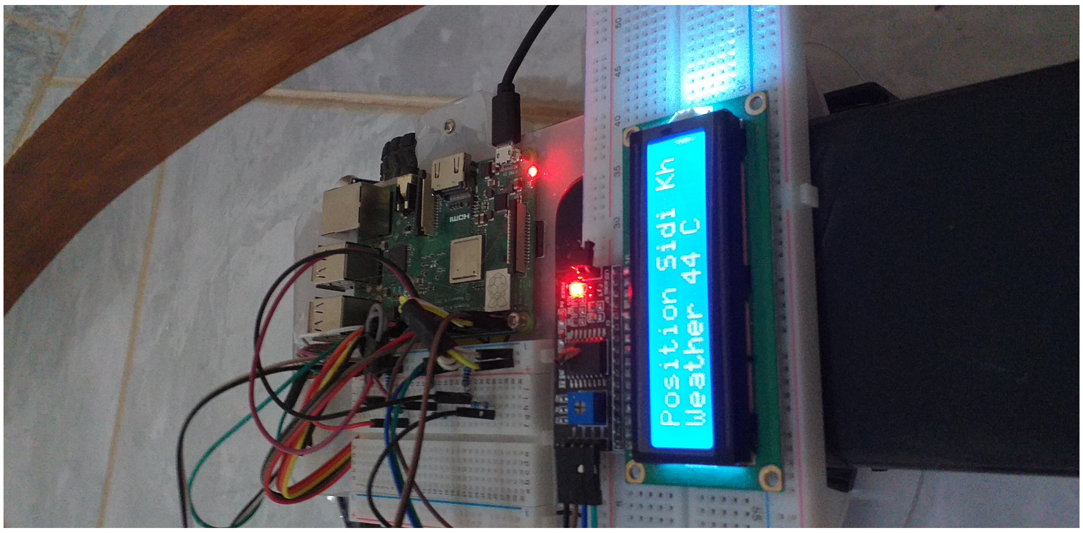


FIGURE 5.24: weather notification.

The figure 5.25 represents Autonomous car overshoot a car a based of analyze environment and information of sensors:

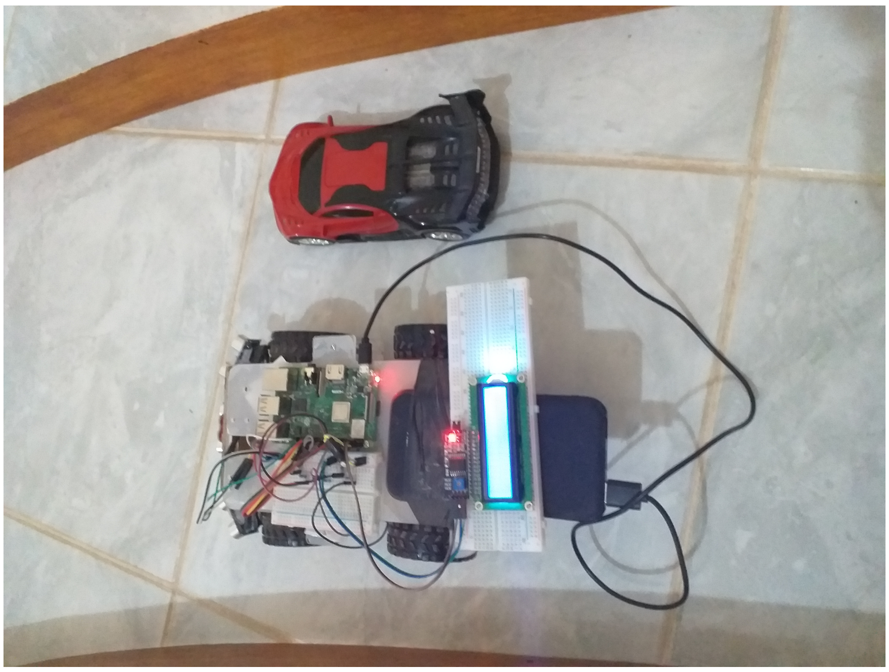


FIGURE 5.25: Autonomous car overshoot a car.

The figure 5.26 and 5.27 represents Autonomous car detect stop sign a based on technology Deep Learning:

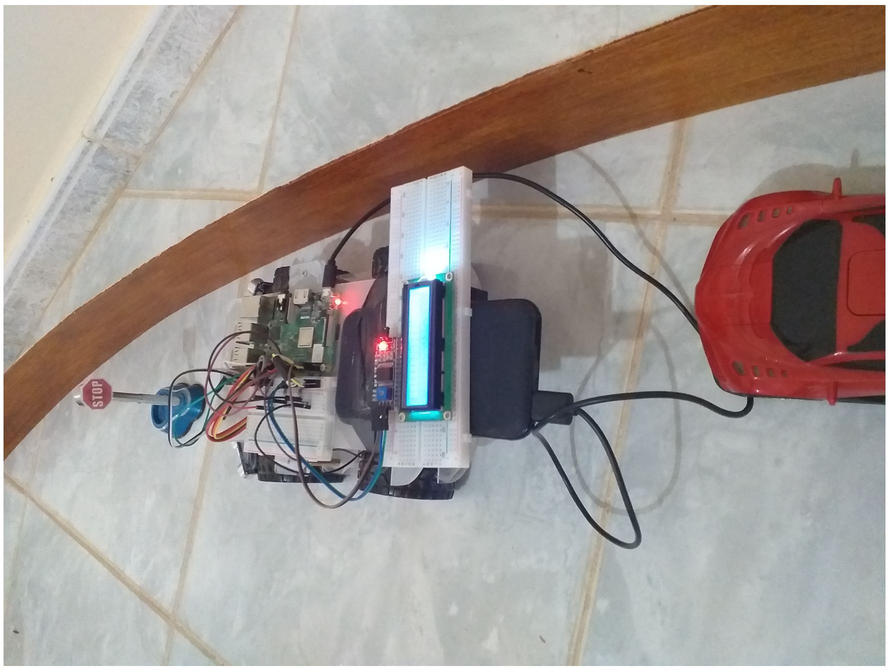


FIGURE 5.26: Autonomous car detect stop sign.

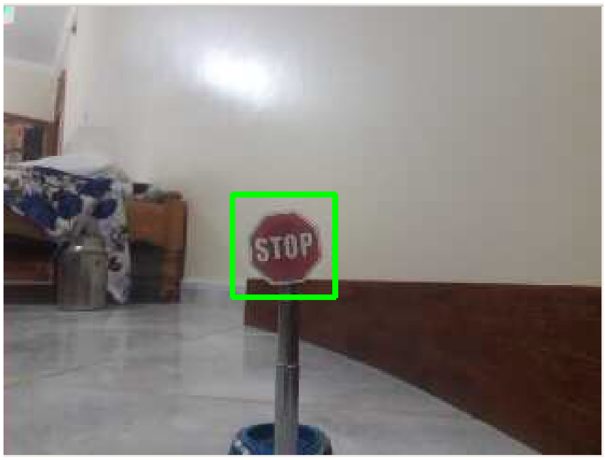


FIGURE 5.27: Result of detection.

The figure 5.28 and 5.29 represents Autonomous car can't detect sign (Maximum speed 80 km/h) cause Autonomous car not train on this sign.

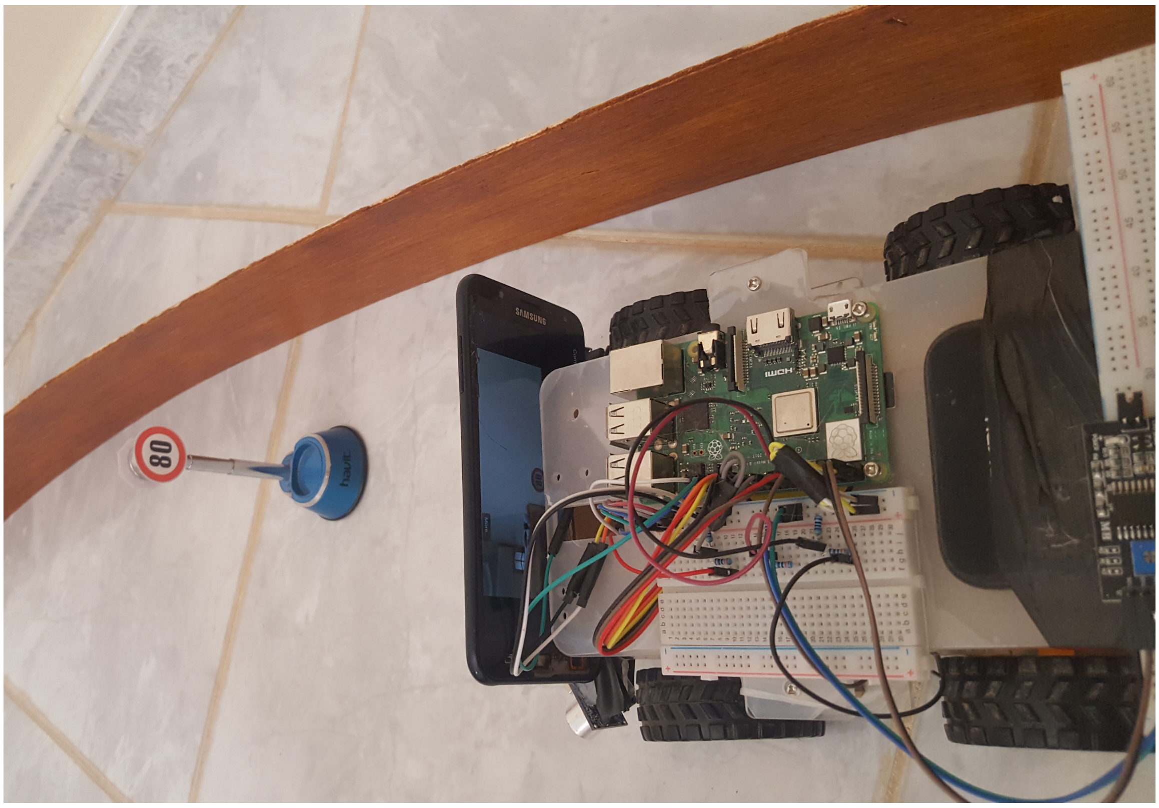


FIGURE 5.28: Autonomous car try detect sign Maximum speed 80 km/h.

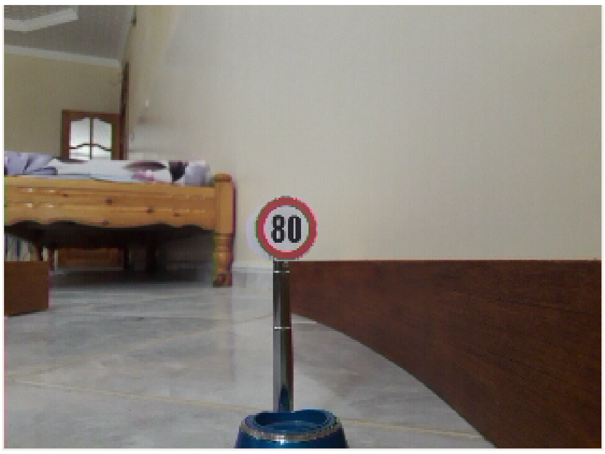


Figure 5.29: Result of failed detection .

Conclusion

Much of classical robotics focused on reasoning, optimal control and sensor processing given models of the robot and its environment. While this approach is successful for many industrial applications, it falls behind the more ambitious goal of robotics as a test platform for our understanding of artificial intelligence. This chapter presented the robot self-learning strategy under explicit feedback. The experiment results show the feasibility and the stability of the proposed method. The robot can complete navigation tasks safely in an unpredicted dynamic environment and becomes a truly intelligent system with strong self-learning and adaptive abilities.

Conclusion

This thesis contributes to the intelligent control design in order to achieve learning mobile robots during autonomous navigation. The results have confirmed the effectiveness and robustness of all methods that we have proposed, and in addition, the application of these methods can largely relieve the tedious job of pre-programming a robot manually.

1. Contributions

The goal of this thesis is focused on empowering mobile robots with intelligent learning and adaptive capabilities that respond to changing environments. Contributions are developed in three areas:

1. We have demonstrated the optimal utilization of sensors in navigation in an environment that is not absolute(Chapter 2).
2. We have proposed the method of effective learning (Chapter 3). This method produces a stable and enjoyable performance with an acceptable number of results . It also saves a lot of time during the learning process.
3. We introduced step-by-step how to build Autonomous-systems through the creation of a robot that works in a semi-real environment(Chapter 2).

2. Perspective

Cannot meet the future requirements, especially when interaction with human is needed. The aim of this thesis has been focused on empowering a mobile robots with an intelligent learning and adaptive ability that can cater to invariable environments.

All the above robot learning methods are model-free algorithms that are more suitable in real applications since a model of an unknown world is impossible to acquire.

Bibliography

[1] Jari Kaivo-Oja, Steffen Roth, and Leo Westerlund. Futures of robotics. human work in digital transformation. *International Journal of Technology Management*, 73(4):176–205, 2017.

[2] Mordechai Ben-Ari and Francesco Mondada. *Elements of robotics*. Springer, 2018.

[3] Christopher Holleman and Lydia E Kavraki. A framework for using the workspace medial axis in prm planners. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1408–1413. IEEE, 2000.

[4] Tamás Haidegger, Marcos Barreto, Paulo Gonçalves, Maki K Habib, Sampath Kumar Veera Ragavan, Howard Li, Alberto Vaccarella, Roberta Perrone, and Edson Prestes. Applied ontologies and standards for service robots. *Robotics and Autonomous Systems*, 61(11):1215–1223, 2013.

[5] Guang-Zhong Yang, Jim Bellingham, Pierre E Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of science robotics. *Science Robotics*, 3(14):eaar7650, 2018.

[6] Leonardo Ricotti, Barry Trimmer, Adam W Feinberg, Ritu Raman, Kevin K Parker, Rashid Bashir, Metin Sitti, Sylvain Martel, Paolo Dario, and Arianna Menciassi. Biohybrid actuators for robotics: A review of devices actuated by living cells. *Science Robotics*, 2(12):eaaq0495, 2017.

[7] Minjoon Park, Jaechan Ryu, Wei Wang, and Jaephil Cho. Material design and engineering of next-generation flow-battery technologies. *Nature Reviews Materials*, 2(1):16080, 2017.

[8] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[9] Corinne Cath, Sandra Wachter, Brent Mittelstadt, Mariarosaria Taddeo, and Luciano Floridi. Artificial intelligence and the 'good society': the us, eu, and uk approach. *Science and engineering ethics*, 24(2):505–528, 2018.

[10] Luciano Floridi and Mariarosaria Taddeo. What is data ethics?, 2016.

[11] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Transparent, explainable, and accountable ai for robotics. *Science Robotics*, 2(6): eaan6080, 2017.

[12] Horst Czichos. Testing. In *Measurement, Testing and Sensor Technology*, pages 25–42. Springer, 2018.

[13] D Patranabi. *Sensors and Tranducers*. PHI Learning Pvt. Ltd., 2003.

[14] David G Alciatore. *Introduction to mechatronics and measurement systems*. Tata McGraw-Hill Education, 2007.

[15] Jacob Fraden. *Handbook of modern sensors: physics, designs, and applications*. Springer Science & Business Media, 2004.

[16] Jack Chou. *Hazardous gas monitors: a practical guide to selection, operation and applications*. McGraw-Hill Professional Publishing, 2000.

[17] Iwan Ulrich and Johann Borenstein. Vfh/sup*: Local obstacle avoidance with look-ahead verification. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 3, pages 2505–2511. IEEE, 2000.

[18] S Malcolm and MD Thaler. The only ekg book you'll ever need, 2007.

[19] AC Peixoto and AF Silva. Smart devices: Micro-and nanosensors. In *Bioinspired Materials for Medical Applications*, pages 297–329. Elsevier, 2017.

[20] Ioan Susnea, Viorel Minzu, and Grigore Vasiliu. Simple, real-time obstacle avoidance algorithm for mobile robots. In *8th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS'09)*, 2009.

[21] Vladimir J Lumelsky and Tim Skewis. Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(5):1058–1069, 1990.

[22] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, 19(5): 1179–1187, 1989.

[23] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.

[24] Sander Soo. Object detection using haar-cascade classifier. *Institute of Computer Science, University of Tartu*, pages 1–12, 2014.

[25] S Nagabhushana. *Computer vision and image processing.* New Age International, 2005.

[26] Nathan Lovell and Vladimir Estivill-Castro. Color classification and object recognition for robot soccer under variable illumination. In *Robotic Soccer.* IntechOpen, 2007.

[27] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1:511–518, 2001.

[28] Yufei Wang. Monocular vehicle detection and tracking. *University of California San Diego*, 2009.

[29] Thierry Bouwmans, Fida El Baf, and Bertrand Vachon. Background modeling using mixture of gaussians for foreground detection-a survey. *Recent Patents on Computer Science*, 1(3):219–237, 2008.

[30] F Serhan Daniş, Tekin Meriçli, Cetin Meriçli, and H Levent Akın. Robot detection with a cascade of boosted classifiers based on haar-like features. In *Robot Soccer World Cup*, pages 409–417. Springer, 2010.

[31] Jacci Howard Bear. The hsv color model in graphic design, April 2019. URL https://www.lifewire.com/what-is-hsv-in-design-1078068.

[32] Jozsef Kovecses, J-C Piedboeuf, and Christian Lange. Dynamics modeling and simulation of constrained robotic systems. *IEEE/ASME Transactions on mechatronics*, 8(2):165–177, 2003.

[33] Damn J Price, Simon P Walsh, and Saeid Nahavandi. Unifying manufacturing simulation models using hla. In *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04. 2004*, pages 190–195. IEEE, 2004.

[34] Claus C Aranha, Schubert R Carvalho, and LUIZ MG GONCalvez. Cambio: realistic three dimensional simulation of humanoids based on computer vision and robotics. In *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*, pages 388–395. IEEE, 2002.

[35] Ye Zhang, Wentong Cai, and Stephen J Turner. A parallel object-oriented manufacturing simulation language. In *Proceedings of the fifteenth workshop on Parallel and distributed simulation*, pages 101–108. IEEE Computer Society, 2001.

[36] Serena Ivaldi, Vincent Padois, and Francesco Nori. Tools for dynamics simulation of robots: a survey based on user feedback. *arXiv preprint arXiv:1402.7050*, 2014.

[37] I2c bus, interface and protocol, April 2019. URL https://i2c.info/.

[38] Jae Young Shin, Yun Seok Kwon, Hyun Ho Kim, Kyung Duk Kim, and Min Jung Shim. Method and device for media streaming between server and client using rtp/rtsp standard protocol, June 28 2018. US Patent App. 15/613,557.

[39] Julian T Dolby, Jim A Laredo, John E Wittern, Annie TT Ying, and Yunhui Zheng. Analysis to check web api code usage and specification, May 3 2018. US Patent App. 15/339,792.

[40] Ruben Verborgh and Michel Dumontier. A web api ecosystem through feature-based reuse. *IEEE Internet Computing*, 22(3):29–37, 2018.

[41] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409, 2015.