UNIVERSITAT ROVIRA I VIRGILI

Master Program:
MASTER IN ARTIFICIAL INTELLIGENCE

Master's Thesis

Artificial Vision in the Nao Humanoid Robot

Tomás González Sánchez

Supervisor: Dr. Domènec Puig

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS
September 2009

Artificial Vision in the Nao Humanoid Robot

Tomás González Sánchez

Rovira i Virgili University
Department of Computer Science and Mathematics
Intelligent Robotics and Computer Vision Group

Master's thesis

Abstract
Robocup is an international robotic soccer competition held yearly to promote innovative research and application in robotic intelligence. Nao humanoid robot is the new RoboCup Standard Platform robot. This platform is the new Nao robot designed and manufactured by the french company Aldebaran Robotics. The new robot is an advanced platform for developing new computer vision and robotics methods. This Master Thesis is oriented to the study of some fundamental issues for the artificial vision in the Nao humanoid robots. In particular, color representation models, real-time segmentation techniques, object detection and visual sonar approaches are the computer vision techniques applied to Nao robot in this Master Thesis. Also, Nao's camera model, mathematical robot kinematic and stereo-vision techniques are studied and developed. This thesis also studies the integration between kinematic model and robot perception model to perform RoboCup soccer games and RoboCup technical challenges. This work is focused in the RoboCup environment but all computer vision and robotics algorithms can be easily extended to another robotics fields.

Contents

Chapter 1

Introduction

Robotics is a research field that has experimented several changes in the last years. Nowadays, robot applications are commonly used in dangerous tasks, such as bombs deactivation, spatial exploration or when repetitive actions are done, such as in industrial mass-production.
Research in mobile robotics is focused in robots that have motion ability in a determined environment, sometimes that environment is shared with human presence or human actions. Some problems are present when robots must deal with an environment interaction including: object detection, navigation, interaction with other robots and humans, robot location , etc. Robots can be defined as artificial agents. Concretely, a robot is a mechanical agent that can sense and manipulate its environment, can move around in a local or global space and can exhibit intelligent behaviors.
The *International Organization for Standardization* gives the following robot definition: "*robot is an automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications*". The ISO formal definition is accepted by some important robotics committees, as *European Robotics Research Network* (*EURON*).
Nowadays, a robot is a computer system with input signals from sensors to obtain information and a set of actuators that allow the robot to interact with the environment. That computer system is programmed to react with its actuators using the input sensory information. Robots have some kinds of information sources to develop their tasks; some information inputs are adapted to robot environment as what happens with marine robots. Mobile robots can navigate over terrain, water or in the air.
The remainder of this Master Thesis is organized as follows. Chapter 2 highlights the objectives and motivations of this work. Chapter 3 presents an overview of the main topics dealt with in this work. Chapter 4 shows how the developed methods and algorithms are distributed over two new software architectures. In chapter 5 a Nao robot is kinematically modeled. In chapter 6 Nao's perception system is studied where the work is focused on Nao's Camera. Chapter

7 presents some segmentation techniques applied to the images obtained with Nao's Camera in the RoboCup environment. In chapter 8 the objects in the Nao's environment are studied and some object detection algorithms are presented. Chapter 9 describes some depth estimation methods. Chapter 10 shows how works the visual sonar method over a biped robot. Chapter 11 talks about robot behavior and the challenge developed for the Graz RoboCup competition 2009 and, finally, in chapter 12 some conclusions are summarized and future work is overviewed.

Chapter 2

Objectives and Motivation

The Intelligent Robotics and Computer Vision (RIVI) group of the Rovira i Virgili University has acquired one of the six *Nao* robots that currently are spread out at four universities in Spain. The Nao humanoid robots have become the new standard platform in the Two-Legged League that takes part in the RoboCup competition [1]. The RIVI group is part of the only one team that currently represents Spain at the RoboCup competition.
RoboCup chose to use soccer as a central topic of research, aiming at innovations to be applied for socially significant problems and industries. The ultimate goal of the RoboCup project is by 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer.
However, the majority of today's robots are programmed to follow a predefined trajectory. This is sufficient when the robot is working in a fixed environment where all objects of interest are situated in a predetermined position relative to the robot base. Another option is teleoperation, where a human operator conducts all the movements during the operation in a master-slave architecture. Therefore, artificial vision is an essential matter for any kind of autonomous robots, especially when robots must imitate human behaviors. In this way, the Nao humanoid robots constitute the new active research platform within the robotics community [2].
Last year, one new challenge started for RIVI group, where the author is developing his research task. The group became a full member of the Spanish RoboCup team. For several years the team participated in RoboCup Four-Legged League with the Sony Aibo robots, meanwhile the German Open was a useful training event where the team has also participated in the last years. The last year, this league came out from RoboCup and it will disappear from German Open in the next years. However, a new standard platform was designed to be the successor of SONY Aibo. This platform is the new Nao robot designed and manufactured by the french company Aldebaran Robotics. The new robot is an advanced platform for developing new computer vision and robotics methods. This Master Thesis is oriented to the study of some fundamental issues for

the artificial vision in the Nao humanoid robots. In particular, both low level [3] and high level computer vision requirements will be performed [4].

The specific goals of this Master Thesis are:

1. Kinematic modeling as a first step in the calibration of the Nao robots.
2. Calibration of the 2D camera mounted on Nao robots: estimation of intrinsic and extrinsic parameters.
3. Revision and implementation of low level computer vision tasks oriented to color segmentation, according to the requirements of the RoboCup rules.
4. Revision and implementation of high level computer vision tasks oriented to object recognition, according to the requirements of the RoboCup rules.
5. Study the possibility of performing stereo vision from the 2D single camera mounted on the Nao robot.
6. Study different code optimization levels in order to increase the efficiency of the implementations previously proposed.
7. Finally, if it is possible, the work developed in this Master Thesis will be published in some related conference or journal.

Chapter 3

Preliminars

3.1 Mobile Robots

Robots usually are divided in two wide robot divisions by its morphology: industrial robots and mobile robots. When people think about a robot, their usually have in mind a science fiction robot or industrial robot. The last one modify its closer environment that usually is structured and quite limited. Industrial robots are not capable to move around their environment, sometimes their can make some displacement eventhough not far ago, and in the most cases the movements are calculated previously in behavior development motion phase.
In contradiction, mobile robots main feature is the ability to navigate around their environment.
There are many environments where they can navigate such as air, terrain or aquatic spaces, also there are mobile robots used in out-space exploration that we can consider different to the last robots by their construction motivation.
The space exploration robots are used because the out-space exploration is quite dangerous and there are not resolved problems that affect human space exploration, as human skeleton degradation when astronauts are in weightlessness conditions.
Human limitations cause that researchers community propose multiple solutions to exploration problem in the last decades. Space exploration robots are similar to terrain mobile robots because their have similar features. Aerial robots are another kind of not manned mobile robots, also knows as *unmanned aerial vehicles UAVs*. The UAVs main activities are military control in wide areas in war conditions or frontier control. Military use is not the only for that kind of robots, also there are investigation motivation use as *Aerosonde* and rescue robots as the american *General Atomics MQ-1 Predator*.
Aquatic mobile robots, also known as *autonomous underwater vehicle AUV*, are underwater unmanned vehicles driven by electric engines. Their main tasks are underwater exploration, also in depths where other manned underwater vehicles cannot, and subaquatic structures inspection as intercontinental submarine

fiber network cable.
Our study case are mobile robots, that kind of robots have a widely application field. Nowadays, mobile robots, as vacuum cleaner, are present in our domestic lifes. Usually indoor mobile robots have a light controlled conditions, flat surface to navigate and the environment is structured, as much as possible. Otherwise, outdoors environments where robot environment conditions are not under control, there are unpredicted situations that may cause malfunctions in robot behavior and damages in the worst case.
Robots can obtain environment information from many types of sensors, the most common sensors are:

Infrared sensor Device is divided in two parts, the infrared (*IR*) emitter and the infrared sensor. Basically, this is a device that measure the infrared light radiating from objects. This sensor has an effective field of view, usually conic, and obstacles situated out of that field of view are not detected.

Ultrasonic sensor Ultrasonic is a technique that uses sound propagation. That kind of emitters use ultrasonic waves to detect obstacles and select the kind of obstacle in function of sonar sensor reading. There are two main ultrasonic devices, the sonar and the radar. Sonar is an acronym that means *sound navigation and ranging*. The sonar is one of the most used obstacles detector in mobile robots.

Contact sensor There are many kinds of contact or touch sensors. The main information, that contact sensor provides, is about the possibility that one part of the robot, as it happens in [5], that use a *tactile sensing array*, is touching some object or obstacle. Contact sensors usually are based on a force sensing resistor or high sensitive push button that transmits a signal if a presence is detected and the force moment applied to the object in some cases.

Odometers A odometer is a device, that indicates the distance traveled by a robot. The odometers are commonly used in wheeled mobile robots. If a robot is doing a motion action for a determined distance D then the odometer is used as a closed-loop control device by the robot *odometry* control that provides real traveled distance D'. The robot *odometry* control is used in *SLAM* algorithms [6] to determine the real distance traveled in each robot movement. Odometers usually are based on encoders with a high sensing resolution.

Gyroscopes As odometers provide the traveled distance, a gyroscope determines the rotation produced by the robot in a movement action. The information that the gyroscope device facilitate is used in *SLAM* algorithm too.

GPS *Global Position System* (*GPS*) is an american global positioning satellite system used for navigation purposes. This system indicates where the device is situated in earth. The situation is based on information provided

by many satellites situated in *Intermediate Circular Orbit* (*CIO*). GPS is used to locate a part of the robot and then with a simple spatial transformation another part, *as robot base frame*, position can be determined concisely. Also GPS can be used to situate an object well determined in robot's *working space* in a global position coordinate.

Cameras The cameras are the most important sensor devices that robots can use. Camera provides much information that above sensors. In many cases in mobile robots the cameras are the only or the most important sensor on-board. This sensor provides a set of environment images, as an input sequence, that robot must interpret to take an behavior decision.
As mentioned above, an image can provide much information and in many cases that information can be redundant or higher to be processed in real-time. Then a *computer vision* method must extract the important information using a *feature extraction* method. Then, features obtained are use by *decision making algorithm* that controls the behavior that robot must follow. A set of cameras can facilitate a 3 dimensional (*3D*) information and it can be used to determine objects position in robot's *configuration space*.

Laser Lasers are other important sensor in robotics. In some environments the cameras use are not useful. This device like cameras provides 3D environment information; 3D information usually is more precise than cameras, then the presence of the two devices (cameras and lasers) are common.

As we mentioned above, robots have some abilities as environment sensing, and the most important in mobile robots is environment navigation ability. To perform that robot navigation ability some kinds of *actuators* are used:

Motor This is an electrical actuator that transforms energy to movement. In robotics this kind of motors are the basic movement actuator, as in wheeled-robots because it facilitates speed to robot.

Stepper Motor A stepper is an electrical motor and as the name suggests rotate in discrete steps. This kind of motors are used when exactly control are wanted, as happen in a industrial robot or in the directional wheel of a mobile robot.

Air muscle Air muscles is an human muscle approximation. This is a quiet new actuator that can be used in humanoid robots. *Shadow robot* is one of them (figure 3.1). This robot has the aim of emulate human skeleton. The muscle works like a pneumatic or hydraulic motor,that were used before. The principal problem of human-like actuators is that muscle-like actuators dynamic response is very slow and the power that it can bring is not high yet.

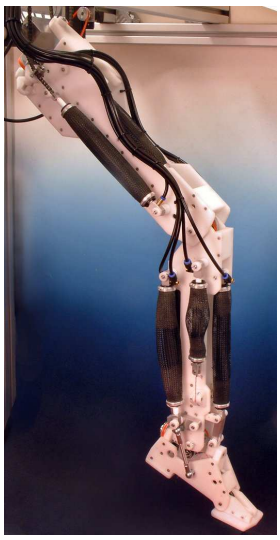

Figure 3.1: Air Muscle

3.2 RoboCup

RoboCup is an international research and education initiative. Its goal is to foster artificial intelligence and robotic research by providing a standard problem where a wide range of technologies can be integrated. Maybe this is the equivalent to the long term milestone of AI community with artificial chess players, and Deep Blue defeated Gary Kasparov in 1997.
The current state of the robotics technology is far from such ambitious goal because this initiative is quite young, the concert of soccer-playing robots was first introduced in 1993. And early, two years after (1995) was made the announcement of the first international conferences and football games. In July 1997 were played in Nagoya (Japan) to present. This year (2009) the international competition was celebrated in Graz (Austria).
The official organization goal is to celebrate in mid-21st century a team of fully autonomous humanoid soccer player shall win the soccer game complying with the official rule of the FIFA, against the winner of the most recent World Cup. The contest has currently five principal competitions:

1. **Simulation League**, The RoboCup Soccer Simulator is a competition focused at multi-agent systems and artificial intelligence. It enables for two teams of 11 simulated autonomous robotic players to play soccer.

2. **Robocup Rescue**, Robocup Rescue is divided in two categories: Simulation League and Robot League. Both categories main purpose are to provide emergency decision support by integration of disaster information, prediction, planning, and human interface. A generic urban disaster

simulation environment is constructed to evaluate the proposed solutions. The intention of the RoboCup Rescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work coordination, physical robotic agents for search and rescue.

3. **Small Size League**, A Small Size robot soccer game takes place between two teams of five robots each. The robots have to follow a strict dimensional and on-board hardware limitations. Robot vision is based on an off-field PC to identify and track the robots as they move around the field. The off-field PC take a strategy decision based on the input information and it sends the motion information to the robots.

4. **Middle Size League**, Two teams of mid-sized robots with all sensors permitted on-board play soccer on a field. The objects situated on field are distinguished by colours. This colours are removed and the standard colours of FIFA rules are going to be fixed as happens actually with goals colour.

5. **Standard Platform League**, In the Standard Platform League all teams use identical robots. Therefore the teams concentrate on software development only, while still using state-of-the-art robots. The robots operate fully autonomously, there is no external control, neither by humans nor by computers. In 2008 the league went through a transition from the four-legged Sony AIBO to the humanoid Aldebaran Nao.

This work is focused in Standard Platform League and with more effort in Two-Legged Standard Platform League. The Two-Legged Standard League use a humanoid robot called Nao [7] developed by Aldebaran Robotics.

3.3 Two-Legged Standard Platform League

Two-Legged Standard Platform League [1] is a soccer game taken by two teams of three robots nowadays. All teams must use Nao humanoid robots manufactured by Aldebaran Robotics. A standard robot architecture was chosen to concentrate the effort to the software development and to guarantee the equal opportunities, absolutely no modifications or additions to the robot hardware are allowed including the additional hardware, the off-board sensing or processing systems.
The game took place in a field of 6 meters length and 4 meters wide, as it is shown at figure 3.2. The field game is divided in two half-fields. These are not identical because one half-field has one yellow goal and the other one blue goal. The robot team equipment can be red or blue, and its equipment change at half time, as well as, the half-field that the team defense. Each half is 10 minutes and clock stops during when robots are not playing. As in common FIFA soccer, the game consists in score more goals that the opponent. One team wins when

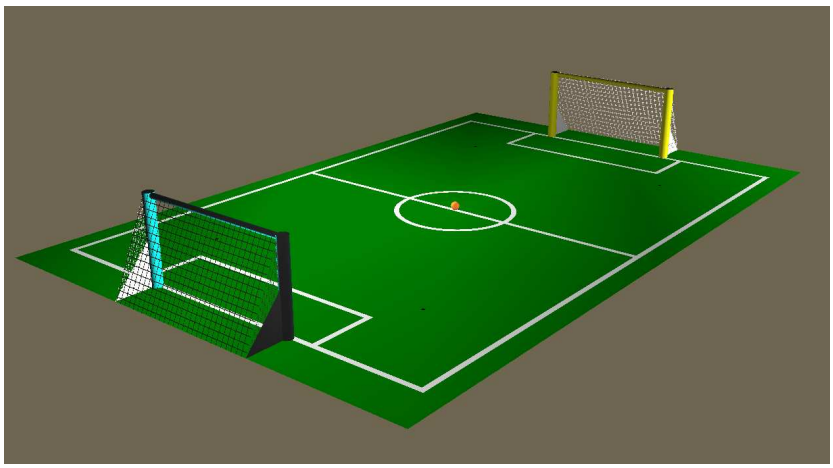

Figure 3.2: Two-Legged Standard Platform League Field

difference between scored goals in opponent goal and goals scored in the own goal is positive. This league rules change gradually to adapt the rules to the FIFA rules.

3.4 Nao Robot Overview

Nao Robot [8] developed by Aldebaran robotics was selected to become the standard platform of *Two-Legged Standard League*. Other standard platform is *Aibo* robot manufactured by Sony. It was the first robot in the Standard Platform League but since Sony decided to close Aibo project, the Four-Legged Standard League is destined to disappear. *Nao* is a biped robot with 23 *DoF* (*Degrees of Freedom*) (Appendix A). *Nao*'s motion is based in DC Motors (direct and stepper types) and the robot has a limited autonomy of 30 minutes approximately.
Nao is equipped with many sensor devices to obtain robot's close environment information:

Ultrasound Nao Robot has 2 ultrasound devices situated in chest that provide space information in 1 meter range distance if a object is situated at 30 degrees from the robot chest (60 degrees all cone combining both devices).

Cameras The last *Nao Robot* version (2.0) was provided with one camera situated at head (forehead). But physical limitation cause that robot can not see the ball when the ball is closer to the robot. Then robot bend over is necessary to detect the ball. The robot bend supposes additional movements to see the ball and its movements are a clearly was of time. Computer vision methods need all time available because information given by the methods are essential for a good robot behavior. Also a close-loop

robot control, called *Feedback*, is used with input images when robot has to kick the ball in a soccer game and another game phases. A few solutions were proposed and a second camera addition was the most voted by Standard Platform Teams, then in Nao version (3.0) a new camera was added.

Bumper A bumper is a contact sensor that help us to know if robot is touching something, in this case the bumpers are situated in front of each foot and they can be used, for example, to know if the robot is kicking the ball or if there are some obstacles touching our feet.

Force Sensors Nao has 8 *Force Sensing Resistors* (*FSR*) situated at sole of feet. There a 4 FSRs in each foot, then. The value returned for each FSR is a time needed by a capacitor to charge depending the FSR resistor value. It is not linear (1/X) and need to be calibrated. When no force is applied the sensor reading is 3000 and when the sensor reading is 200 means that is holding about 3 kg. This sensors are useful when we are generating movements sequences to know if one position is a zero moment pose (*ZMP*) and it sensors can be complements with inertial sensors.

Inertial Sensors *Nao* has a gyrometer and an accelerometer. This sensors are two important devices when we are talking about motion concisely *Kinematics* and *Dynamics*. This sensors help us to know if the robot is in a stable position or in unstable one when robot are walking. Also this two sensor help us with *odometry*, we will talk about odometry in chapter 8.3.

Nao's head is equipped with an AMD Geode 500Mhz CPU motherboard based on x86 intel architecture with 256MB SDRAM and one ARM7 at 60Mhz microcontroller located in the torso that distributes information to all the actuators module microcontrollers. Nao's CPU is the main processor system in the robot and it limited computational capacity is shared by operating system, Nao main controller (*NaoQi*) and user wished tasks or behavior executed on Nao robot.

Once, the Nao hardware and mechanical architecture are revised, a breve introduction to software architecture is necessary to know main *Nao*'s software characteristics. *Nao* software architecture is called *NaoQi* by Aldebaran Robotics. *NaoQi* is designed as a distributed system where each component can be executed locally in robot's on-board system or executed distributed between systems meanwhile *NaoQi Daemon* is running in the main system. The only limitation in the distributed system is that the main process, called *Main Broker*, has to be executed on Nao Robot if we want use actuators or sensors.

NaoQi is divided in three main parts *NaoQi Operative System*, *NaoQi Library* and *Device Control Manager* (*DCM*) figure 3.4. The *Nao operative System* called *OpenNao* is an Open Embedded Linux Distribution [9] modified to be executed on Nao onboard system. Once *OpenNao* is running in Nao's on-board

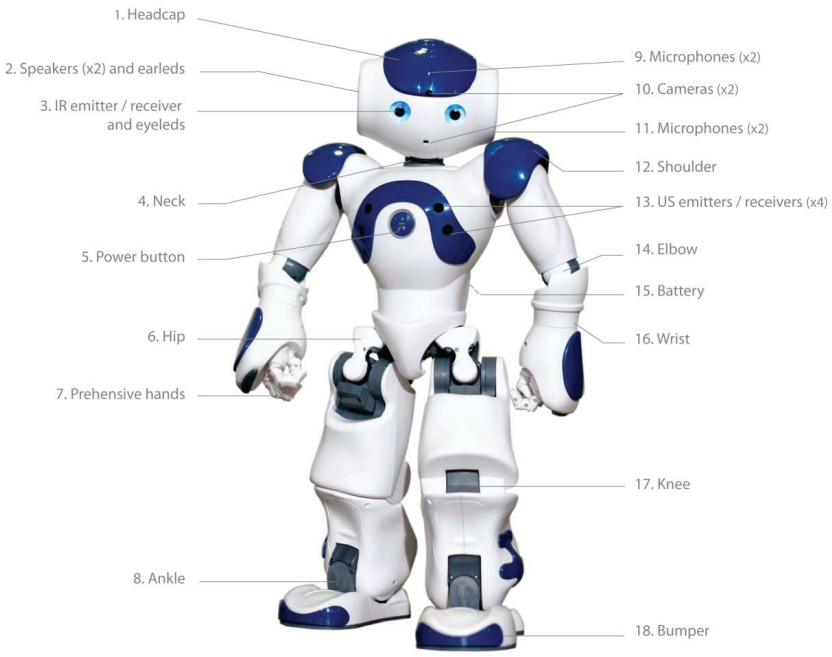

Figure 3.3: Nao's components

system and the operative system initialization process is finished, then *NaoQi Daemon* starts.
NaoQi Library is divided in objects, as we can see in figure 3.5. This objects have implemented some actions that robot provide us. That functionality allow us that our implementation can be designed with an abstraction layer, that avoid us to access directly to the hardware. Otherwise, that abstraction only is useful in the first steps of our development because if we want a high performance then we have to adapt the implementation to Nao's hardware architecture.

The last division is *DCM*, *DCM* like *NaoQi Library* is a set of libraries that allow us to control the robot. The difference between *DCM* and *NaoQi Library* is that *DCM* control the robot directly sending function calls to Nao's *ARM* controller. As ell as, *ARM* is *Nao*'s hardware architecture part that controls and senses all motors and input devices except cameras, because they are connected to head's on-board system. *DCM* is essential if we want real-time access to images, when a new image is captured by the camera, or if we want generate a new sequence for walking, or reach a position. But *DCM* has some disadvantages, the first one is that robot stability are disabled and that stability has to be guaranteed by the developer using some stability techniques of biped robots.

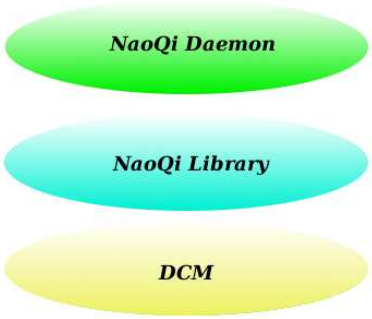

Figure 3.4: Nao Software schema

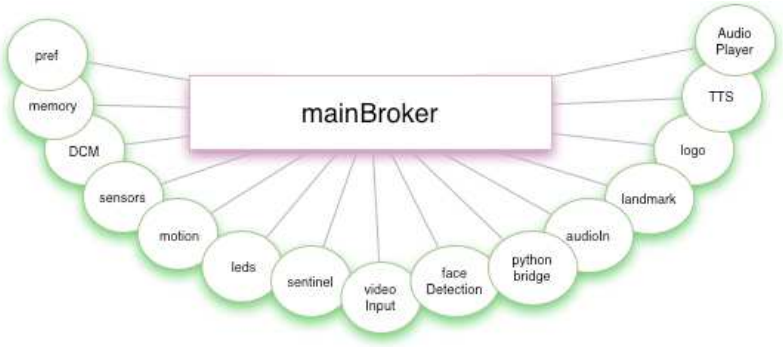

Figure 3.5: NaoQi Library Objects Schema

NaoQi Library satisfy the minimum requisites to perform a quickly development with *Nao Robot* but it has several limitations and disadvantages. To resolve this problems and provide a high level functionality and behaviors to the robot two frameworks was designed and developed: *NaoVi* and *NaoMo*. A full description in chapter 4 of those new frameworks.

3.5 Color Representation

Color is a property of enormous importance to human visual perception. Computational performance problems limited image processing to gray-level images for a several years because data reduction helps researchers to get algorithms with acceptable response times. Nowadays, computational limitations are not as restricted as in the past, but an embedded system has not got computational power than other systems.
Humans detect colours as combinations of the primary colors red, green and blue but its components not imply that all colours can be synthesized as combinations of these 3. RGB colour space is the most used colour model system

to represent gamma colour in image processing methods because it is the most intuitive representation model. Devices as digital cameras, scanners and some kinds of displays use it as embedded colour model system.
However, the RGB space is not the best colour model representation then researchers with aim to have the best colour model have been proposed historically a several models [10]:

1. HSI, HSV or HSL
2. YUV
3. Cielab

The cited colour models are some of them to cite few. All of them are valid colour model systems with some advantages respect the others and few limitations in some cases. This work are focused on RGB, HSI and YUV models because this are the colour models that Nao robot can native handle. Its clearly that computer vision algorithm can handle another colour models representation because an colour model conversion can be done once image camera is acquired. But our solutions must be focused on resolve vision problems spending the few time as possible to use the rest of the time in other artificial intelligence tasks.

3.5.1 RGB Colour Model

The most common colour model is the RGB space where colours are represented by its red, green and blue components in an orthogonal Cartesian space (figure 3.6). There are three kinds of photoreceptors in the retina called *cones* and this is in agreement with the *tristimulus theory of color* according to the human visual system. The main RGB colour representation model problems are:

1. It is not intuitive colour model, because it is very difficult to know what colour are representing with a determined red, green and blue value.
2. It is not uniform because with given 2 RGB colours and applying the same variation to the 2 colours a different chromatic variation happens.

Computer vision researchers have a valuable example to follow in visual perception methods. This example is the human visual perception system. Human vision perhaps is not the most complete vision system in the nature, its commonly known that some animals have better vision system than human in a wide range of characteristics, but human vision is the most studied and modeled for a while. With aim to have a colour representation model as same as possible to the human visual perception model, the HSI model was created. In the next section the HSI model will be revised.

3.5.2 HSI Colour Model

HSI model is a three component colour representation as RGB. Hue (H) refers to the perceived colour, the saturation (S) component measures its dilution by

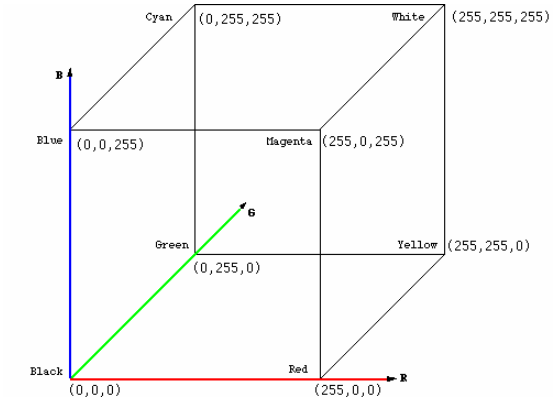

Figure 3.6: RGB orthogonal space shown at its common cube RGB representation.

white light given light red for example. The third component is the intensity (I) of the color. As mentioned before, HSI model is a useful representation model but it has an important disadvantage. The principal acquire device usually not use HSI model in the acquiring sensors. This means that a model conversion method must be computed to obtain an image in HSI colour model. HSI model is an independent orthogonal model system in all cases excepts when intensity value is zero where the colour black is defined. Black colour is independent of hue and saturation values.
HSI model is more intuitive color model system because works a as human visual perception system. The basic colour is defined in the hue component, it is defined as an angle, and it works from 0 degrees to 360 degrees. Then the three complete colour palette (red or green or blue) can be defined with 120 degrees each one as Newton formulated, in *Newton's circle*. The other colour represented by Hue is lineal distributed in the 360 degrees and all colour can be determined with an angle.
Saturation component usually is defined as a normalized value from 0 to 1. Where a pure colour is determined when saturation is fixed on 0, and then saturation is higher when saturation value is fixed on 1.
The last model component is the intensity value. This component is a reference of the light amount that falls over colour. Usually this component is defined from 0 to 1 in the HSI colour model. HSI is considered a good colour model representation, but some authors defined RGB and HSI models as not uniform color representation model. To solve the problem some researchers created Cielab. The colour model that solved that great colour space modeling problem.

3.5.3 YUV Colour Model

YUV colour model divides colour in two main elements: *luminance* and *chrominance*. Y component in the model space defines the luma component and

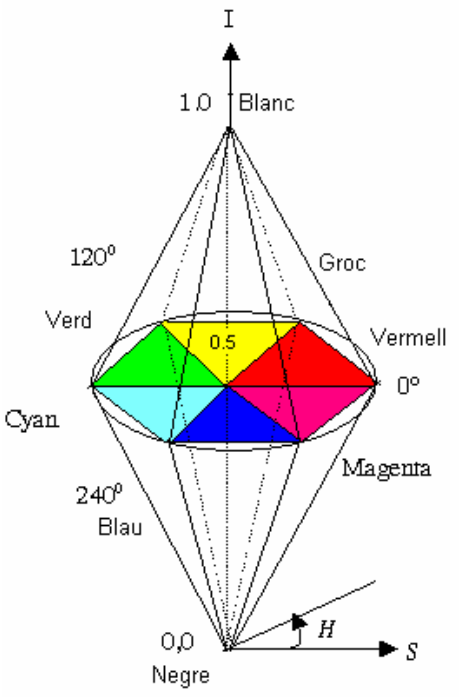

Figure 3.7: HSI colour model space.

chrominance is defined by two components U and V. YUV colour model takes into account human perception model and uses luminance as brightness perception over a colour. In contradiction to HSI, YUV model is the native device acquiring sensor colour model in many input devices, and it is the color representation model at Nao's camera, too. This means that input acquired image have not got to be converted to work with this colour model. It issue eliminate conversion problems as happens with HSI model and in some cases RGB devices where RGB is not the native camera's colour model.
Previous black-and-white systems used only luma (Y) information and color information (U and V) was added so that a black-and-white receiver would still be able to display a color picture as a normal black and white pictures.
YUV models human perception of color in a different way than the standard RGB colour model used in computer graphics hardware. The human eye has fairly little color sensitivity: the accuracy of the brightness information of the luminance channel has far more impact on the image discerned than that of the other two.
YUV has not developed to be used in a computer vision algorithms, YUV was developed to be a colour model to be used in telecomunication ambit and it is used specially in Europe. It is similar to YIQ in American telecomunications.

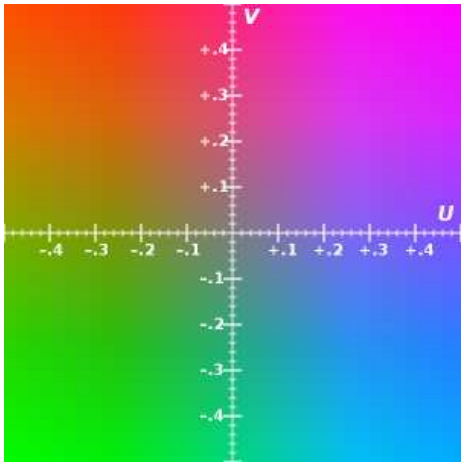

Figure 3.8: YUV representation on a plane when Y channel is fixed at 0.5

YUV allow video compression in its native formulation YUV422 and it is used in some video compression encoders. Nao robot works with the three above colour model (RGB, HSI and YUV). HSI is the best colour model because RoboCup field environment has perfectly known and objects located at field have got different colours, then H component can distinguish perfectly objects. In the other hand time response as we will see in chapter 7 will increase.

Chapter 4

Software Architecture

NaoQi Library is a software architecture that is part of *NaoQi* software provided by Aldebaran Robotics. This software implements the minimum functions to interact with the robot, sensors and actuators, to develop our own behaviors and actions. NaoQi is designed as a framework (figure 4.1) that support function calls in C++ and python since the last version. NaoQi is an *OOP* (*Object Oriented Programming*), which minimum object element is called *Module*. NaoQi is an event-based programming where modules interact one to the others using shared memory (*ALMemory*). Also OOP has inherit method interaction but *NaoQi* was designed by Aldebaran Engineers like event-based programming framework. *ALMemory* is quite important because each image captured by the cameras is allocated in one internal variable (*ALValue*). That variable is accessible using *ALVision* module; this module is a wrapper over ALMemory that allows concurrent access to image data and mutual exclusion when image memory is updated.

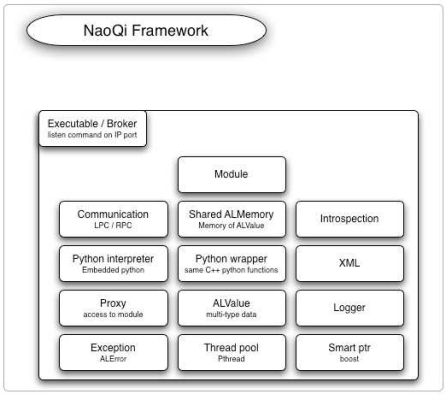

Figure 4.1: NaoQi Framework

Aldebaran's framework has several disadvantages, the main one is related with real-time functions. In motion case, if we desired to do a set of movements with *ALMotion* module, it does not guarantee that joints values reach positions that there are determined in method call. Also, if movements were executed as developer determined, joints motion times are not also guaranteed. This means that, if the developer wishes that robot arrive to a fixed position in a determined time, the robot can arrive to the position several seconds late.
Other important problem is related with *Nao*'s camera and *ALVision* module. This module, as explained before, has critical exclusion implemented over image access. It exclusion treatments is a good idea if the camera is considered as a resource that many process (modules) have to share. But in this case the camera will be accessed only by on module, then mutual exclusion and *ALVision* wrapper is not necessary, also it introduces overhead when image is captured. In real-time and deterministic problems, like computer vision and robot motion, are important to reduce, as possible, the overhead to dedicate all free time to resolve the environment interaction problems. The RoboCup soccer scenario is dynamic and it changes in few seconds, then *NaoQi Library* not satisfies all our needs. To resolve *NaoQi* problems we designed and developed two new frameworks called *NaoMo* and *NaoVi*. In *Nao*'s new software framework, *NaoMo* (*Nao Motion*) interacts with *NaoQi Library* and *DCM* to use the best features of each software interface. NaoMo implements the basic movement functions provided by *NaoQi Library* without the undetermined movement and time that *NaoQi Library* has. Also, *NaoMo* offers to *NaoVi* (*Nao Vision*) some other functional methods that *NaoQi Library* does not. *NaoQi Library* does not offer to developers any kind of computer vision functionality, only camera access and some image processing demos without utility for us, then a computer vision framework utility has to be designed for RoboCup competition and future use. In next section will be revised NaoVi and NaoMo characteristics.

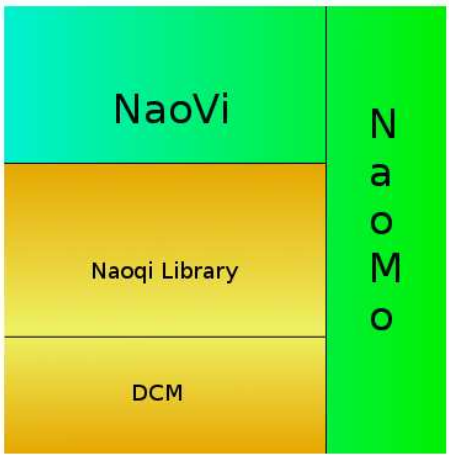

Figure 4.2: NaoMo and NaoVi Frameworks

4.1 NaoVi

A developer can interact with *NaoQi Library* in *Python* and *C++* languages. In older versions of *NaoQi* ruby language is useful too, but with *Nao* Robot version 3.0 it is not possible.
Python is high-level programming language and it mains characteristic relapsed that *Python* is object oriented language. Python is easy to use and developer has not to worry about low-level software or hardware architecture. Python is the perfect start-up language to quickly develop easy robot actions. In contradiction, Python is a scripting-language, then python has a minimum latency when it reads instructions from python script file and it executes them. Developers when use *Python* in situations when time is important they tend to use *Cython*. *Cython* is a syntax based on *Pyrex* language and allow to developers to call directly *C* methods.
The other language that *NaoQi Library* support is the well know *C++* language. *C++* is regarded as middle-level language. It is an object based language born as a extension of *C*, initially known as *C with classes*. *C++* effectiveness for critical and real-time applications has been tested and proved in several fields; some fields are industry applications, device drivers, embedded software and in robotics. In robotics field *C++* use is quite extended because *C++* has low-level languages facilities, as well as, high-level facilities.
Once *Python* and *C++ NaoQi's Library* support were compared, we considered that the best language in this case is *C++* because it is the best language, nowadays, in real-time applications. Also, *Nao* developers reference explains that *ALMethods* in *Python* are not implemented at all, yet. This means that access to images captured from the camera in that language cannot be done. With this great disadvantage, the better languages to develop *NaoVi* is *C++*. In Addition, an optimization can be done at code execution if a *cross-compilation* is made for AMD Geode processor to increase the computation efficiency using *C++*. It free time can be used in critical actions.
NaoVi is divided in modules as *NaoQi Library*. Each module implements methods or algorithms to be used by high level computer vision algorithms or robotics artificial behavior algorithms. These modules are created with aim to provide low or middle level computer vision algorithms, and high-level methods in the future. Also, NaoVi is an abstraction Layer to be used in future work and it can be considered then as a image processing framework for the *Nao* robot. *NaoVi* facilitate the follow support:

Color Segmentation Few segmentation techniques are developed to response real-time requirements. A review is done in Color Segmentation chapter.

Features Detection In computer vision scope, we can consider a *feature* as image pixel that have different property or colour value, when around pixels are compared with it. Harris corner detector and SIFT methods are developed, as a first step to develop a future object tracking method. Once inverse perspective projection at robot's camera is known, it can be

used as a tracking method as happens in [11]. Also, some classical methods can be used combined with statistical and particle filtering techniques[12].

Object Detection Some objects are detected as goals, ball and other Naos at field. Also field lines are detected using a white channel segmentation combined with a border extractor method. Then, the classical Hough transformation is used to detect field lines. Also, lines intersections are computed to be another beacon element to be used as a visual mark to a visual odometry method.

Visual Odometry Biped robots in contradiction with wheeled robots can not implement easily a classic robot odometry techniques based of sensing actuators encoders. In RoboCup field environment, only there are two elements that can be taken on account as a beacon, goals and lines. In many cases, the only element that is present at images is the field lines, because goals are not perceived for a long time. To avoid that situations, visual odometry can be used combined with field lines information. Visual odometry uses known field geometry to situate robot in field space and compare the last estimated position with the new one. Visual odometry, also, can be used as an generalized solution as in [13] where the perception system is an monocular camera as in Nao's case. Campbell article shows how camera coordinates can be easily mapped into ground plane when intrinsic and extrinsic camera parameters are known. Visual odometry fails when robot is moving meanwhile image is acquiring, in this situations some articles proposed some solutions to the problem. The solution is detected invariant feature to the motion blur [14]. Proposed solutions can not be applied into a real-time problem because they are computational expensive. NaoVi Visual Odometry will use field crosslines form two consecutive images to detect the motion of the robot. Error estimation was done but motion correction are not used by NaoMo yet.

Perspective Projection Complete camera models will be revised in chapter 6.

Distance Estimation Depth Estimation is commonly implemented in stereovision systems. With two camera, that are situated in a well know position, a distance to an object or point in the image can be estimated applying a trigonometric function. Recently, robots with only one camera can estimate distance. The estimation is based on using a camera with zoom [15], then changing the focal length we can acquire a set of images that help us to estimate the position. Also, robots with a omnidirectional single camera [16] can estimate distance using a concave lens and a convex mirror but it is not our study case.

Visual Sonar A new approach of Visual Sonar is developed to work in a biped robot. It uses kinematic analysis even when robot is supported by only one leg. The visual Sonar method was introduced by Scott Lenser at [17].

4.2 NaoMo

Nao Motion is a framework developed to supply motion needs in RoboCup contest. This framework is mainly a set of head, arms and legs motion methods. As said before, *NaoQi Library* and concisely *ALMotion* module provides a set of basic motion movements. These methods provided by Aldebaran are useful for a quickly robot start-up. It Was clearly accepted that another implementation of motion library will be necessary for fast and correct behavior developments in the future. Also, a full study of functions that Aldebaran provides to manage the robot was done to seek where *NaoQi* does not work well, and where *NaoQi* does. This is the conclusion that we considered:

Robot Stability *Nao* is 58 cm high [8] and its weight is more or less 4,3 kg. The battery is positioned at *Nao*'s back, this position is situated upper the middle of the robot, when it is completely stand up. The other heavy component of the robot is the on-board computer and its peripherals, these parts are situated in robot's head. Obviously, the fact that robot heavy components are situated upper the middle robot cause that robot stability is not as good as can be. Also, the feet weight seems to be light when they are compared with the rest part of the robot. To resolve this problem high level *NaoQi Library* provides a stability control based on *ZMP* (*Zero Moment Pose*)[18],[19] but it must be disabled to use *DCM* controls. Johannes work is faced on implements an omnidirectional ZMP-based engine with a simple inverted pendulum model. His model introduces a egocentric coordinate frame to define the step placement. Also, these coordinates frames allow a direct translation to a CoM trajectory. CoM control is the easier robot's stability control and the most reactive, then this approach is the most dynamic. A dynamic control allows to cancel or adapt the robot trajectory or the robot motion.

Robot Walk When *Nao* walks, the developer has to set distance to travel, the distance of robot step and time per step. Then, *NaoQi Library* uses calculated before Kinematic and Jacobian Matrix to generate desired movement.

When the joint values are calculated the on-board system stores them into internal memory (*ALMemory*). *ARM* controller loads these values from *ALMemory* and sends the values to the joints in sequences. Sequence time step is fixed at method.

The main problem of *NaoQi Library* movement generator is that joint values are not tested while robot is doing motion. This means that robot can fall and damages can be produced if a joint not reach the value sent by the controller. In robotics, is very common when an actuator is working for a long time that each movement has an error, that is produced respect to the desired movement. And it error is accumulated with the next movement. This error is critical in *SLAM* (Simultaneous localization and mapping) methods where *odometry* is an important source of information. But *odometry* is not completely reliable because joints have an accumu-

lated movement error. In that situations a correction method, like *visual odometry*, is common. At *Nao* robot there is not a feedback control from joints and error is accumulated without any type of motion correction.

NaoMo like *NaoVi* is implemented in *C++* to facilitate the interaction task. *NaoMo* supply the follow actions:

Walk Straight *NaoQi Library* basic movements are based on distance. With a monocular vision and a camera without a zoom, distance is always estimated with some little error, but this is present. Distance estimation error is bigger when object that is going to be estimated is small or is to far. To overtake this problem, we designed a set of methods to base robot walk in velocity, in-front of *NaoQi Library* based on distance and step per distance. *DCM* motion is based on step per distance, too. Then, a basic distance and step per distance were defined to perform correctly the function. The method was designed to make motion in the direction of a detected object when it is far. Then, when distance is small is used a second motion walk method based on distance .

Turn Around As the before motion movement, *NaoQi Library* defined turn as a distance and step problem. To normalize the motion methods in *Nao*, we defined *Turn Around* movement as velocity problem where basic distance rotation movement are defined, and velocity value modify elapsed time per step.

Diagonal Walk *NaoQi Library* does not implement Diagonal Walk. Then, if Diagonal Walk is necessary then robot has to combine Walk Straight motion with Turn Around motion in *NaoQi Library*. *NaoMo*'s Diagonal Walk is based on [20] where one side leg actuators values are scaled-back on walk. Scaled-back actuators values locomotion are suitable because we generate a stable basic locomotion previously. Then to perform the wished locomotion, a simple algebraic function has to be applied to the basic motion. The easy and the faster way to modify the base locomotion is to scale joints angle but there are other solutions, such as joints values approximation using a continues sinusoidal function or generate legs independently movement, as in sequence generator [19] where legs movements are modeled with an *ANN* (*Artificial Neural Network*).The main disadvantage, in the two first solutions, is that locomotion algorithm is slow because dynamic movement generation is necessary for each leg. It is computationally expensive in biped robots because they have much DoF. Solution time response problem is usually solved using an *Evolutive Algorithm* as *ANN* or *Genetic Algorithm*.

Arc Walk *Arc Walk* generation is similar than Diagonal Walk. Locomotion generation is based on scaled-back leg actuators values. In this situation, joint values are modified to reach a curve trajectory (figure 4.3). Initially, scaled-back value is calculated considering locomotion distance and arc

angle. As motion methods commented above, this method resolution is based on distance problem to generalize *NaoMo* developer interface.

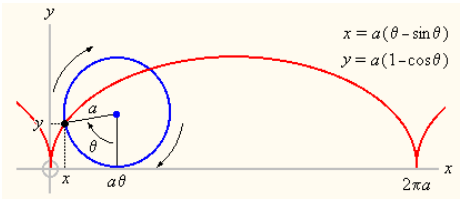

Figure 4.3: Arc Walk Schema

Environment Scan *NaoQi Library* provides us with motion actions like we overview above. *Nao* biped robot can be used in many cases as well as *RoboCup* environment. In all possible uses are useful to have some methods to scan the environment with the aim of search some object or feature. *Environment Scan* methods allow to situate an object in robot 3D space, search some features.

Environment Sensing *Nao* robot has two main sensors besides cameras. These sensors are sonar and bumpers. Sonar sensor is suitable to detect possible obstacles to avoid possible damages. Sonar sensor values can be used combined with camera perception when camera cone view angle orientation is different than body angle orientation. The other important secondary sensor, if we consider the camera the main one, is the feet bumpers. We use feet bumper to test environment for possible collision with another fell robot, also for ball detection. To prevent possible robot damages when one of the secondary sensors indicate an object presence, the head actuators are modified to situate the effective vision angle over bumper or sonar effective angle. If the sonar or bumper sensor was wrong, the last behavior continues its work. Trajectory algorithm and obstacle avoiding method update the perception values if input information is confirmed with camera image information.

Robot Fallen Sensing When *Nao* robot is playing soccer in RoboCup competition often the robot falls. If robot falls, *RoboCup* rules [1] allow to do two kinds of actions: *Request for Pick-up* and robot auto stands up. Fall situation is important to be sensed to avoid possible damages at actuators, because if robot has not the knowledge that it is laying on the floor may continue the current locomotion actions.

Robot Stands up When fall situation is detected, the robot changes its behavior automate to a priority state called *Stand-up*. In this state, the robot stops all current movements and *stand-up* state sends to actuators a value sequence to reach the lift up position again.

Motion algorithms are implemented using *Simbicon* [21] method approximation. Nao locomotion is based on one main hierarchical automate (figure 4.4). NaoMo is a three layer hierarchical automate, that allows mutual exclusion between motion methods (walk straight, arc walk, ...), avoiding unstable situations if automate behavior state changes without stability control. Top automate layer is the high-behavior layer (*Nao Player*). Automate state changes when perception environment data changes, like ball position. When automate state changes, the robot behavior changes too. The bottom automate layer (Move Controller) is an action or motion layer. Each state in this layer performs an action or actions that robot must follows to reach a behavior, like walk straight.

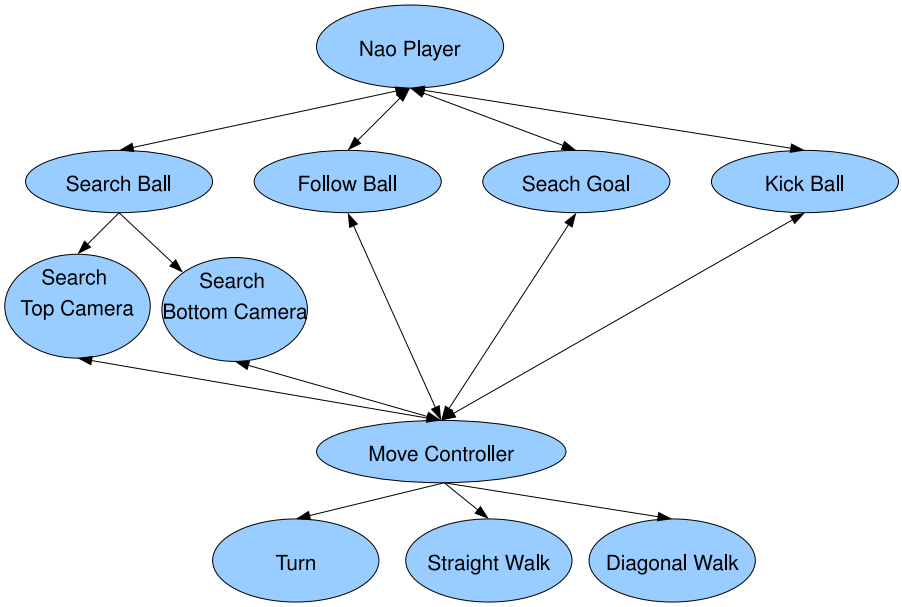

Figure 4.4: Behavior automate schema

Motion methods comparative

Two characteristics was used to compare Aldebaran motion method with the motion developed by us, these are stability and velocity. The stability was chosen as a qualitative characteristic because robot safe is priority for us. Velocity is the second one characteristic, because the velocity is fundamental to perform a good soccer game. NaoMo motion is always stable and NaoQi motion is unstable in Turn action. Also, NaoMo implements Walk Diagonal and almost all motion actions are faster than NaoQi methods. In contradiction, Turn action in NaoQi is faster than NaoMo method. The maxim step length (figure 4.5)

at NaoQi is defined as the half step per second of the defined walk velocity, although the step at DCM is refreshed every 20 ms. In NaoMo architecture the step per cycle is defined as maxim step length/3 to adapt the step in a homogenius time base.

Table 4.1: Image acquiring times in Nao robot.

Image	Walk Straight	Turn	Walk Diagonal	Arc Walk
NaoQi	8cm/s	0.5 rads/s	x	8cm/s
NaoMo	10cm/s	0.3rad/s	10cm/s	10cm/s

Nao locomotion is based on kinematic robot modeling and dynamic robot modeling is not used. Nowadays, we are interested on motion planning based on reach positions or joint values. Joint positions can be fully described using kinematic models, like D-H explained in the follow chapter, and dynamics commonly is used to describe joint forces acceleration and mechanical actuator direct control [22]. In future work dynamic Nao model could be abroad if it will be necessary to generate better locomotion behaviors. In chapter 5 robot modeling and kinematics will be explained accurately.

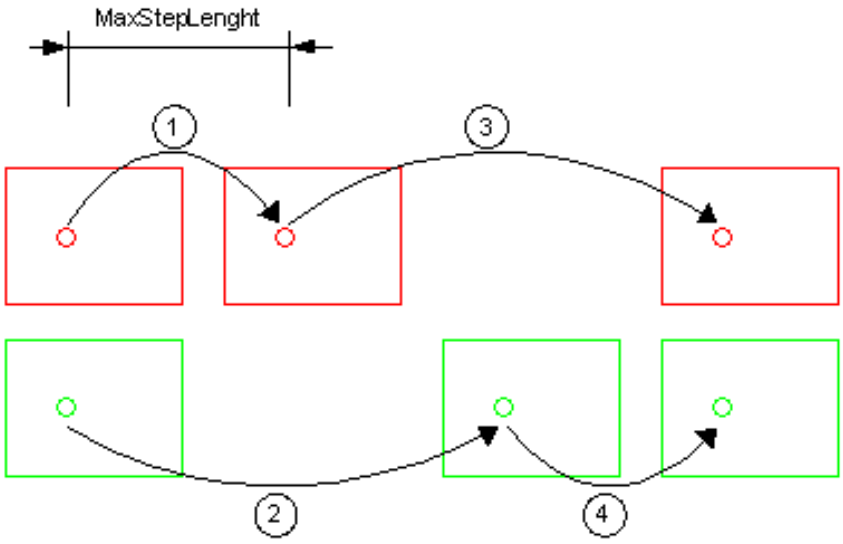

Figure 4.5: Walk Straight Step Schema

Chapter 5

Kinematics Analysis

Kinematics is the study of how things move. Kinematics is a branch of classical mechanics which describes the motion of objects without consideration of the causes leading to the motion. Kinematics is the basis of robot control. Many different solutions have been proposed for solving the problem of robot mobility in the last years. However, the design of biped robots able to walk like humans do is still a difficult task to achieve, mainly because of stability problems derived from the use of only two support legs. Before facing up the stability problems, the robot morphology must be described.
In *robot modeling* [23], robot parts are divided in two main components: *joints* and *links*. A link represents each interconnection between two joints and a joint is an actuator representation. Usually, in mobile robots those actuators are revolute joints, but joints can be prismatic too. The main difference between prismatic and revolute joints is that the first one is a linear relative motion between two links. However, a revolute joint is a simple axis rotation.
Revolute joints and prismatic have always one degree of freedom but some joints can have more than one degree. Then, a joint with more than one DoF can be approximated as a set of one degree of freedom joint to simplify the problem without loosing information. Then, a robot with n joints will have $n+1$ links, since each joint connects two links. Then joint i connects with link $i-1$ to link i, if we number the joints from 1 to n and links from 0 to n.
Robot kinematic can be modeled with many analytical methods:

Denavit-Hartenberg *Denavit-Hartenberg* convention (*D-H*) uses coordinate transformation matrix methods to describe mathematically robot structure. DH method is widely used in the robotics literature because can be applied sequentially to obtain robot geometry in few easy steps (see more at Appendix B). The DH model is obtained by describing each link frame along the robotic chain with respect to he preceding link frame.

Rotary Transform Tensor Method

Plural Polar Vector Plural Polar Vector Methods are often used in locomo-

tion analysis when the robot space and robot motion can be well defined in a 2D dimension plane. In 2D case the robot motion is determined by two scalar values ρ and θ. The typical example is a wheeled-robot kinematic analysis. In this kinematic model, the translated distance between one point P_i to point P_j is represented by ρ meanwhile the angle rotation is represented by θ as figure 5.1 show. Plural Polar Vector method needs for each joint a three component vector to represent the module for each axis and a three component vector to represent each argument, or a unified six component vector. Then 6 vectors are needed to represent a 6 DOF robot. It is important to understand, 6 vectors are component dependent and it means that the information that they represent can be stored or defined by vectors with less dimension. Taken on one side that robot kinematic can be represented with less data dimensions with other models and that Nao robot has a high level of Dof (23 DoFs).

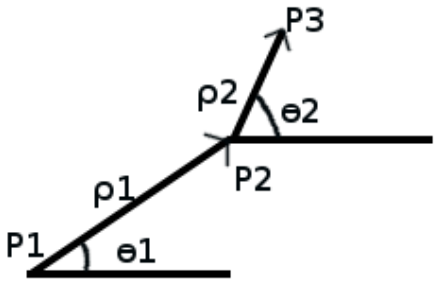

Figure 5.1: Polar Axis

Screw Theory Screw theory is a powerful mathematical tool to solve the kinematics questions of a robot. Screw Theory was developed by Robert Stawell in 1876, as the way to express displacements, velocities, forces and torques in 3D space inclusive the rotational component. Some authors defends the use of screw theory based on the evidence [24] of the number of frames used in D-H method. Also, they argument that screw theory is more convenient to solve dynamic robot analysis.

Objects in 3D space have 6 *DoF* (*Degrees of Freedom*). Three of that degrees are for position at space and the other three for orientation. Also, if space is referenced to the common three orthogonal axes *frame*, then position of an concrete object is determined by a three component vector (x, y, z). Usually (x,y,z) position is fixed at its center of mass, but another configuration is possible too. Once object is well situated in 3D space any orientation is possible. Orientation versus *world frame* or *space frame* is common represented with another three component vector (α, β γ) where each component is the rotation in x-axis, y-axis and z-axis, respectively.

In mobile robotics is very common to situate an object or a point in 3D space referenced by robot base frame. Once the important object is referenced, a simple transformation projects the object in *space frame* to the world *world frame*, if the robot location is known.

5.1 Forward Kinematics

The *forward kinematics* process consists of computing the position and orientation of a robot *end-effector*. In biped robots the orientation of called *end-effector* means the orientation of each chain (legs, arms and head). The knowledge of all joint variables is necessary to calculate the robot *forward kinematics*, the easy way to obtain the end-effector pose is based on D-H parameters. These parameters describe each frame F_0, F_1 F_2, ..., F_n in a n-joint robot. Each link frame is fully described by its pose matrix with respect to the preceding link frame along the robotic chain, and the sequence for pose matrices, which are also homogeneous frame transforms, are used in the forward kinematics process to compute the pose matrix $\mathbf{M}_0^n$ of the end-effector frame F_n with respect to the base frame of the robot F_0.
A frame F_i is described with respect to F_{i-1}, over the z-axis using D-H method, by its pose matrix as determined in Equation 5.1

$$M_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

where α_i, ρ_i, a_i and d_i are the D-H parameters at frame F_i. Also, M_i can be represented as

$$M_i = \begin{bmatrix} R_i & P_i \\ 000 & 1 \end{bmatrix} \tag{5.2}$$

where R_i and P_i are called Rotation matrix and translation vector. R_i matrix indicate the orientation of F_i from F_{i-1} and P_i vector express where the origin F_i is from F_{i-1}.
The objective of the forward kinematics is to compute the pose $\mathbf{P}$ of the end-effector with respect to the base frame. Using D-H link frame assignment convention, the frame attached to end-effector is simply F_n at n-joints robot. Then robot frames transformation can be characterized as:

$$M_0^n = M_0 M_1 M_2 ... M_n \tag{5.3}$$

M_0^n matrix is composed as M_i matrix by R_0^n and P_0^n and it can be transformed to homogeneous matrix too. Then M_0^n is only a transformation matrix to frame F_0 to end-effector frame F_n.

5.2 Inverse Kinematics

Forward Kinematics gives us a useful tool to determine a pose **P** in 3D space, but robot control actions are naturally executed in the joint space while robot motions are specified in the task space. When we analyzing the kinematic of a robot, the natural first step is always to produce the forward kinematics equations of pose **P** of the known joint values $(\theta_1, \theta_2, \theta_3, \cdots, \theta_n)$ in revolute n-joints case. Note that M_i is a function depending on θ_i robot actuator. Composing each M_i as shown in section 5.1 we obtain M_0^n.

$$M_0 M_1 M_2 ... M_n = \boldsymbol{P} \tag{5.4}$$

Also, to reach a end-effector pose **P** the sequence set could not be unique, especially in humanoid robots where is common that robot has DoF redundancy. Then a poses can be reach by a multiple motion planning and multiple actuators values. The complexity of the inverse Kinematic (IK) increase as the number of joints increases from one to six. Since an unconstrained object can move in space with six degrees of freedom, a six-joint robot has the minimum number of joints required for 6 DoF. Robots with more than six joints are said to be kinetically redundant.
Once robot geometry is determined by forward kinematics using D-H or another methods explained above. In the 3D cartesian space a point is determined by a 3 dimension vector $\vec{p} = (p_x, p_y, p_z)$. Each component of vector $\vec{p}$ is obtained by an algebraic composition of actuators values q_i. In revolute actuator case q_i value is the θ_i around angle over z-axis. Then vector $\vec{p}$ can be defined by a matrix P as

$$P = \begin{bmatrix} n & b & t & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.5}$$

Vectors n, b, t and p can be defined as an algebraic decomposition:

$$n = (n_x, n_y, n_z)^T \tag{5.6}$$

$$b = (b_x, b_y, b_z)^T \tag{5.7}$$

$$t = (t_x, t_y, t_z)^T \tag{5.8}$$

$$p = (p_x, p_y, p_z)^T \tag{5.9}$$

then matrix P aspect is:

$$P = \begin{bmatrix} n_x & b_x & t_x & p_x \\ n_y & b_y & t_y & p_y \\ n_z & b_z & t_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.10}$$

Each component in matrix P (5.10) is determined by a function f_{ij} of joint value θ_i.

$$f_{ij}(\theta_1, \theta_2, \theta_3, \cdots, \theta_n) = p_{ij} \tag{5.11}$$

Replacing each matrix P component we will form 12 equations with n independent variables (one per each joint) and p_{ij} dependent variable. Then the inverse kinematic problem is reduced to algebraic or geometric problem that can be solved in few steps.
In real inverse kinematics robot problems, we can reduce equation 5.11 because not all joints are present in the most kinematic motions. For example, the head joints can not be activated or modified when robot are walking. If robot has 2 or 3 joints at head, we reduce equation avoid it joints as shown in [25].

5.3 Nao Kinematics

Nao is an humanoid robot (appendix A) with 5 *DoF* (*Degrees of Freedom*) in each leg, 1 *DoF* at pelvis actuator and 12 DoF at the rest of body robot. Nao kinematic chain is formed only by simple revolute joints and in that way, the robot geometry help us to analyze it *Kinematic geometry*.
D-H method (appendix B) was chosen to perform Kinematic analysis. The decision criteria is:

De factor Standard Denavit-Hatenberg robot modeling technique has become the standard the factor in mathematical representation of robotic structures. And nowadays is the most used representation method used in robotics. It can represent many kinds of robot like industrial robots or mobile robots with mobile parts.

Parameters D-H is an easy way to represent robot structure only defining some features (4 parameters) for each link.

In literature, D-H modeling method was designed and used to describe industrial robots. Also, D-H can be used to describe an humanoid robot [19, 26] like Nao. It is easy to arrive to the idea that a biped robot with two legs, two arms and one head can be modeled as a industrial robot with 5 independent arms, if we are considering chest the *base frame*.
Aldebaran Robotics engineers follow that design when kinematic model was made. As shown in figure 5.2, five frames are present called *[0], [1], [2], [3]* and *[4]* for head, left-hand, left-leg right-leg and right-hand respectively. Using body frame, called *waist* by Aldebaran, a developer can obtain head-foot (or vice versa) frames transformation using 2 *NaoQi Library* methods. NaoMo development aim was reduce the waste time, as much as possible, in motion and kinematics transformation. To reach that objective two kinematics transformation are developed; the first one that make transformations from the floor fixed foot to the head and the second transformation that implements the complementary way, from the head to the fixed foot.

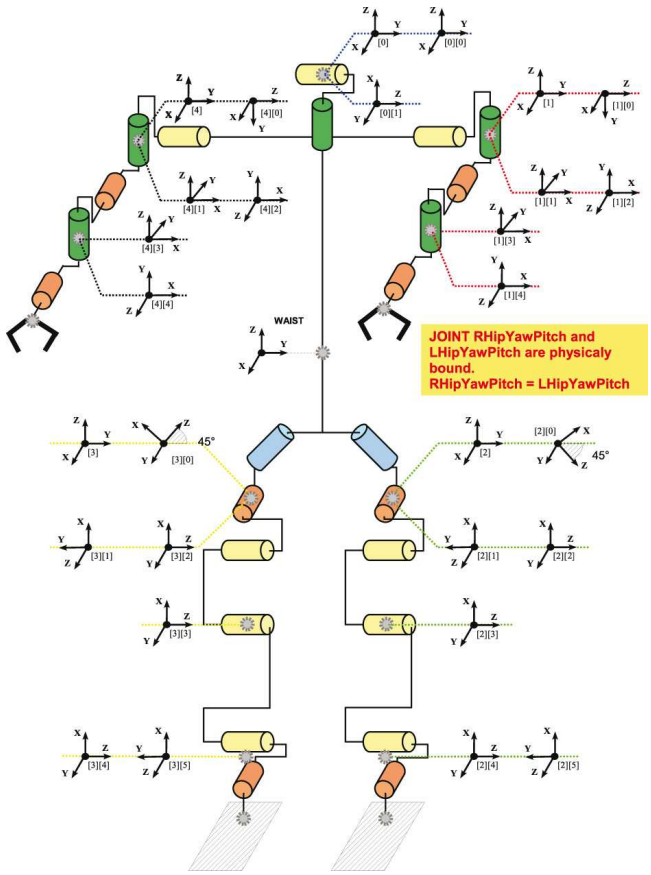

Figure 5.2: Nao Kinematic Schema

Solving *Forward* and *Inverse kinematics*, as show in previous sections, two main problems are face up. The first one help us to know the 3D head orientation and the second one tell us which is the set of joint values to reach a determined position.
Head 3D orientation is an useful information in locomotion, location, object avoiding and distance estimation because the main sensor in Nao is situated on robot's forehead. To situate a point in 3D space using the camera is important to know the accurate frames transformation. Inquiring images only with one camera and using robot modeling precise information, for distance estimation and location, is possible as we will explain in the following sections.
Also, Inverse Kinematics is the first step to generate all full optimal motion algorithm. Aldebaran Robotics supply developers with a basic motion methods in the two principal framework layers: NaoQi Library and DCM. NaoMo current version uses DCM method calls to reach a motion position. And NoaMo is full functional for *Forward Kinematic* and *Motion Planning*. In future work *Inverse*

Kinematic development will be full functional to reduce computational effort, as done in Forward Kinematic and to develop a new motion stable step. Many RoboCup teams follow that strategy basing they decision on robot motion observation and robot stability.
As already mentioned, one of the uses of stable robot system is to prevent locomotion falling situations. To stabilize an arrangement of joints developers can change the balance of gravitational force to avoid damages. In general, the forces exerted on the feet of a biped robot by the ground involve the vertical reaction forces and the horizontal friction forces. To reduce physics complexity the ground friction forces are omitted in this initial stability developing phases. The physic model can be reduced changing a little our basic motion, if robot on each step reaches a higher feet position. This higher position must be lower as possible to avoid innecesarely movements produced in motion.

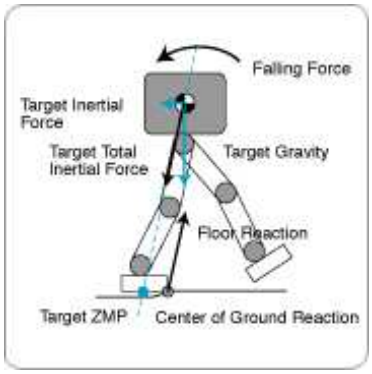

Figure 5.3: ZMP stability action

Many stability methods are used humanoid robots [21]. There are two principal stability solutions:

COM Center of mass[26] in a rigid body is the position of its center and it is fixed in relation to the object, when an object has several mobile parts, all mass object is considered concentrated for body physic study. To generate a stable robot position the COM of the robot must be upper the leg that hold the robot. COM is a method that we can classify a *static balance*.

ZMP A widely-studied class of control algorithm [19, 27] can be developed by computing trajectories that are known to be physically feasible and therefore satisfy the *Zero Moment Point* constraint[21].

$$F^{gi} = mg - ma_G$$

$$PZ = \frac{n \times M_P^{gi}}{F^{gi} \cdot n} \tag{5.12}$$

Small disturbances to the motion can be accommodated by adjusting the ZMP dynamically during the motion. ZMP is classified as *dynamic balance*

and is defined as the point on the ground around which the sum of all the moments of the active forces equals zero. If the ZMP is within the convex hull of all contact points between the feet and the ground, the biped robot is stable and will not fall over. The basic idea is to maintain position by accelerating the upper torso in the direction in which it threatens to fall when the soles of the feet cannot stand firmly (figure 5.3). ZMP method is used in ASIMO humanoid robot developed by Honda. ZMP method with foot planting method are used to generate a stable walking and go downstairs [28].

A ZMP approximation for biped robots are implemented in NaoMo. This new method is called zero friction moment point (*ZFMP*) by authors [27] using ZMP-XYZ as orthogonal coordinate frame. NaoMo implementation a minimum method modification was done to control robot stability in cartesian space over D-H robot model.

Chapter 6

Image Perception in Nao Robot

The Nao robot's principal perception sensor is based on a camera image input. Nowadays, computer vision applied to the robotic environment increases drastically in the same range that processors computation capacity do. The computation capacity and the advantage that an image environment can facilitate much more information than other perceptual sensors, as sonar or laser inputs, explain that increase. Vision is undoubtedly the human sense that we have come to depend upon above all others, and indeed the one that provides most of the data we receive. Nonetheless, the amount of information that an image provide could be redundant or unnecessary then the image perception system based must reduce or treat that information to overcome the objectives that system are looking for. Some people use that reason to say that Computer vision is the science and the technology of machines that see.
The current Nao robot standard architecture has two cameras [8] localized at head. The previous Nao Robot version had got only one camera and the RoboCup Standard League community considered convenient to include a second camera in the new release. The architecture needed change has two principal reasons. The first one was based on the hardware limitation to see the ball when that ball is closed to the robot and the robot is completely erect. The second one become with the need to see the ball while the robot was kicking during a soccer game to implement a feedback movement.
The two cameras are identically, if we are talking about the CCD model [29, 30]. These cameras are exactly located at robot face, as we can see at figure 6.1. Each one provide a 640x480 resolution at 30 frames per second. They are located with an offset of 40 degrees one for each other in the vertical axis, known as Z-axis using the kinematic model shown at chapter 5. The camera angle orientation between cameras and the implicit forbid in RoboCup Standard Architecture Rules about the hardware modification suppose that classic stereo-vision techniques are not useful to be used in this case. Obviously techniques that implements

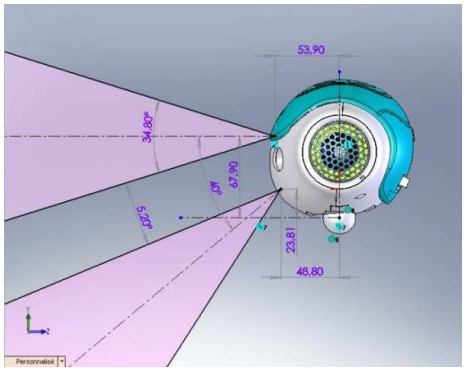

Figure 6.1: Side view of cameras offsets and positions

stereo-vision with only one monocular camera is useful in this case but not the stereo-vision based on zoom because that camera has not got any kind of zoom. The only intrinsic camera parameters provided by robot manufacturer[29] are the diagonal angle of field of view, also know as diagonal angle FOV. And the CCD active area size in pixels (640x480), also in mm are provided by camera manufacturer[30].
As we well known the intrinsic parameters are not the same in all cameras at one determined model. And consequently the cameras' intrinsic parameters has to be determined with a camera calibration method. When researchers talk about camera calibration we often are thinking about determine camera geometric entities such as the projection center and image plane. The first thing that we have to do before is obtaining the intrinsic parameters from a camera is to choose the camera model that we will use in the camera calibration process. The main ones len camera models at acquiring sensor are:

1. Pinhole Model
2. Orthographic projection

In the following sections camera models basic knowledge will be overview of these two camera models.

6.1 Camera Models

The projection of an object surface point of a three-dimensional scene into the two-dimensional image plane can essentially be described as a central perspective or parallel projection. This projective relationship between the two coordinate systems has much importance in scene reconstruction.
The camera models can be classified as two wide classes: *finite cameras* and *infinity cameras*. The main difference is that an infinity camera principal plane is

the plane at infinity. The perspective projection and orthogonal parallel projection are finite camera models. This work uses Pinhole camera model to model Nao cameras' and will be discussed in chapter 6.1.2.

6.1.1 Orthographic projection

Orthographic projection is a kind of parallel projection. Parallel projections are very useful for engineers schematics or working drawings of objects that preserve its shape and scale. Points on an object are projected to the view plane along parallel lines.
When viewing plane normal is parallel to one of the 3D axes the transformation for a determined point (x_i, y_i, z_i) is

$$u_i = s_x \cdot x_i + k_x \tag{6.1}$$

$$v_i = s_z \cdot z_i + k_z \tag{6.2}$$

where s is an arbitrary scale and k an arbitrary offset that are obtained from the viewport situation. These equation can be shown using matrix notation

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & 0 & s_z \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \bigoplus \begin{bmatrix} k_x \\ k_z \end{bmatrix} \tag{6.3}$$

Parallel projection is useful for computer graphic but in camera model with perspective or conic projections cameras can be modeled better using that mathematical model.

6.1.2 Pinhole model

Pinhole model is the most general model and it is considered by some authors the simplest. This camera model is the most used in X-ray cameras, scanned photographic and it principally model CCD like sensors. If we define the *center of projection* as the origin of the Euclidean coordinate system of the projection plane that generate the image. We can consider that the central projection of points in space falls onto a plane. Camera models are used to represent 3D points projections onto a 2D image. Then a point X that can be represented in euclidean space as $X = (x, y, z)$ is mapped to a point on the image plane $X' = (x', y')$where a line joining the point X to the center of projection meets the image plane X'. Mathematically it can be described as a mapping from Euclidean 3 space $\mathbb{R}^3$ to Euclidean 2 space $\mathbb{R}^2$. Pinhole model help us to find a relationship between these two spaces.
We can took under attention the image generation schema at figure 6.2, where a 3D point **P** is represented in a 2D plane as **p**. Note that the origin of the *camera coordinate frame* is located in the lens plane. Len center position is known as *camera center*. The distance between the lens and the image plane is called the effective *focal length* (f). Using basic trigonometry relation we can compute that a point $(x, y, z)^T$, at world frame, is mapped to a point $(f \cdot x/z, f \cdot y/z, f)^T$

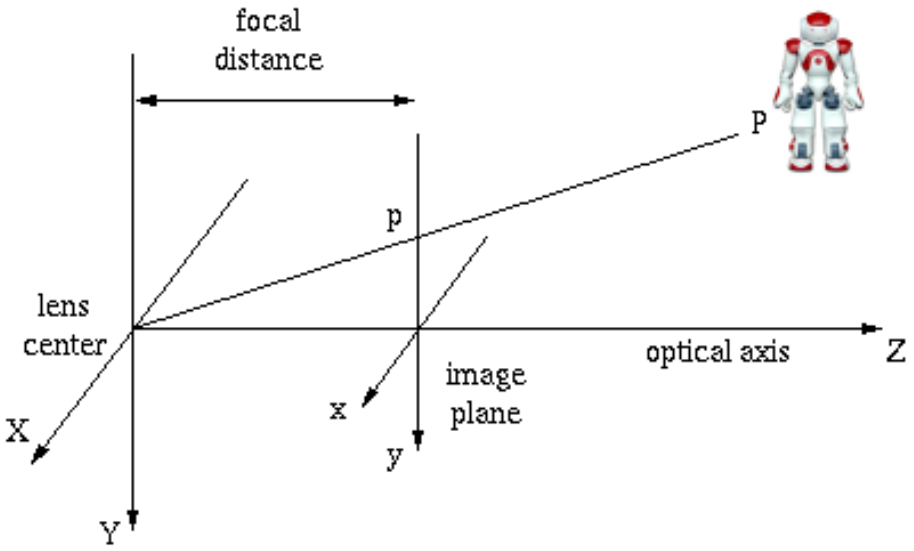

Figure 6.2: General Image forming at Pinhole schema

on the camera frame. Then a point is projected from camera coordinate frame to image frame like formula 6.4.

$$(x, y, z)^T \rightarrow (f \cdot x/z, f \cdot y/z) \tag{6.4}$$

The *principal point* is the point of image plane where *principal axis* or *principal ray* or *optical axis* intersects. Where *principal axis* is defined as the line from the camera center perpendicular to the image plane.
Then a image projection can be modeled as an algebraic operation over a 3D robot working space X point.

$$p = K \cdot P \tag{6.5}$$

Developing the previous formulation (6.5) we can obtain

$$p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{6.6}$$

Ideal camera models determine image plane *principal point* as the geometric image center. In contradiction, real images commonly has not got the *principal point* situated in geometric image center. It phenomenon is denoted in formula

(6.6) using p_x and p_y offset values.

$$p = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{6.7}$$

Other phenomenon that image suffer is non-square pixels. CCDs manufacturing is not precise as vendors wish and CCDs pixel cell usually are not square. It means that euclidean coordinates have not got equal scales in both axial directions. Non-square pixel effect can be removed using a simple algebraic transformation. We can denote m_x as pixel per unit distance in image coordinate in x-axis and m_y for y-axis. Then K matrix will be

$$p = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.8}$$

Where p_i is scaled by the factor m_i as $\alpha_i = m_i \cdot f$. Obviously, axis scale factor transformation affect also to previously matrix components p_x and p_y with factors m_x and m_y respectively.

$$x_0 = m_x \cdot p_x \tag{6.9}$$

$$y_0 = m_y \cdot p_y \tag{6.10}$$

It camera model assumes that the image can be constructed by drawing a straight line form the observed point, through some fixed center point in the optical systems, to the detector surface. That kind of camera model is collinear and considers no distortions. Some models dealing with distortion first convert an observed pixel point to another point that would correspond to a pinhole model, and after that use the pinhole model [31].
Camera model selection is not a easy decision and many articles and surveys are focused at this way as [32] where some criteria are proposed:

Depth *Affine cameras* usually uses a infinity camera model, in situations where object depth is much smaller than the distance to the object. But some CCD device models can not be modeled as affine camera models. Then in that situations a *pinhole model* is the best choose.

Matrix Estimation and Motion Model Some related works deal with fundamental matrix estimation and motion model selection points out to combine two camera models for efficiency and accurate estimation.

Estimation Parameters Affine solution is characterized with less parameters that others as projective camera models. Then affine solution is more stable based on mathematical complexity model. In this case if affine solution is recommended when approximation is good.

Figure 6.3: (a)Left: Image with fish-eye distortion that is easy observable in image corners. (b)Right: Image acquired with forehead Nao's camera.

Simplicity Another interpretation to such choices is the *Occam razor principle*: "Choose the simplest model which still explains the data well".

Kinoshita [32] concluded that projective camera models are more suitable in the most cases. In the cases where the affine camera models are used a small advantage will be given. The final conclusion is that the two models are roughly the same if we consider the information obtained or the camera characterization solution in monocular vision case. Based on the model description pinhole mathematical camera model is the most useful because Nao cameras have CCD device sensor that is better characterized by that model. It is important to denote that all decision over camera models are focused on a monocular camera models.
In this section, image distortion are not took on account Nao's camera has not got much perspective distortion. Also, in the most cases in computer vision algorithm, used in this work, the external image areas are not used. Talking about distortion, eye-fish distortion or perspective distortion always are present in images, and its distortion can be negligible in some cases. In our case, fish-eye distortion can not be appreciate and perspective distortion are almost zero as we can appreciate in figure 6.3. There are two principal perspective distortions: radial and tangential. The radial distortion move the points from the center to the image bounds and the tangential distortion make a perpendicular transfer points over *radial line*. The first distortion is caused by bad len manufacturing and the second one is caused by bad alignment of optical components [33].
Mathematically, it is possible to define a lens that will introduce no distortions. In practice, no lens are perfect. It is much easier to make a *spherical* lens than to make a more mathematically ideal *parabolic* lens. It is also difficult to mechanically align lens and images exactly. The radial distortions arise as a result of shape of lens. Tangential distortions arise from the assembly process of camera.
Radial distortions bulging phenomenon is the source of the *barrel* or *fish-eye* effect that it is usually present in cheap cameras. The distortion is commonly

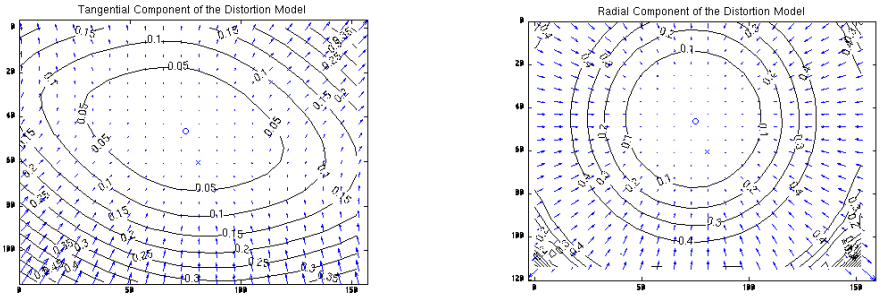

Figure 6.4: Nao camera tangential and radial distortion

small and can be characterized by the first few terms of a *Taylor serie*. These terms are usually called with the convention $k1, k2, \ldots, ki$ but in general cases no more than 3 terms are used. When the third component is zero then the camera distortion can be considered small and a quite good camera. It is the our case where the third component of the Taylor serie is zero. With a three component distortion serie the corrected point can be calculated with the formula 6.11

$$x_{corrected} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{6.11}$$
$$y_{corrected} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6)$$

Also tangential distortion can be characterized by two parameters p_1 and p_2

$$x_{corrected} = x + (2 \cdot p_1 + p_2(r^2 + 2 \cdot x^2) \tag{6.12}$$
$$y_{corrected} = y + (p_1 \cdot (r^2 + 2y^2) + 2 \cdot p_2 \cdot x)$$

6.2 Camera Calibration

In previously section basic camera parameters was introduced for pinhole model and orthographic projection. In this section the common methods to obtain this internal camera parameters will be overview. As mentioned above, to determine camera parameters we approximate Nao's cameras with *Pinhole camera model*. Once pinhole camera model is selected as the mathematical model to characterize Nao's cameras. The follow step is to determine camera internal parameters. Camera parameters can be determined using 3 main techniques[34, 35]:

Self-calibration, Just moving a camera in a static scene, the rigidity of the scene provides the internal parameters of the camera. When camera's internal parameters can be modified is important that these parameters will be the same in the calibration process.

Figure 6.5: Images acquired with Nao robot. Images with principal rays on it crossing at vanishing point.

Photogrammetric Calibration, Camera observes objects in 3D space whose geometry and location are known with a very high precision. The calibration objects usually consists of two or three planes orthogonal to each other and setting up 3D space phase is very difficult and calibration apparatus are usually expensive. In contradiction when 3D space is well defined calibration usually are very good.

Vanishing points for orthogonal directions, Processing Vanishing points at image some internal parameters can be found but previous knowledge of focal length is necessary. Figure 6.5 shows vanishing point and principal rays on images acquired with Nao camera.

Nao's cameras calibration apparatus that are necessary for an photogrammetric calibration can not be used because we can obtain that expensive technology. In other side, self-calibration methods are most used by researchers over the world because calibration results are similar that the other techniques and are cheaper and the easier way. To perform calibration 3 steps was follow:

1. Camera intrinsic parameters processing using *DLR CalLab* developed at Institute of Robotics and Mechatronics.

2. Camera intrinsic parameters processing using *Camera Calibration Toolbox for Matlab* developed by Jean-Yves Bouguet at Intel Corporation.

3. Empirical and mathematical test of previously obtained values

Abdel-Aziz and karara work and their *direct linear transformation* paved the way for computer vision algorithms finding a solution to linear equations based on the basic collinearity camera model. Nowadays, it can be seen as unacceptable because their ignore lens distortion in the process but their founded the bases. Later, Tsai gives a more complete camera model with some restrictions as Abdel-Aziz model but simplify the formulation using *radial alignment constraint* that basically reduces dimensionality of the problem. As earlier works,

the calibration process requires calibration objects and accurately moving on a planar surface. Other approximations were presented as Weng work. The best contribution on camera calibration was done by Zhang presenting a closed-form solution using *linear least-squares techniques* with aims to relax scene conditions and it gives freedom to camera moving.
Camera calibration technique selected to obtain internal parameters is self-calibration. To perform that calibration a grid pattern was used (figure 6.6) as a fixed scene and then only camera motion was needed to obtain internal parameters. One sequence of 3 images are necessary to obtain calibration but a more precise calibration a 5 images sequence are required. In Nao's cameras calibration process a 5 sequences of 7 image each one was employed to calibrate internal parameters. Also, when more images per sequence are used than the minimum number required give strong to calibration process. And an error while featured points are selected from grid pattern images then is minimized. Internal

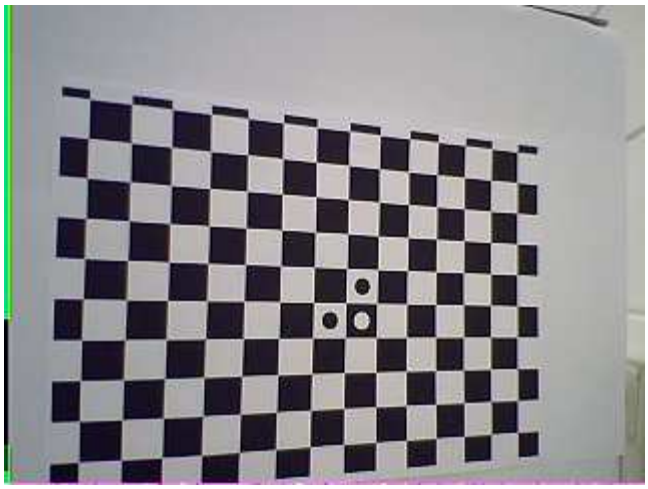

Figure 6.6: Grid pattern image used at internal camera calibration

camera parameter calibration process was designed with 3 phases because the two selected software tools are guaranteed by two prestigious companies (Intel and German Aerospace Center) but tools were not used before by us. It supposed some kind of suspicion over they functionality. Using the new software utilities always a mistake can be done supposing that developer instructions are following rigorously, then a test is a essential part. Obviously using two very different software, results can be compared directly and any problem can be easily detected. The third step, an empirical test and mathematical prove, is used to control the correct tool functionality of the geometric, or mathematical, camera model.

6.2.1 DLR CalLab intrinsic parameters calibration tool

DLR CalLab tool was developed by Institute of Robotics and Mechatronics. Institute of Robotics and Mechatronic is part of German Aerospace center. This tool is a platform independent software that provides a basic camera calibration tool for monocular systems and stereo-vision systems. As explained in previous sections, Nao can not implement stereo-vision algorithms in acquired data because images are not superimposed. This software utility also can calculate the image distortion quotient as many other image characteristics. DLR CalLab tool calibration process is detailed described in [36, 37, 38]. DLR CalLab tool has some advantages: it can calibrate any type of grid dimensions, can detect automatically features in the calibration grid (corners) and you can erase isolated or bad detected features before the calibration process to increase calibration precision.
The authors consider that research on camera calibration for computer vision applications has arrived at a point where most of its components have become standard: Pinhole camera model (or another perspective projection model), tangential and radial lens distortion model, feature detection usually based on corner detection algorithms, the planar calibration object although some authors defense other calibration objects and camera parameters estimation algorithms. DLR CalLab tool give a robust solution to all these steps.
DLR CalLab calibration process as the common calibration utilities is based on Zhang approximation using *linear least-squares* techniques but erase the method restrictions on accurate knowledge of the grid pattern used in calibration process.
The accurate camera calibration process start with a detection and identification of the control points denote as ${}_0x_i$ in a cartesian space determined with an homogeneous vector ${}_0x_i = [x_i, y_i, z_i]^T$. ${}_0x_i$ is perspective projected onto the image plane and compared with the expected ones ${}_nm_i$ denoted as ${}_nm_i = [{}_nu_i, {}_nv_i, 1]^T$ at point in image n. Then a Euclidean decomposition of the perspective projection matrix P are estimated where an arbitrary s scale factor is used. Camera calibration software usually uses a region window to locate ${}_0x_i$ at image.

$$s \cdot m_i = A_c T_0 \cdot_0 x = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6.13}$$

Where A matrix is the internal camera parameters also called *intrinsic* matrix. ${}_cT_0$ is the rigid body where camera are fixed. If we locate camera matrix frame at ${}_cT_0$ position this matrix is identity matrix and minimize calculates.
In this particular case can be solved with a linear least-squares criterion if almost 3 different views are available because are linear independents. All depends only on the orientation of the plane and not on its distance or scale and solution can be obtained with geometric similarities or with a Euclidean distances.

6.2.2 Camera Calibration Toolbox for Matlab

Camera Calibration toolbox is a simplest compared with DLR CalLab. In contradiction, this toolbox is based on Zhang method [39] and it is useful to compare result from the automatic method presented by DLR CalLab and the Zhang classic method.
In this method, camera is modeled by the usual pinhole relationship between 3D point M and its perspective projection image m defined by the author as

$$s \cdot m = \begin{bmatrix} R & t \end{bmatrix} \cdot M \tag{6.14}$$

where s scaled factor are used too and (R, t) are the camera rotation and translation frame from base frame. This method find a model point M and its image m relationship using homography H. Given an image of the model plane, an homography H can be estimated. The algorithm limitation is that homography has 8 DoF and there are 6 parameters that represent rotation and translation then only 2 contraints on the intrinsic parameters can be obtained for each calibration process. An iterative algorithm solved that problem finding pairs solutions.

6.2.3 Empirical and mathematical test

Calibration process was not a trivial process as commented before until Zhang presented his method. Nowadays, there are some mathematical tools that helps to calibrate the camera's intrinsic parameters.
These tools are a good support to computer vision researchers which focused they work in perspective projection field but are not interested in camera calibration process.
In our case, the camera models were studied and some tools were used to do camera calibration process. But before continue with perspective projection techniques a complete result revision was needed.
The test was divided in two phases:

1. **Mathematical test**, where results obtained from the two tools are compared
2. **Empiral test**, where the camera parameters are tested.

If the equation 6.14 is observed can be seen a relation between a pixel point m and a world three-dimensional point M. It relation is multiplied by a scaled factor s. Also we know that the camera's parameters are unique. Then the calibration matrix it will be unique, too. As it was expected the two tools used to calibrate cameras returned different results. It happens because the matrix have a different scale factor, as equation 6.14 indicates. Once the matrix was compared, we can denote that they are a equivalent matrix, as algebra defined. Also some mathematical test were done to compare expected results with obtained with camera calibration tool. We know that

$$x = f_x \frac{X}{Z} + c_x \tag{6.15}$$

Figure 6.7: Two images from a set used for calculate horizontal FOV.

$$y = f_y \frac{Y}{Z} + c_y$$

where $f_i = F \cdot S_i$ and i can be x or y. The F value is the physical camera focal length that in our case is $2,80$ mm. Also we know that

$$f_i = \frac{Image\_Side_i}{2} \cdot \cot(\frac{FOV_i}{2}) \tag{6.16}$$

Once, the matrix values were applied to the respective equations the $FOV_{vertical}$ and $FOV_{horizontal}$ were fixed:

$$FOV_{vertical} = 34,80^\circ$$

$$FOV_{horizontal} = 45,5858^\circ$$

Finally two empirical simply tests were done. Some Images where taken from a known distance D from some objects and then the focal length angle was calculated. Picture 6.7 shows some of the used images.
The objects dimensions are know previously, then with a geometry equation (6.17) we can determine an empirical FOV.

$$FOV_{horizontal} = 2 \cdot \arctan(83/D/2) = 45,0769^\circ \tag{6.17}$$
$$FOV_{horizontal} = 2 \cdot \arctan(62/D/2) = 34,4468^\circ$$

where D is the distance from camera to object. The values 83 and 62 are the dimensions of the object width in millimeters.
The empirical results and the theory results obtained from camera calibration are similar. The values differences lie in the empirical error estimation because the distance must be measured from robot camera to the object. Also, we consider that results are good and camera calibration parameters are tested successful.

6.3 Intrinsic and Extrinsic Camera Parameters

By one hand, we obtained with kinematic Nao model a mathematical transformation from feet to any robot chain as seen in chapter 5. By the other hand, camera model allow us to translate a point in 2D space to a point in 3D space. Then a point in 3D object space can be described in robot coordinates. And a point in 3D object space can be described as a point in the input device. Our input devices are principally the Nao's cameras and a point in 3D object space is perspective projected to an 2D image space as seen in this chapter. Using a simple algebraic transformation from camera frame to feet frame and object in object space can be situated in 3D robot space. Object-robot transformation are commonly known as hand-eye calibration [31, 38, 40, 41].
The name comes from industrial robot environment where closer to the robot end-effector a camera was plugged to supervise or to control the robot tasks. It is important to denote that 3D object point can be situated in 3D robot frame space but not in world space if robot are not clearly located using a GPS, a static position as in industrial robot case or located using vision algorithm with beacon techniques for example.
In hand-eye problems the internal camera parameters are called *intrinsic parameters* and robot kinematic pose is called *extrinsic parameters*. Usually then a point in 2D space is denoted as $[u, v, 1]$ in homogeneous coordinates and a 3D point as $[x, y, z, 1]$. Using 6.3 equation, it assigned extrinsic parameters in TCP (Tool Center Point) denoted by ${}_cT_0$ and intrinsic camera parameters denoted by A.

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = A \cdot {}_c\, T_0 \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{6.18}$$

Hand-eye calibration technique gives to computer vision algorithm a useful information because objects detected by vision system can be situated in 3D robot space. Also, this information can be used to avoid detection errors. For example, in a controlled scenario as RoboCup game field where goals have known height all image points situated over goals' height can be discard.
Robocup Field conditions are controlled but all the elements outside the field are unpredictables. People closer to the game field can be dressed with same colors as elements in game field, or environment location can contain colours than can be considered as noise (figure 6.8). In literature the line, over pixels are upper-floor level, is know as *horizon* [42, 43]. *Horizon* is determined in 2D space using kinematic transformation. Horizon line is the line created at infinity where soccer field seems to disappear from the camera perspective obtained using the height of the camera. When head goes down the horizon line in the image goes up and vice versa. Another important geometry image element is the *geometral line*. *Geometral line* in a 2D image is a plane in 3D space as *Horizon line*. Geometral line also known as *ground-aligned plane* is the line over pixels are avoided. All two planes Horizon and Geometral are parallel and perpendicular two the scene image plane.

Figure 6.8: RoboCup game field where environment elements and game field have the same colours.

Using Kinematic *Head frame*'s, as the transformation frame axis, the algebraic transformations to obtain Horizon Plane:

1. Rotation about X-axis if there is body roll θ_{BR} angles.
2. Rotation about Y-axis if there is body pitch θ_{BP} angles.
3. Translation from *Head Frame* to *Camera Frame*, $(0, T_y, T_z)$
4. Rotation about Z-axis if there is head yaw θ_{HY} angles.
5. Rotation about Y-axis if there is head pitch θ_{HP} angles.

The transformation matrix will be denoted as R in future use. The three first steps are obtained directly using NaoMo Forward Kinematic transformation as a matrix Rotation R_B. The fourth and the fifth steps are calculated using a matrix Rotation over Z and Y-axis with θ_{HY} and θ_{HP} respectively. Once Horizon Line is determined by $(h_l,\ h_r)$ points in Horizon Plane. To calculate horizon line two main internal camera parameters are needed: the width screen resolution $2 \cdot s$ and *horizontal opening* $2 \cdot \alpha$ *angle* $(aFOV_h)$, with these camera parameters knowledge we can determine horizon line with two points $\vec{b}_{l/r} = (b_l, b_r)$.

$$\vec{b}_{l/r} = \begin{bmatrix} \frac{s}{\tan(\alpha)} \\ \pm s \\ z_{l/r} \end{bmatrix} \tag{6.19}$$

Then the only unknown parameter are $z_{l/r}$ using (r_1, r_2, r_3) components of kinematic rotation matrix R obtained in this section in a previous step.

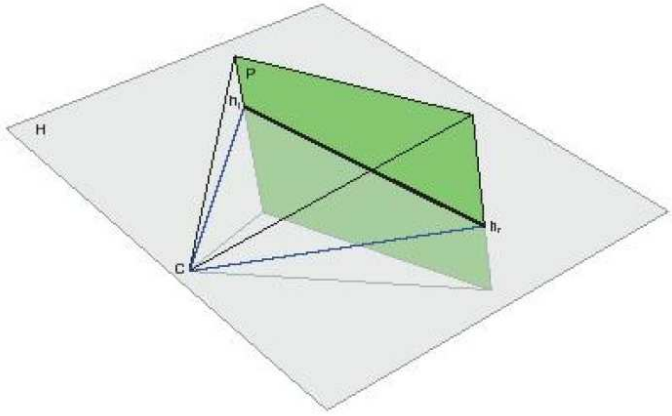

Figure 6.9: Image plane (P) intersection with Horizon Plane (H) denoted by (h_l, h_r).

$$z_{l/r} = \frac{\pm r_{32} \cdot s - r_{31} \cdot s \cdot \cot(\alpha)}{r_{33}} \tag{6.20}$$

The importance of compute Horizon line and Geometral line in image space are discussed on Segmentation chapter. 7.

Chapter 7

Image Segmentation

One of the most important steps in computer vision is the image segmentation. Image Segmentation aim is to divided an image into parts that have a strong correlation. Then it divides image in objects or areas of the robot world contained in the image. Then a segmentation is a set of disjoints regions corresponding with objects in the input image. It is a difficult goal to get a complete disjoint areas depending on object regions. Commonly the principal objective of image segmentation algorithm consists on find disjoints areas that correspond to one or more objects. Then areas are divided into separate regions that are homogeneous with respect to a chosen property such as color, texture, brightness or another image property. Segmentation methods can be divided into three wide groups[44]:

Global Knowledge The most common algorithms based on global knowledge are represented by *histograms*. Histogram thresholding is the simplest segmentation process. Many objects or regions at images are characterized by constant reflectivity or light absorption of their surfaces. Then a *threshold* can be determined to segment objects and background. Many techniques have been proposed to implement in 1D where peaks and valleys of the brightness histogram can be easy identified respectively with objects and backgrounds of gray-level images. Mathematically thresholding is the transformation of a input image f (function f) to an output binary function g as an image representation:

$$g(i,j) = 1 \text{ if T} \geq \text{f(i,j)}$$

$$g(i,j) = 0 \text{ if other case of f(i,j)} \tag{7.1}$$

A complete segmentation of an image I is a finite set or regions $I_1, I_2, \cdots, I_n$

$$I = \bigcup_{i=1}^{n} I_i \tag{7.2}$$

where $I_i \cap I_j = \emptyset$ and $i \neq j$. Many practical segmentation problems need more information than is contained in one spectral band. Color images are a natural example in which image is coded in three spectral bands called red, green and blue in RGB color representation. In the case of color images, the algorithm computation is little more complicate because it has to identify different parts of a scene by combining peaks and valleys of three dimensional data. Some authors show us different ways to segment images using thresholding based on maximizing within-group variance, watershed algorithm, using LUV system and coarsened trough convolution with a spherical windows, segmentation using a unique component Hue in HSI system or entropy based thresholding in LUV system to cite a few.

Edge based Edge-based segmentation methods are based on information about edges in the image. Perhaps, these kind of methods are the earliest approaches and still remain important in some computer vision fields. Edge-based segmentation algorithm based it functionality on find discontinuities in color values. Some edge-based algorithm are edge thresholding, border tracking when regions have been defined and borders are unknown and the well known Hough transforms.

Region based Edge-based and Global knowledge segmentation are methods that are based on find borders. In contradiction Region-based methods construct regions directly. The *region growing* methods are usually good when noise is present at regions. Homogeneity is an important property of regions and it is used as the main segmentation criterion in region growing. A criteria for homogeneity can be based on gray-level, color, shape and some others. Its criteria was first well defined by Haralick and Shapiro and some other authors proposed some criterias later. Region growing can be obtained through a growth process which a preselected seed are used. The region growing ca be considered a sequential clustering or classification process. The main advantage offered by this kind of techniques is that regions obtained are certainly spatially connected and rather compact.
Clustering can be defined in contradiction as a non-supervised classification of objects in which one has to generate classes or partitions without any prior knowledge. In last years, several techniques have been proposed in the literature of cluster analysis. The most important and use method is the called *k-means*. K-means have several variants too, some of this can be the possibilistic approach of k-mean fuzzy.
Other segmentation method that have a closer relationship with homogeneity is the *Split-and-merge* technique. The most important characteristic of the method is that start in a inhomogeneous partition of the image and they keep performing splitting until homogeneous partitions are obtained. When all regions are homogeneous partitions a merge phase join all similar regions to obtain the minimum homogeneous regions. The most known data structure used to implement the method is called *Quadtree*.
Another clustering algorithm is Mean shift. Mean Shift is an unsupervised clustering algorithm (Fukunaga and Hostetler, 1975), which estimates the

gradient of a probability density function to detect modes in an iterative fashion. Mean shift segmentation has been successfully applied to several applications (Comaniciu and Meer, 1999). In contradiction when for larger images or applications where processing time is very crucial, the mean shift segmentation algorithm might be still too time consuming. Even with optimization techniques applied to Mean Shift by Comaniciu and Meer based on EDISON algorithm.

As few years ago when segmentation techniques are focused on gray-level images because processing color images requires computation times considerably larger than those needed for gray-level images. Nowadays, a common computer systems permit high level performance but embedded systems that must execute a determined task in real-time response not allow the use of computational expensive methods. We are talking about classic or robust methods that results are not obtained in a real-time elapsed. Then when almost segmentation problems are solved or segmentation optimal result are closer to be obtained, real-time requirements one more time put to the test researchers ingenuity to obtain the best results.

7.1 Image Segmentation in Nao Robots

Image segmentation is a low-level method but usually it is the most important stage in computer vision. Middle-level and high-level algorithms good functionality depends on a part of the quality of image segmentation process results. When an image is acquired from camera sensor the image segmentation process is used as an information reduction or an information enhancing. Image segmentation methods usually have a closely relationship with colour image representation model. Then before take a full overview of segmentation techniques used on Nao robot a full reflection of the colour model chosen in computer vision are going to be taken.
Nao's camera described at chapter 6 can work in few colour models. These colour models are YUV, RGB and HSI described in the preliminars chapter. In colour segmentation techniques and *feature extraction* algorithm the most important and recent algorithm are based on HSI colour representation model [45] especially in face detection.
Then if robot support HSI colour model approach is easy to think that HSI model is the best colour model to use in image segmentation. At table 7.1 acquiring times of Nao's camera are presented. If values are observed accurately, the most significant thing that can be observed is that HSI color model acquisition times are the biggest and then the slowest time response. Nao's perceptual vision system has real-time response needs. Then the HSI colour model has to be ruled out and we have to choose between RGB and YUV. YUV colour model is most intuitive and it has more information than RGB as seen in the previous chapter. Then YUV model has been chosen as native colour model in NaoVi framework.

In RoboCup soccer field objects have a specific colours to facilitate vision task, concretely image segmentation. Few segmentation methods have been introduced in this chapter such as global knowledge based, region based algorithm or edge-based. These methods can be used in image segmentation but before doing a segmentation we have to determine a function or method to distinguish useful or avoiding information.
The segmentation algorithm must be a multi-spectral or multi-channel filter to detect all objects in image scene. Also, the segmentation algorithm must be a simple channel filter, but it has to detect 8 objects, with different colours. Then, a simple channel filter use means that a 8 image scan must be necessary to detect objects in the game field. Then to optimize image segmentation response time the best solution is to implement a multi-channel filter.
For each channel segmentation process has to distinguish between channels and noisy background, in this case a multiple thresholding per channel is the best solution. Each channel are going to be determined by a maximum value of Y channel, a minimum value of Y channel, a maximum and a minimum value of U channel and a maximum and a minim value of V channel. These minimum and maximum values has to be disjoints between all colours present in soccer field to perform a good segmentation method.
Rodrigo's article [46] made a complete study of YUV used at RoboCup soccer field and he concludes: all pixels belonging to the interesting color classes are grouped in a determined region of the YUV space and the regions intersect one another. Also, he finishes the conclusion with known that it is impossible to make a perfect training in with every pixel is uniquely classified because color confusions are inherent to the problem.
Another thing has to be on mind, embedded robot camera is common to be cheaper than other industrial cameras. This factor joined with image resolution or image size makes common to increase noise at image, and it causes that colours are very different as in real scene.
Principal Naovi color representation model is YUV, however in some situations where light conditions are bad or when light has a huge yellow component the segmentation becomes a hard task. Then to solve the segmentation problems another approach was developed in NaoVi architecture; time limitations (7.1) force us to use YUV422 image format that is served in 5.0 ms and is faster than the others and concretely as HSI (24 ms). Segmentation light problems are lower or less influence where HSI color model is used, then we considered necessary to work in HSI. Obviously, we can not acquire an image in HSI format because camera time response is too slow. In contradiction a pixel conversion

Table 7.1: Image acquiring times in Nao robot.

Image	YUV422	YUV	RGB	HSI
QQVGA	1.6ms	5.3ms	7.5ms	9.7ms
QVGA	5.0ms	6.2ms	15ms	24ms
VGA	20ms	24ms	60ms	94ms

from YUV422 to HSI will be very fast. Then a pixel conversion(7.3) was used to acquire image in HSI format.

$$r = Y + ((1436 * (V - 128)) >> 10) \tag{7.3}$$

$$g = Y - ((354 * (U - 128) + 732 * (V - 128)) >> 10)$$

$$b = Y + ((1814 * (U - 128)) >> 10)$$

$$h = \begin{cases} 0, & \text{if } \max = \min \\ (60^\circ \times \frac{g-b}{\max - \min} + 360^\circ) \mod 360^\circ, & \text{if } \max = r \\ 60^\circ \times \frac{b-r}{\max - \min} + 120^\circ, & \text{if } \max = g \\ 60^\circ \times \frac{r-g}{\max - \min} + 240^\circ, & \text{if } \max = b \end{cases}$$

$$l = \tfrac{1}{2}(\max + \min)$$

$$s = \begin{cases} 0, & \text{if } \max = \min \\ \frac{\max - \min}{\max + \min} = \frac{\max - \min}{2l}, & \text{if } l \leq \frac{1}{2} \\ \frac{\max - \min}{2 - (\max + \min)} = \frac{\max - \min}{2 - 2l}, & \text{if } l > \frac{1}{2} \end{cases}$$

Image conversion makes some overhead in image acquiring process but it always is less than 1 ms and comparing times with image acquiring from camera is 24 times faster.

The chosen color model can be used to segment the image scanning all image doing a 6 comparisons per each channel. Then a 8 channel filter needs a 48 comparison per each pixel. To reduce this comparison cost a better solution can be used. Using a LUT (Look-up table) segmentation time can be reduced. LUT segmentation uses each pixel value as a table index, in the table are stored a label that determines which region is part of.

For example, using a pixel value of an image that is part of soccer field as a table index we will obtain a label that it determines that this pixel is part of soccer field. In other example, a goal pixel value as a table index will be determined that this pixel is part of a goal.

LUT is the faster that inexpensive region-based or another introduced before. LUT segmentation approach only has one disadvantage, LUT needs a minimum memory allocated to work. One-dimensional table with a three component colour model representation, the index is created using equation 7.1

$$index_{LUT} = \text{ Y} * 65536 + \text{U} * 256 + \text{V} \tag{7.4}$$

Then in a colour model with 3 component colour representation with 24-bits, where maximum number representation is 255, a 16MB (256 * 256 * 256)is the minimum memory allocated [3]. In robot embedded systems as in Nao case, where total available memory is 256MB, a simplest structure as LUT that needs 16MB is to expensive. In Sen article, a new LUT approach is presented to avoid expensive allocation. This new approach has a simple idea but usually

simple ideas are better than others because it can be implement easily. Author proposed a binary LUT, then a 8-bits component representation is enough and only 768 bytes are needed in new approach.
Also LUT table access become more simple and computationally efficient:

$$index_{LUT} = \text{ Y } << 16 + \text{ U } << 8 + \text{V} \tag{7.5}$$

This work introduced NaoVi colour representation (YUV) and a technique to distinguish easy and computational efficient method (LUT). The next step is to introduce to real-time segmentation techniques in the recently state-of-the-art. Researchers, that focused it work in RoboCup problems, have been proposed in the last years some segmentation algorithm to be applied during a RoboCup robotic soccer game ([42, 47, 48, 49, 50, 51] to cite a few) or to be used in a wide range applications ([10, 52, 53, 54, 55] to cite a few).
All algorithms and methods that are going to be developed in NaoMo and NaoVi frameworks has two common characteristics: robust and real-time approach. A segmentation method as the classical *Mean Shift*, *Run-Length Segmentation* or *Region Growing* algorithm has clearly the first characteristic. A robust method in segmentation process will give to us a good data information as an input to middle-level or high level computer vision algorithm. Maybe a fast implementation approach of previously cited segmentation algorithm can response in real-time elapse, but in that cases usually a computational effort is required. Nao's cpu has a computational limitation.

A simplest data reduction in image case is to reduce the input image data. In other words, Nao's camera works with VGA, VGA/2 and VGA/4 image format. Image information that segmentation algorithm can be extracted from a VGA and VGA/2 is almost the same and then a VGA/2 is the best choose in a real-time application because we have a half data reduction. In VGA/4 image size the colour areas are too small and in some cases pixel colour change caused by camera hardware. Then the best choose is VGA/2 image size.
Using a intermediate image size we obtain a data reduction that it is directly related with any algorithm. The next step to optimize image segmentation then we will focus our attention on segmentation algorithms. A solution to data treatment when its data is too huge is based on *image sampling*. Image sampling as it name says, consist of computing only some pixels at image. Objects of interest often span over relatively large areas, it is usually not necessary to check every single pixel to find them. It is possible to locate them with a low-resolution search. Sampling can be in 1 direction or over the 2 image directions. Also sampling can be done choosing some random pixels at image that can be distributed with some pattern.
1 direction sampling can be done scanning image up to down(or vice versa) or left to right (or vice versa), also known as *scanlines* technique [56]. Two directions sampling consist on doing image scanning for the two image axis, this approach often is known as grid [57]. Image scan frequency that can be based on empirical experiments, based on the objects geometry in scene or object size in scene.

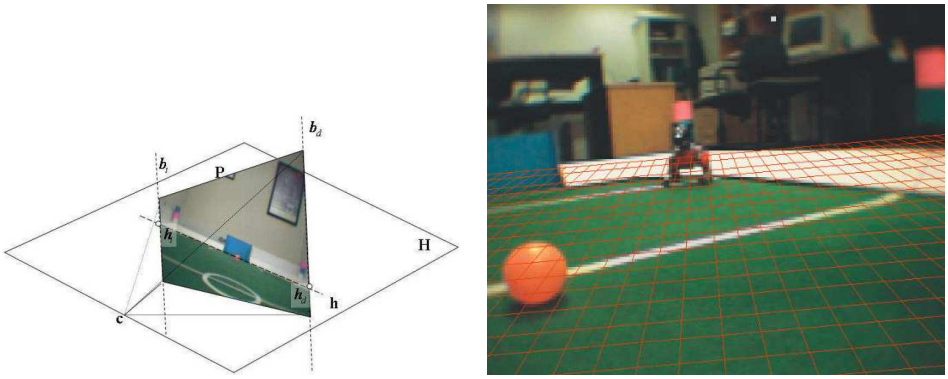

Figure 7.1: a) Horizon Line detection b) Segmentation based on dynamic grid sampling method.

Grid sampling can be done with a static dimensional grid or with dynamic size. Also as in Bach 2005 article, the grid can be applied to a virtual floor situated over the real floor at image plane. To apply dynamic size grid as shows figure 7.1, where closer square are bigger to further square, the geometral line has to be well defined. Vertical grid-lines construction is almost easy, these can be obtained by considering lines that originate at the vanishing point an meet a line at the footing of the agent with required spacing the horizontal lines are not as easy. If optical angle from camera to ground θ is known, with kinematic model it is resolved and ρ the optical vertical angle we can denote:

$$\theta = \frac{r}{2} \frac{sin(atan(\frac{x_3}{\sqrt{(x_1^2+x_2^2)}}))}{\sin \rho/2} \tag{7.6}$$

where x is the vector in camera coordinates parallel to the optical axis of the camera. Another image sampling technique that is commonly used in RoboCup environment is to generate a two levels image sampling in 1 direction case. The image are divided in two regions that we can call *upper-region* and *bottom-region*. In each region scan frequency is different to the other region to adapt better the scan algorithm. In RoboCup soccer game case there are two approaches:

1. Upper-region has less frequency than bottom-region because objects usually situated in the top part of the image are bigger than the objects in the rest of the images. These objects are usually robots and goals.

2. Bottom-region has less frequency than the upper-region because objects are closer to the robot.

Our segmentation algorithm [58] use two levels 1-direction image sampling. The division line is determined with using *Image Geometral Line*. Image Geometral line set the upper line where green field are present at image. The Geometral line is parallel to the robot feet plane. In images where green field are not

present the image geometral line is fixed to the last image row. Then goals are between horizon line and geometral line located. And game ball and field lines are present down the geometral line. We defined *Horizon Line* as the line at image where over it line the pixels at image are ruled out. Nao robots can be situated in whole image but under horizon line.
Horizon line is more important than Geometral line. Horizon line is useful to discard image regions that it is not of our interest. For example, we can discard the images that contains regions upper the goals. Discard upper region is not as easy as it can be thought. Upper goal regions can be removed using two techniques in our work:

1. The goal can be found and then the upper region can be removed. This first approach needs to make a low frequency vertical scanning. For a $VGA/2$ image format only 10 lines are necessary to detect the goal and then the horizon line is fixed. If goal is completely seen, we can define horizon line using a linear regression over the 32 points obtained in scan-lines phase. Also the RANSAC method can be used to define the horizon line. This first method to detect horizon line is only useful is the goal can be seen, partially o completely, in other case the computational effort does not give any additional information to us.

2. Once robot kinematic model is defined, as camera's intrinsic parameters, an inverse perspective projection can be done for a set of points at image to locate upper goal pixels than can be removed at image. In this second approach, a *virtual scanline* over the image is necessary to locate the ruled out region. Two only scanlines are necessary to detect the upper-goal region. The horizon line method scans upper to down two pixel columns at image. For each pixel an inverse perspective projection is done to situate a 2D image's point in the 3D robot coordinates. Then the height is compared to goal height, on which a constant is added to minimize error estimation. The if the point is situated up and very close of the theorical upper goal position the horizon line is fixed to that position. The compare operation is done to both scanlines.
   Also, perspective projection and kinematic model can be applied to locate the horizon line at image, once robot height and goal height is known.The two points, to generate a line in the image plane, in 3D space where upper pixels can be removed must be known previously.
   The two theorical points are projected to image 2D space and three cases can happen:

   (a) The line is located up the first image line. Then no one pixel or region is removed.

   (b) The full image is located up the horizon. In that situation robot is looking at ceiling and then the full image is removed.

   (c) The horizon line is located at image, then the region upper than the horizon line is removed.

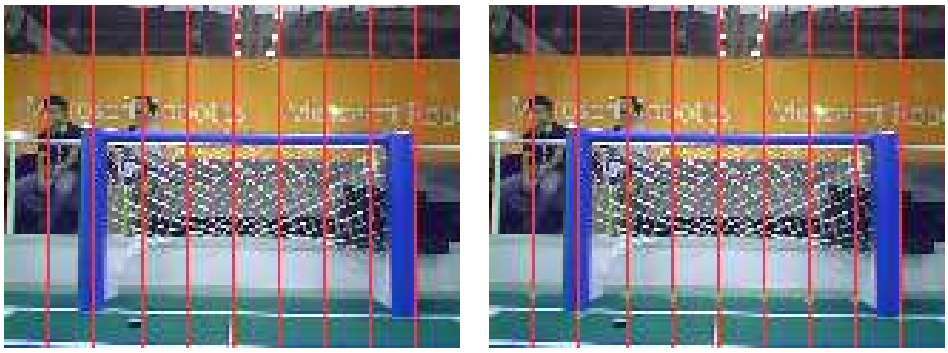

Figure 7.2: a) Image scan searching geometral line b) Points selected to generate geometral line.

Geometral line is defined as the line where geometral plane (green field) end at horizon. In other words, geometral line locates the place where green field ends. Geometral line divides the image in two regions, the bottom region where green field is present and upper region where there is not green field.
Geometral line helps us to eliminate false detections because we know that balls and lines must be located bottom geometral line. Vertical goal post are closer to the geometral line and the goal is between horizon and geometral line. There is developed two methods to detect when a point is upper or lower than the geometral line. The two methods begin making a 9 lines vertical scanning. Then the first green point detected while line were scanned is stored. The first one trace one line using the selected points at image using a linear regression equation or RANSAC method. It method is fast because the calculate only take on account 9 points. In contradiction the method does not take on account all possible cases. For example, if on field corner is seen by the robot the method will scan the image to found a geometral line.
The second method once the selected points where found, they are used as a *representative* of a region. The region is defined between two scanlines $(-freq\_scan/2 < scanline_i < freq\_scan/2)$ and when a pixel is going to be characterized as a up or down the geometral line, it sis compared with the close *representative*. This second method is a split of two regions and the regions limit must not be a line.
Segmentation methods main problem is the light conditions. In RoboCup competition robots are playing soccer over all day and light usually change drastically in few hours. A segmentation method that can be independent in changing light conditions is difficult to develop. Some recent publications said that differences between YUV components are invariant also when light change. This means that an uncalibrated segmentation algorithm based on YUV colour model can be designed only with regions difference as a homogeneity measure. Our effort now are focused on a YUV differences between colours to get a robust segmentation algorithm. But until the new segmentation algorithm will be de-

veloped as an unsupervised algorithm, nowadays our LUT based algorithm has to be calibrated. In the first development phases the calibration was be done manually but it required a few effort every time that light conditions change. To overcome that problem a k-means clustering technique is used nowadays to calibrate LUT segmentation structure. K-means algorithm and calibration phase will be revised in section 7.2. All colours at game field are known and it facilitates the convergence of k-means algorithm determining previously some seeds that they are used as a centroids. Also centroids are used to be compared with obtained regions in the LUT auto-calibration method in the region labeling phase.
When an over-segmentation is present in k-means based LUT auto-configuration method a human supervise is necessary. In this work sampling segmentation is used because it is a simple and a fast segmentation technique that empirical result shown that it is robust too. Also another fast computing techniques can be used in segmentation to reduce segmentation time response as fuzzy-logic segmentation [59] that can be adaptive to overcome changing light conditions or artificial neural network methods. Statistical methods can be used too, but it usually tends to be computational inefficient because a second moment calculate is necessary to do a robust statistical method.

7.2 Auto-Calibration Segmentation Method

NaoVi's segmentation algorithm is based on LUT (Look-Up Table) approximation as mentioned in the previous section. LUT segmentation is a quick an robust segmentation method but LUT has to be well defined previously. Robocup competition takes place over a week and the are soccer games in morning sessions and evening sessions.
Obviously, the illumination changes along the day even if the enclose is not an open-air place and it has artificial illumination.
To solve the illumination problem several methods were proposed but there is not unique solution or a close solution. Then, some teams have dedicate members that make periodically calibrations. It calibrations are useful to have a set of margins of field's objects (ball, lines, green field, goals, ...) and when a game takes place the best set of values are used. This technique requires an effort during one ore more days and it has to be done manually. In contradiction, we developed an auto-calibration method to generate LUT values. The auto-calibration method uses a K-means implementation. K-means is a wide used unsupervised clustering technique [60]. K-means aims divide image in K clusters or regions. Each region needs K seeds to search the closer seed, or centroid, to each pixel.
The seeds can be randomly chosen, randomly chosen from data observations or previously selected as our case. The centroids are set with a empirical centroids that were be obtained in previous calibration process.
The result of K-means method is K regions where the main color of regions is C_i. We can denote each pixel as an observation of 3 dimensions (x_1, x_2, x_3). One

image is 320 x 240 pixels then we have 76800 observations. K-means divides the space in sets $\{S_1, S_2, S_k\}$ so as to minimize within-cluster sum of squares

$$\min \sum_{i=1}^{k} \sum_{x_j \in S} \parallel x_j - \mu_i \parallel^2 \tag{7.7}$$

where μ_i is the mean of S_i.

Chapter 8

Object Detection

Image Segmentation has been explained in the previous chapter. Segmentation methods are low-level computer vision techniques. On the other hand, object detection can be defined as a middle-level or high-level computer vision technique. In the robot game field the number of objects that must be detected are defined by RoboCup rules. In Nao environment the expected objects are goals, other Nao robots, game field and the ball. Maybe the most important elements are the game ball and the goals.
All techniques to detect objects can be classified within the pattern recognition field. It is often used as a first step of other strategies as depth estimation or image understanding. No recognition is possible without previous knowledge. The set of objects can be divided into disjoints subsets from the classification point of view. The knowledge can be represented in many forms; some forms are:

1. Descriptors or features
2. Predicate logic
3. Production rules
4. Fuzzy logic

Object detection or object recognition is based on assign classes to objects. The number of classes are usually known in the *classification* phase. Also, there are algorithms that does not know the number of classes previously. In follow sections this knowledge and image input information will be used to classify possible object candidates as a first step to take a high level robot decision by a behavior algorithm.

8.1 Ball Detection

Game ball is determined in technical reports of RoboCup league. Usually, it is small and orange. RoboCup committee since several years want to avoid colour

limitations in game ball. Remember that the RoboCup main objective is to play with FIFA rules in few decades. Then, segmentation algorithm has to be focused in general balls detection. As in others Robocup leagues many researchers are developing new techniques to do ball detection in all conditions to consider the problem solved. But the great majority of ball detection algorithms are focused in colour characteristics and their detection are based on finding a blob with a fixed aspect. The objective of our work is to determine the sphere shape where a homogeneous orange is detected but focusing the problem to detect a none orange ball.
There are some approaches that were been presented by researchers about ball detection:

Classical Approach The ball is detected as an orange region [61] over game field. The detected region is subjected to an aspect analysis and if region has a pre-defined dimensions with a round aspect then algorithm labels the region as a ball. Usually, the next step is to include the region into a blob structure to manage it easily.

Pattern-matching method The proposed method once ball candidate is detected, the ball shape was found to be used as input of a pattern-matching method[57]. The method uses a grid-based segmentation and a special region growing. A eight directions (4 symmetrical) region growing is done for each pixel obtained in grid-segmentation process while ball shape is found. Then, these eight points are used to characterize the ball. The method concludes if it region is a ball or is not. Also, this method is proposed as a solution when a ball has a partial occlusion. If an occlusion happens during the play, a symmetrical ball region growing algorithm can reconstruct the ball if robot saw more than an half ball.

The two previous approaches are useful when ball colour is known. But what happens if it conditions change in future? The answer is that robot can not detect game ball. Fortunately game fields can not contain other objects, as in pictures 8.1, where some objects has similar aspect as balls, if color is not taken on account. Our ball detection approach is based on detect objects in game field and then classify the objects, as balls and no balls. The main difference between ball and other objects at field game is it circularity and density. Some approaches have been developed to detect balls for Robocup competition:

Scanlines Based, Scanlines segmentation give us a set of lines closely connected of an orange object, or a set of lines from a segmentation over all objects situated at field avoiding green field and white lines, as show figure 8.2 a. The extreme points of each scanline are used to define the object shape. Then shape circularity was search to define object as a ball or as an not classified object. Scanlines approach is fast and robust technique based on [57] algorithm when ball colour is known but it works worst when ball has an unknown colour.

Background Suppression, Robot vision perception uses geometral line as an upper limit to detect objects at image. Then ball must be under the

Figure 8.1: a) Objects situated at field b) Possible candidates.

line and it must be on the green field. Also, on the green line there are white field lines. Background suppression remove green field from image and all pixels upper the geometral line. In the next step all elements located at image are new candidates, then object features as aspect relation, density and circularity are computed to make a classification. Background Suppression works well when ball has not got white areas.

Field Soccer Game Suppression, removes green field and white lines from acquired image. White lines suppression can not be done directly because ball can be white. Then a erode and dilate are applied to the image at white channel. The objective is remove white lines with erode method and then the dilate method reconstruct the possible white balls. As seen in figure 8.2 b the method is applied with good results in all objects. The white objects are well segmented and it segmentation is used as in Background Suppression to detect the ball on the game field.

8.2 Goal Detection

Goal detection is maybe the most difficult object to be detected in game field. This is an object that can be seen by a robot partially. When robot has a partial perception of a goal this can be caused by an partial visual obstruction as happens when another robot is situated between robot camera and goal, or when robot is bad oriented to the goal direction, or when robot is to closer to the goal. Many techniques were tested to find the best solution that define absolutely well the goal with a small time response as possible.
Also the classical run length segmentation was used to have a time comparison base. As mentioned before grid-based segmentation is used as a segmentation method where two approaches were tested, the one directional *scanline* approach and the two directional classical grid-based approach, as it is shown at figure 8.3 shows. Conclusions were obtained with image segmentation comparison:

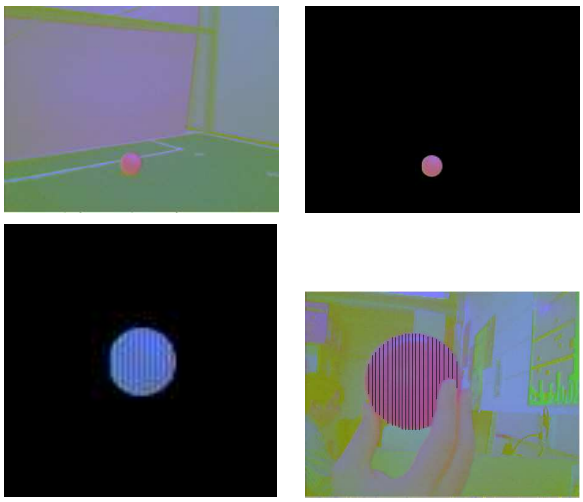

Figure 8.2: The three first images show the ball detection process and the last one shows the segmentation based on colour homogeneity.

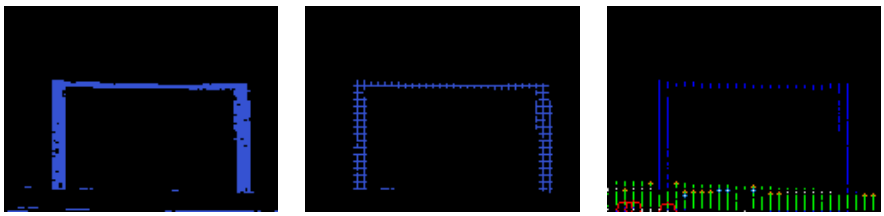

Figure 8.3: a) Classical Run Length Segmentation b) Classical Grid Based Segmentation b) Scanlines Segmentation.

1. Grid-based segmentation techniques give similar spacial information as classical run-length implementation.

2. Grid-based segmentation gives width information and the scanlines approach has not give that information. But scanline horizontal frequency gives a relative width information.

3. The best goal detection time is obtained when scanlines segmentation is used. The detection algorithm is still robust.

In another way, when figure 8.3 is observed accurately, it can be seen that images are almost in black colour. Usually, this means that the area that goal is filling in the images of game field is small. Sampling segmentation process reduces the pixel treatment but it can be reduced a little more. Using kinematic and inverse perspective projection, seen in previous chapters, top image areas can be avoided before the scan process. Inverse perspective projection can give the

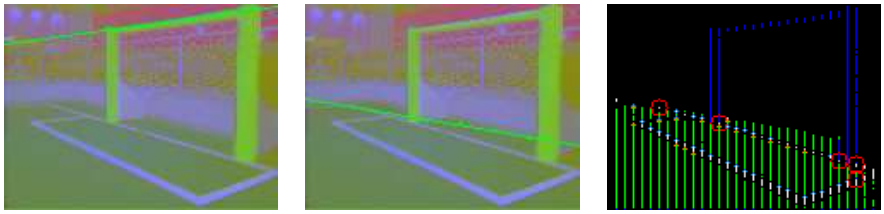

Figure 8.4: a) Horizon Line b) Geometral Line c) Oriented scanline segmentation

position where pixels are located over the goal in the 3D scene. Another solution can be applied to obtain the horizon line that is located above the goal. Goals colours are known and fixed by RoboCup rules, then a sampling technique with a very low frequency is used to obtain upper pixels that are forming part of goal. Once sampling lines are used to detect goal crossbar, the next natural step is to use that set of lines to detect horizon line too as show figure 8.4.

Table 8.1: Goal detection.

	Run-Length	Grid-based	scanlines
Time (ms)	16	6	4

When a sampling line detects a point that can be a part of goal crossbar or green field, this point is stored. When all sampling is done, crossbar points and green field detection points are computed to generate a line that delimit the top image that have to be processed and the horizon line. A linear regression is used to avoid isolated points that can generate a bad line slope, another technique like *RANSAC* can be applied too but it is more computational expensive.

The area, that was segmented by scanline method, is well defined and it covers pixel between geometral and horizontal line. Then scanline segmentation method is applied to goal detection. Using scanline detection over only one dimension reduces time response and it produces similar results as two directions sampling, as show figure 8.5 c. When goal detection scanline is done, another scaline segmentation algorithm is done down the horizon line to detect ball and field lines. Field lines detection are important to make robot localization. Robot localization is possible using goal information as shows [58].
Upper and down horizon line areas are segmented using scanlines method. Goal detection algorithm is based on a blob forming algorithm is used following the steps:

1. Scan all lines obtained in segmentation phase that can be a part of a goal.
2. All lines with aspect relation are merged forming a blob.
3. All blobs with closer relation and aspect relation are merged. Finally, blobs that are far to the rest are considered noise.

4. An object detection algorithm uses blobs obtained in previous phase to generate an estimated goal location in the 2D plane. the Object detection process uses blobs spacial information, aspect information, moment information and inertial blob information.

There was developed three object detection methods: fuzzy logic, automate and statistical pattern methods. The statistical pattern is based on minimal distance estimator. Blobs were processed and features were extracted from blobs. It method can get to us the estimated position of the goal and if the robot is watching the full goal o only one part. The automate method process sequentially the input blobs with aim to locate a set of blobs with goal geometrical similarity. The last one is fuzzy logic, it uses the blobs as an input data to be used in membership functions. This membership is used in the reasoning phase to get a 2D position. Fuzzy logic approach either the blob input information uses another input feature, when down horizon scanlines are treated a set of points were stored: blue-white and blue-green. The algorithm use feature points, that represents goals base, to help statistical decision to find goal using weighted values having more importance goal detection that features points. Once the two methods were tested empirically, fuzzy implementation gets best results (table 8.2). The automate is more sensible to scale factors and has more false goal detections.

Table 8.2: Goal detection.

	Automate	Statistical	Fuzzy
detection	65%	75%	80%

Goal detection algorithms make a decision using input blob generated after scanline segmentation. Once noise blobs are avoided, closer and similar blobs are merged and when detection was done the detection algorithm calculate the mass center using moment equation. At goal center decision five cases are possible:

1. Only on vertical poster is present at image. The goal position is situated at center of poster or on a featured point if statistical method is used.

2. Only a crossbar is present or not enough poster to be consider. The goal position is fixed at crossbar's center.

3. One vertical poster and one crossbar is present. This case is especially difficult to consider the goal center but using inertial center the problem is resolved.

4. All goal can be seen at image.

5. There is not a blob that can be classified as a goal part.

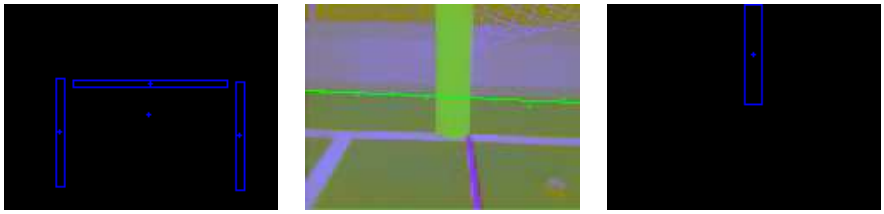

Figure 8.5: a) Blob Forming and inertial center b) Goal partial view c) Goal Position detection

The inertial center defines the goal position in 2D scene. Nao robot is working in a 3D space, to extend 2D goal detection to 3D goal detection some techniques can be used as proposed in [58]: *Thorus based 3D detection* and *Projective rays 3D detection*. The first method needs 1ms once posters and crossbar are well detected and the second method needs 2ms determine robot position in game field.

8.2.1 Fuzzy Logic

Fuzzy logic is capable of representing uncertain, non-exact and inaccurate knowledge. Fuzzy logic uses qualifiers as human way of expressing knowledge. Fuzzy logic can represent complex knowledge and even contradictory knowledge. We can define *fuzzy set* S in a fuzzy space X as a set of ordered pairs:

$$S = \{(x, \mu_S(x)) \mid x \in X\} \tag{8.1}$$

where $\mu_S(x)$ represents the grade of membership of x in S. The function $\mu_S(x)$ is commonly defined at range $[0, \ldots, 1]$. Zero representing no membership and one representing the full membership. In fuzzy reasoning, fuzzy membership function is usually required at least one element of the fuzzy set domain to have a membership value of one. There are some possible membership functions as sigmoid, linear, beta curve, trapezoidal curve and others. In fuzzy logic the membership are merged with other membership function using logic, the objective is generated a set of rules. Once rules are defined, they are used in the decision-making process. To design fuzzy logic system the steps are:

1. Determine the system data input, basic approaches and data output.
2. Define the fuzzy sets by decomposing each input variable of fuzzy system.
3. Convert previous knowledge into the if-then fuzzy rules.
4. Use a training set to determine the system's performance.

The fuzzy logic rules use maximum three input blobs u_1, u_2, u_3 previously selected. The rules used in fuzzy logic method are:

- if $found(u_1, u_2, u_3)$ and $is\_type(u_1, A)$ and $is\_type(u_2, B)$ and $is\_type(u_3, C)$ and $closer(u_1, u_2, u_3)$ then $found\_three\_posts$
- if $found(u_1, u_2, u_3)$ and $is\_type(u_1, A)$ and $is\_type(u_2, B)$ and $is\_type(u_3, C)$ and $closer(u_1, u_2, u_3)$ and $features\_closer(f1, \ldots, fn)$ then $found\_three\_posts\_confirmed$
- if $found(u_1, u_2)$ and $is\_type(u_1, A)$ and $is\_type(u_2, B)$ and $closer(u_1, u_2)$ then $found\_left\_corner$
- if $found(u_1, u_2)$ and $is\_type(u_1, A)$ and $is\_type(u_2, B)$ and $closer(u_1, u_2)$ and $features\_closer(f1, \ldots, fn)$ then $found\_left\_corner\_confirmed$
- if $found(u_1, u_2)$ and $is\_type(u_1, B)$ and $is\_type(u_2, C)$ and $closer(u_1, u_2)$ then $found\_right\_corner$
- if $found(u_1, u_2)$ and $is\_type(u_1, B)$ and $is\_type(u_2, C)$ and $closer(u_1, u_2)$ and $features\_closer(f1, \ldots, fn)$ then $found\_right\_corner\_confirmed$
- if $found(u_1$ and $((is\_type(u_1, A)$ or $(is_type(u_1, B)$ or $(is_type(u_1, C))$ then $found\_poster$
- if $found(u_1$ and $((is\_type(u_1, A)$ or $(is_type(u_1, C))$ and $features\_closer(f1, \ldots, fn)$ then $found\_poster\_confirmed$

Where A is the left vertical post, B the middle crossbar post and C the right vertical post. Features $f1, \ldots, fn$ are the blue-white and blue-green points stored in the segmentation process.

8.2.2 Automate

An automate was developed to test other kind of method. The automate was designed following these reasoning:

1. The Input data is know. This is a set of blobs with well know characteristic features.
2. The blob forming algorithm is know and the blobs are formed from left to right at image.
3. The scan frequency is know and we can determine is two blobs are connected.
4. The blobs are classified as verticals and horizontals where verticals blobs are the goal vertical posts and the horizontal is the goal crossbar.
5. Goal geometral similarity shape is searched in blobs' list to detect the ball.
6. Blobs with small areas or isolated from the others, where the great majority are closer one to the other, are removed.
7. When blobs candidates are selected, the center of mass is computed to determine a 2D point.
8. The 2D point is selected as goal target position.

8.2.3 Statistical Pattern

Statistical object description uses numerical named *features*. We can denote the features as $x_1, x_2, x_3, \ldots, x_n$. Then a pattern of this features can describe an object $x = (x'_1, x'_2, x'_3, \ldots, x'_n)$. We can define, *pattern space* X as the set of all possible patterns or feature space. If features are characteristic, similar objects will be close in feature space and objects of other class will be far. The classes will form clusters in the feature space and they will be separate by an hypersurfe. To determine if one object form part of one cluster or another, a discriminant function is used. The most common discriminant function is minimum distance principle. A classifier based on discrimination functions is a deterministic method, because one pattern always will be classified into the same class.

$$d(x) = \min_{i=1,\ldots,n} \mid x_i - y_i \mid \tag{8.2}$$

where y is defined the center object of the cluster or the representative. The Statistical Pattern method has three phases: formal description phase, training phase and classification phase. In the description phase the characteristics were selected from all characteristics of the input data. The second phase, called the training phase, the characteristics are tested and bad characteristics are removed or changed while the algorithm does not classified well the feature space. In the second phase the representative object y of each class was assigned. The classificator one more time is tested with know objects of each feature space to prove the classification.

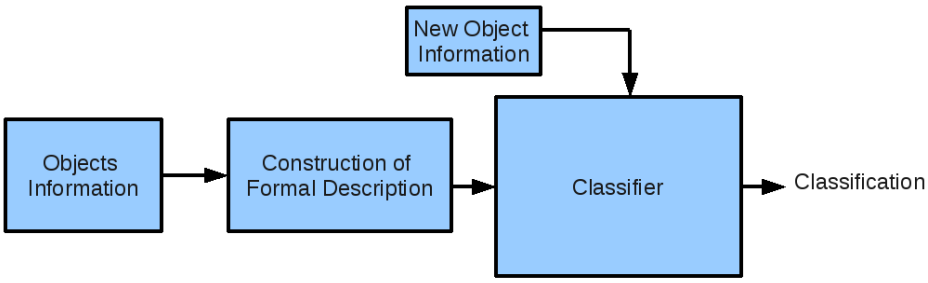

Figure 8.6: Classification Schema

The classifier input will be a set of characteristics extracted from the obtained blobs in the segmentation phase. The characteristics used in this object classificator method are:

- Center of mass
- Aspect relation

- Area
- Distance between blobs
- Vertical o Horizontal blob classification.

The classification method is based on mathematical expectation where an object determined by a characteristic features can be defined as a continuous probability distribution that describes data that clusters around a mean or average. Also, the error where the variables or features are bad selected can be measured by the Bayes criterion. If minimum distance principle is used in the classification process, the minimum error criterion can be used to quantify the loss incurred if a pattern x is incorrectly classified. If object x' is classified as C_1 and it is an object of class C_2

$$P(C_2|x) = \frac{P(x|C_2)P(C_2)}{P(x)} \tag{8.3}$$

where $P(x)$ is the mixture density. Then $P(C_2|x)$ is the error probability once object is classified. Also we can define the priori error probability as

$$P(C_1|x) = \max_{i=1,\ldots,R} P(C_i|x) \tag{8.4}$$

8.3 Nao and Field Lines Detection

The two most important elements at RoboCup's competition (ball and goal) are detected using sampling segmentation techniques combined with blob forming technique. The other two elements in Robocup's game field are Nao robots and field lines. First, Nao detection problem will revised in this chapter and then line detection will be focused. At Two-Legged Standard Platform League there are 6 Nao robots per game. Three of them are part of our team and the other 3 robot are rival team's robots. A well localized robot can share information with other team robots to determine they position. It information obtained with a shared perception algorithm that is a important field of research nowadays [62] can be used for a combined behavior in defense o attack state.
Also, three robots have unknown position. Nao detection aims is to develop some high-level algorithm to track rivals robots over the game field.
Nao robot is composed by part of two colours: white and other colour. It colour depends on team colour at game and it can be blue or red. In segmentation process all detected unknown areas that are not labeled as goal field, goals or lines are treated by Nao detection algorithm. Then an isolated blue or red regions at image are the input data to Nao detection method. The areas are detected in the scanlines segmentation phase. The first algorithm step deletes the isolated blobs of the selected regions. Then blobs are processed by a region growing algorithm. Region growing algorithm is not very expensive because blobs have not got big area because huge objects as goals are discarded, jet.

Figure 8.7: Nao detection process

Then usually a set of small closer areas are obtained as input data as figure 8.7 shows.
Then some characteristic data is computed from each blob. This characteristic data is also known as feature. The features selected in the Nao detection method are perimeter, centroids, center of mass, aspect ratio, circularity, density and principal vector data vector alignment.
As in figure 8.7 can be seen segmented regions are basically focused in the upper body robot part, then the upper body part is considered the *center of masses*. It significant 2D point indicator of Nao's position.
There are some artificial intelligence methods that can be used to detect Naos from a segmented regions. It classificator method must fulfill:

- It must differentiate between blobs formed with an image with a Nao on it and blobs formed with an image with a goal when regions are blue.
- It must differentiate between 4 principal Nao positions: front, back, left and right. The left and right one are difficult to differentiate between them but it will be interesting to get the discrimination for a high-level behavior algorithm.
- It must be a unsupervised classificator method.
- It must permit to extend the training set when more Nao pictures are available.
- It must be based on feature classification.

Given the above premises, the best method is clustering method. A clustering method is an assignment of a set of observations into subsets, called *clusters*. Clustering is a method of unsupervised learning. In this method, clusters and data can be easily incremented, but when data change, in the most cases, the clustering phase must be done one more time. There are three common clustering methods used nowadays: k-means, C-Means Fuzzy Clustering and artificial neural network.
K-means is a fast parametric clustering method when data dimensional is low.

Parametric methods attempt to minimize a cost function or an optimality criterion which associates a cost to each instance-cluster assignment. The goal of this kind of algorithm is to solve an optimization problem to satisfy the optimality criterion imposed by the model, which is often means minimizing the cost function. This type of method usually includes some assumptions about the underlying data structure. The K-means algorithm, probably the first one of the clustering algorithms proposed, is based on a very simple idea: Given a set of initial clusters, assign each point to one of them, then each cluster center is replaced by the mean point on the respective cluster. These two simple steps are repeated until convergence. A point is assigned to the cluster which is close in Euclidean distance of to the point, in N-dimensional space (8.5).

$$d_{C_1} = \sqrt{\sum_{k=1}^{n}(x_k - x_{C_1})^2} \tag{8.5}$$

where C_1 is the closer centroid to point x_k.
The k-means method has a drawback. It is very sensitive from the initial representative cluster point (figure 8.8). K-means clustering method is divided in two phases: training phase and classification phases.

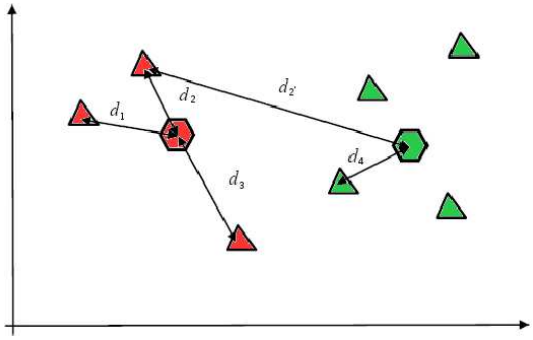

Figure 8.8: K-means process

In the our first developed phase the k-means was training with a set of 30 images where Nao was present. Once training phase was done, another set of 10 images was used to test the clustering method. The K-means method does a well classification in a 70 per cent of the images. Then, we can observe that the images but classified was not well segmented, too. Other error source was the intention of classify the observed robot when it is in left and right position. If only three classification types are used: front, back and lateral then the classification process correct answer increase to 90 per cent. It means that k-means method is very sensitive of the input data. It happens because blob areas are usually small at image and a bad segmentation causes a that blob areas change drastically.

Figure 8.9: Nao image acquired with webots simulator (160x120 pixels)

The images were taken from a external camera because we only have one Nao robot. There is a robot simulator called *webots* that can simulate a robot camera acquisition. Webots could be an image acquisition but there are two drawbacks. The first one is that camera simulator only works in QQVGA size and good results are obtained when algorithm works with bigger images. The second problem and maybe the worst one is that image quality is too bad to be used directly by the segmentation algorithm as shows figure 8.9.
Field lines detection is easier if it is compared with Nao detection. Hough transform was used to detect game field lines. Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing.
The classical Hough transform was concerned for the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses. The basic idea of the method can be seen from the simple problem of detecting a straight line in an image. A straight line is defined by two points $A = (x_1, y_1)$ and $B = (x_2, y_2)$. All straight lines going through the point A are given the expression $y_1 = kx_1 + q$ and $y_2 = kx_2 + q$. Then, there two common parameters k and q. We can interpret A and B in the parameter space $k - p$ and the only common point in both straight lines is the point that represents (k, p). This means that any straight line in the image is represented by a single point in the k,q space and any part of this straight line is transformed into the same point.
The Hough transform algorithm uses an array, called accumulator, to detect the existence of a line $y = kx + p$. The dimension of the accumulator is equal to the number of unknown parameters of the Hough transform problem. For example, the linear Hough transform problem has two unknown parameters: m and b. The two dimensions of the accumulator array would correspond to quantized values for m and b. For each pixel and its neighborhood, the Hough transform algorithm determines if there is enough evidence of an edge at that pixel. If so, it will calculate the parameters of that line, and then look for the accumulator's bin that the parameters fall into, and increase the value of that bin. By finding the bins with the highest values, typically by looking for local maxim in the accumulator space, the most likely lines can be extracted, and

their (approximate) geometric definitions read off. The simplest way of finding these peaks is by applying some form of threshold, but different techniques may yield better results in different circumstances - determining which lines are found as well as how many. Since the lines returned do not contain any length information, it is often next necessary to find which parts of the image match up with which lines. Moreover, due to imperfection errors in the edge detection step, there will usually be errors in the accumulator space, which may make it non-trivial to find the appropriate peaks, and thus the appropriate lines. Once all field lines present at image are detected by Hough transformation algorithm, the line intersections were detected using equation 8.6

$$P_x = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{8.6}$$

$$P_y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

In future work, the lines intersections will be used as a landmarks. These landmarks will be useful for localization algorithms and visual odometry. Localization methods will help us to make a more dynamic play and robots can make a collaborative game behaviors. Also, goal detection and crossing lines points can be used as a landmarks to generate a visual odometry technique. Visual odometry [13] usually is used combined with classical odometry and visual odometry correct false motion estimation. Nao robot has not got any type of odometry. If motion can be modeled using visual odometry and error of the method can be estimated, it can be used to correct Nao trajectory.

Chapter 9

Depth Estimation

All proposed middle-level detection algorithms are oriented to 2D detection, but robot c-space is a 3D space. Then, all detection points estimated in Nao detection, ball detection and goal detection are going to be transformed to a three-dimensional coordinate. Usually three-dimensional information in computer vision is associated with stereo-vision problem where two camera situated in fixed known positions acquired two overlapped images where correlated points were found using a comparison algorithm as used in SURF, SHIFT, SUSAN corner or other simple pixels value correlation.
Nao robot has two cameras where images are not overlapped, then classical stereo-vision methods can not be applied directly. Other solutions has been proposed in recently years where monocular vision systems are used to simulate stereo-vision conditions. Its works are based on acquiring two images moving camera, lineally or rotationally, a fixed known variation that comply the overlapping condition. When a pixel position is estimated from a two dimensional image in a three dimensional space, the vertical and the horizontal position of the pixel in the three dimensional space can be estimated. Even the straight line that cross the pixel from the visual point (VP) can be defined. Then only one image is not enough to estimate distance in three dimensional space. But some geometric assumptions can be used to do distance estimation when objects are on the floor ($z = 0$).
Objects as goals and lines can be static for a long time when robot is not moving. When robot that is acquiring images, is in scene-scanning-state. The objects, as ball and other robots, can change they position in a short elapsed time. Then the acquiring image phase must be, as fast as possible, to remove estimation error caused by a change in the scene. If meanwhile the two images are acquired the ball position change then the stereo-vision system can not determine the correct position. Also, if ball disappear in the second image the depth estimation position can not be determined using stereo-vision techniques. In that cases the distance will be estimated using other techniques.
NaoVi framework implements the following approaches:

Figure 9.1: Some balls with same size and different color situated over the field

1. Objects width and height size is used to estimate object distance form the camera. Object size technique is a classical depth estimation approach commonly used to detect the distance to the ball. The ball area increase when it is closer to the robot, then ball area is small when it is far. An example can be seen at figure 1 where farthest ball is situated at 240 cm and the distance between balls is 40 cm. In our study the farthest ball is situated at 3 meters far and the pictures were taken every 10 cm. Also, another methods can be used as an indexed table by area. It table return the distance to the ball with a accurate measure. A previous calibration of the table is previously necessary. Some pictures of the ball are taken at different distances and then the table is generated. The distance estimation error is the half distance of the distance used between the captured images, it error is 5 cm in our table because the distance is 10 cm.

2. Objects situated at floor height can be estimated easily using a trigonometric properties (figure 9.2). Objects at game field are situated on floor level, then we can consider that objects are situated at $z = 0$ in robot frame. Robot kinematic and inverse perspective projection are known. An image's point can be situated at three dimensional robot's frame. The robot's pose is known and robot's height h can be determined. Then the distance D and the angle α from robot to the object can be determined with the simply geometric equation 9.1.

$$D = \frac{h}{tan(\alpha)} \tag{9.1}$$

   This approach was developed using kinematic information gave by NaoMo framework.

3. Stereo-vision techniques using a pure rotation between two images.

Distance Estimation Trigonometrical Properties
A rectangled triangle has one of its interior angles measuring 90° and all interiors angle sum is 180°.

$$180^\circ = 90^\circ + \alpha + \rho$$

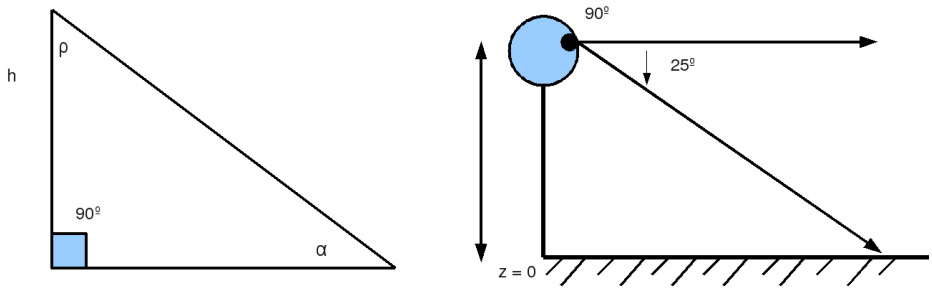

Figure 9.2: (a)Left: Trigonometric principles. (b)Right: Nao Vision Schema.

Nao kinematic model is know then we can calculate the angle formed from camera and the floor. The camera frame sets $\rho = 0°$ when camera is looking forward. In the trigonometric triangle 9.2 b, it is set to $\rho = 90°$ then we can calculate the α angle as in the example:

$$180° = 90° + \alpha + (90° - 25°)$$

$$180° = 90° + \alpha + 65°$$

$$90° = \alpha + 65°$$

$$\alpha = 25°$$

Or just we can use the ρ angle as the α angle value when we consider that the $x - axis$ is situated forward camera point as in the schema.

Stereo-vision techniques using a pure rotation
An homography is an invertible transformation from a projective plane to a projective plane that maps straight lines to straight lines. Any two images of the same planar surface in space are related by homography when we are using a pinhole camera model. In cases such camera motion, rotation and translation, between images. If the camera motion between two images is pure rotation, with no translation, then the two images are related by a homography too.
Given two cameras A and B looking at point P_i in a two-dimensional plane. Passing the projections of P_i from ${}^bP_i$ in B and ${}^aP_i$ in A. One camera position can be defined as a rotation and translation of the other, then

$${}^aP_i = k_a \cdot H_{ba} \cdot k_b^{-1} \cdot {}^bP_i \tag{9.2}$$

where H_{ba} is the homography matrix, also K_a and K_b are the cameras' intrinsic parameters.

$$H_{ba} = R - \frac{t_n^T}{d} \tag{9.3}$$

t denote the translation vector and R the rotation matrix where B is rotated in relation to A. Two other parameters must be defined: n the formal vector of the

Figure 9.3: a) Image took with a rotation angle fixed at 0° b) Image took with a rotation angle fixed at 25°

plane and d the distance to the plane respectively. In out case, the projection from one camera to another is determined only by a rotation then

$$H_{ba} = R \tag{9.4}$$

As happen when Nao vision uses only one camera a calibration process was done. Nao stereo-vision based on pure rotation matrix uses only one camera, then the intrinsic camera matrix are the same and it must be equivalent to the obtained in chapter 6. Stereo-vision method needs a overlapping images to detect common points. The common point to be determined in three-dimensional space are the points detected by the objects detection algorithm and then a correlation method is not necessary. Although, to guarantee a overlapping over the to images the maximum camera rotation angle must be fixed. At figure (9.3) can be observed that halt of the first image is not present in the second image. Then maximum rotation angle was fixed to 15° to guarantee more than a half overlapped images. A empirical test was done, the objective was determine the distance from the pattern used in the calibration phase using the stereo-vision method. Figure 9.4 shows the results obtained where distance from the pattern was a 80 centimeters and stereo-vision method value was 85 centimeters, then estimation error was fixed to 5 centimeters. The cameras's frame was moved at figure to observe the small rotation over the $z - axis$ and remove occlusions at graphic.

Usually in stereo-vision systems the distance estimation to a determined point is calculated using the equation

$$\frac{T - (x_{left} - x_{right})}{Z - f} = \frac{T}{Z} \Longrightarrow Z = \frac{fT}{x_{left} - x_{right}} \tag{9.5}$$

where T is the translation from one observation to another and Z is the distance to the object and f is the focal length.

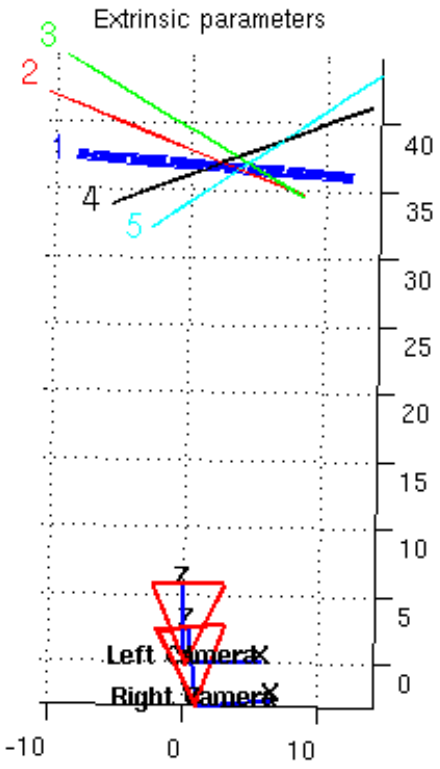

Figure 9.4: Depth estimation using stereo-vision method

Chapter 10

Visual Sonar

Visual Sonar is a technique proposed and partially developed by Scott Lenser (2006) [17]. It is a quite novel obstacle detection that is not usually implemented when classical obstacle detection methods works. Also when robot has sonar sensors, usually, vision techniques are used on a second plane in closer distance. We proposed a real-time response technique developed in NaoVi framework that is based of sensing image in a controlled scenario as game field over a biped robot. Similar solutions are used by AIBO robot in RoboCup environment too [4]. In previous works, the sonar was oriented using only head actuators values. In our new solution, the entire body kinematic model, that was developed in NaoMo, is used to project rays that Visual Sonar technique need to detect obstacles. Last kinematic approach does not contemplate when robot has a stable pose with a only support point, which can be possible in a biped robot.
Visual Sonar is an approach that uses oriented sampling image analysis to obtain a real-time response with maximum information extraction. It is usually based on detecting some specific feature at image and then try to project that point to the robot 3D space.
Our new approach uses the projection from 3D space to 2D space in order to generate a set of optical rays from bottom camera frame to depth image. This approach is proposed initially for soccer RoboCup environment, but a more generalized solution is under study in structured environments. The ray projection scan the image while an object is not detected. Then, an instant depth estimation is obtained.
Also, Visual Sonar is considered a good solution to replace the Nao detection algorithm explained in the Object detection chapter if the planning algorithm only uses obstacle information when it is present in robot trajectory. The most important characteristic of visual sonar is that it can be also used when the background is known as a obstacle detection and when there are unknown obstacles in scene, as Lenser cited being the principal method contribution. This means that visual sonar will be used in other situations and in a future RoboCup field conditions, even if it changes.

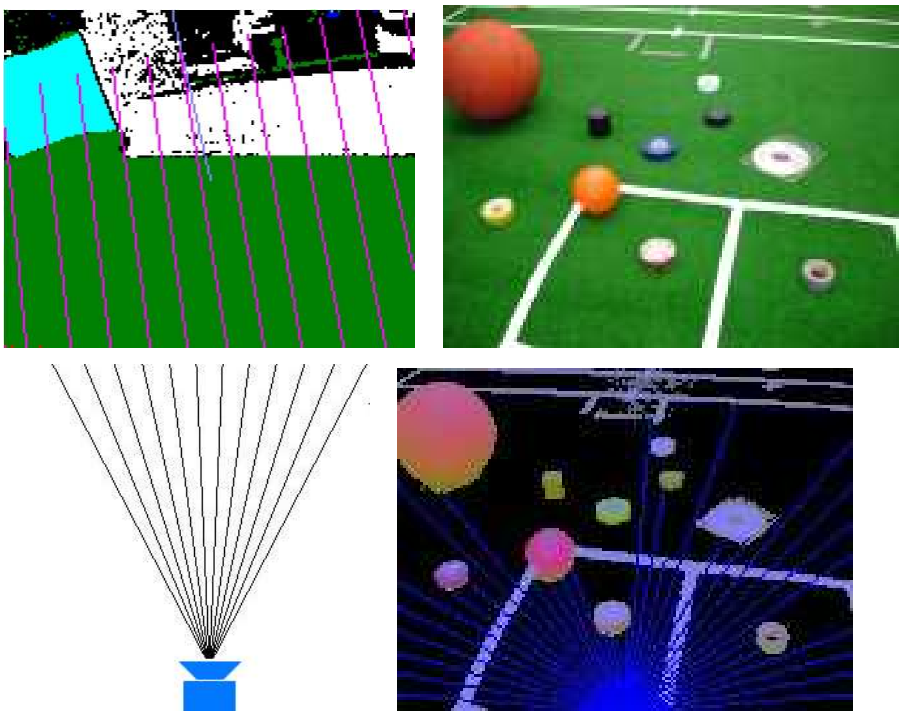

Figure 10.1: (a) Classical Visual Sonar, (b) Original image, (c) Projective rays from the camera to depth scene and (d) Projective rays over the image

The steps to generate the visual sonar map are:

- Calculate the plane parallel to the robot $y - axis$. It plane must be orthonormal to the $x - z - axis$. The plane is known as geometral plane.
- Remove the background from input image.
- Generate perspective projection or inverse perspective projection from each point at 2D space.
- Evaluate for each pixel if it is an obstacle or it is not.
- Generate environment map.

The Visual sonar map is generate meanwhile the robot is moving over the field (figure 10.2).

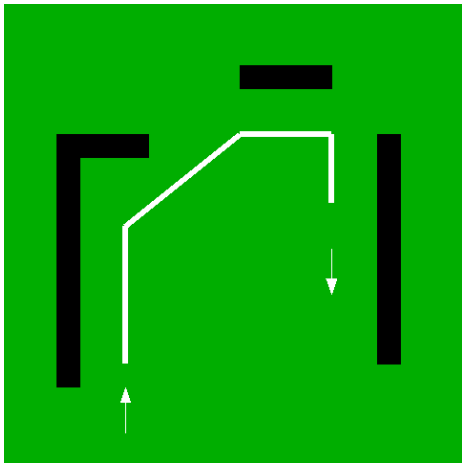

Figure 10.2: Visual Sonar Map

Chapter 11

Behavior and Challenge

All algorithms developed to work over Nao robot architecture, such as path planning algorithm(appendix C) have to be a light implementation solution and they must have a real-time response. Some path planning solutions can be applied to real-time problems if the available computational power is high. In our case, Nao has a CPU (500Mhz) with a reduced computational power. Also a long term artificial algorithm is not developed yet and it will be developed in a closer future.
NaoMo and NaoVi can be considered as a *reaction based* frameworks that make a response to a given stimulus. Otherwise, path planning algorithm will give to the robot the capacity to perform a long term specific task. Nao main task is to play soccer in RoboCup competition. Nao robot's field is modeled with the potential field approach. Potential field approach is the more reactive technique presented previously. Objects detected with goal detection method, Nao detection and visual sonar are then mapped into a potential field approach. There are two kinds of objects at soccer field: repulsive and attractive. The attractive are ball and goals. The repulsive ones are other Nao robots at field and the field limits. The RoboCup rules explicitly told that robot can not have any type of contact with other robot to avoid damages. And robots that are going to crash with another robots are penalized. Then a virtual force field is applied over the robot and repulsive objects. This is a fast method to choose a path, this is known as *Virtual Path* (VP). A Virtual field can be determined form each field point x taken on account all objects at field with the equation

$$VF(x) = \frac{F_i \cdot (x - o_i)}{|x - o_i|^2} \tag{11.1}$$

where F_i is the force vector weight and o_i is the position to the object. The weighted value can be positive in attractive objects and negative for the repulsive objects. At figure 11.1 can be seen a RoboCup field with potential fields modeled using objects situations. Then A^* is used then to find the best path that robot must follow avoiding the possible robot contact. Also A_* algorithm must generate a locomotion to the goal when robot own the ball (figure 11.1).

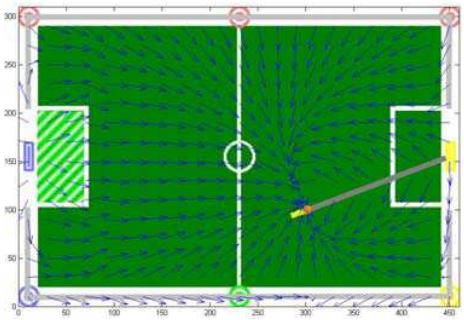

Figure 11.1: Potential Fields methods used to generate a trajectory path to the goal.

Behavior framework named *Nao player* has three principal behaviors that are controlled with an automate state process:

Obstacle avoiding, This is the basic robot behavior that consist of avoiding contact with other robots. This control has the higher priority in Nao Player controller.

Objects localization, Once objects are located using robot frame its objects are mapped into a local potential field approach to be consulted by other behavior models.

Ball attraction and kicking oriented, As known in a soccer game the team that scored more times win the game. At current time, the principal objective is not to win competition because it is an ambitious objective in the first year.

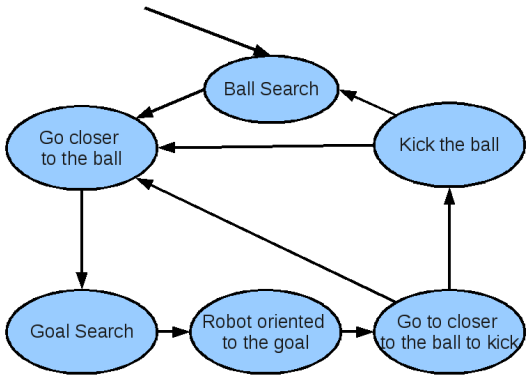

Figure 11.2: Robot's behavior automate.

The robot behavior is modeled as an automate, see 11.2. The first state search the ball in the entire field. Meanwhile the ball is searched the field lines and robots' position at field are stored. And potential field map are refresh. The field is modeled with a grid where each cell is 10 cm x 10 cm length. Then the robot is oriented to the ball and motion began to reach a closer position to the ball. When robot is closer that 10 cm the robot search the goal. Once goal is detected and stored then it changes the active camera to the bottom one. Then the robot reach a closer position oriented to the goal and do a kick oriented to the goal.
Meanwhile the robot is moving along the field it can detect a other robot closer to it. Then if a robot is detected closer the robot using the sonar then the robot stops and it try to surround the obstacle. In robot soccer, flexible and robust strategy design is fundamental. Several works present approaches [63] have been done to find the best strategy method. The most used strategies are behavior-based, decision trees ans fuzzy logic systems. A high level behavior has been developed to be used in four-league with different roles for the players. In the literature some roles are common: *goalkeeper*, *attacker*, *support attacker* and *defender*. Roles can be statically assigned or dynamically assigned with goalkeeper exception. RoboCup has two scored competitions:

1. Soccer games
2. Technical challenges

The technical challenges [64] are actions or behaviors that RoboCup SPL Technical Committee thinks that they will necessary to be developed in each competition year to fix a development rhythm. This year three challenges were proposed:

1. The "Any Ball" Challenge
2. The Passing Challenge
3. The Localization with Obstacle Avoidance Challenge

The aim of this challenge is to test whether the robots can detect balls other than the orange ones on the field, and manipulate them as precise as they can manipulate orange balls.
At Graz RoboCup 2009 we presented a solution to detect homogeneous any colour balls to participate in the first challenge. The proposed approach looks for homogeneous regions at field. The first step was remove the green field from images. Then, a erosion and dilation binary operation was done over the while regions. For each region, white and other colour except green, a feature discriminant function was used. The features selected were: area, aspect relation, circularity and density. The geometral line is used to remove noise from the image. The figure 11.3 shows the results obtained with any ball detection algorithm.
Then when the balls are detected,the closer ball is chosen as the ball to be kicked first.

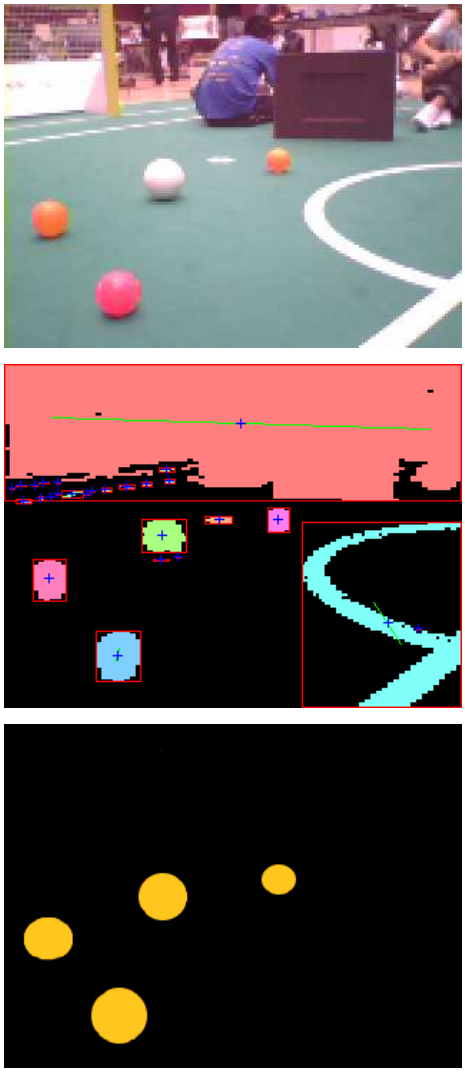

Figure 11.3: (a) Original Image, (b) ExtractedBlobs, (c) Balls detected

Chapter 12

Conclusions and Future Work

Nao robot is going to be the only allowed robot in RoboCup Standard Platform League. RoboCup Graz 2009 competition was really the first year that participating teams played with some minimum player behaviors. The last year in Suzhou competition the teams went to compete in bad conditions because Nao robot manufacturer served they units few months ago. In some cases, teams were not capable to make walk the robots during the game because they fall all time. This year in German Open competition some teams went with low-level and middle-level algorithms that were partially working as expected. Then, few months later during RoboCup Graz 2009 competition (celebrated from June 29th to July 5th), the teams improved their behaviors to face competition under good conditions. Intelligent Robotics and Computer Vision Group (RIVI) at Rovira i Virgili Univerity has integrated into the Spanish interuniversity team known as *TeamChaos*. *TeamChaos* was born as a Swedish team but now is fully integrated by Spanish universities.
Focusing then our research effort to make a quick start-up in Nao robot architecture, some problems were solved and others are going to be solved in a small elapse time. But a realistic criterion shows us that Nao robot behavior is still far to be as good as the achieved with Sony's Aibo robot few years ago by *TeamChaos*. In this work context, many problems are faced:

Nao development environment, A new robot architecture always implies a new challenge where ambitious goals are fixed. NaoQi software architecture was studied and tested to find the best characteristics and the worst ones. Once a complete study was done, NaoVi and NaoMo design was decided. DCM fulfilled our needs without the time limitations that NaoQi Library has. Then a new lighter framework was developed to perform locomotion actions (NaoMo) and computer vision algorithms (NaoVi).

Kinematic analysis, Kinematic analysis is the first step when a new locomotion is going to be designed. Some classical approaches were studied as screw method or D-H method. Forward and Inverse robot kinematic analises were done to transform joint space to a tridimensional space and vice versa. Kinematic transformation results were compared with obtained by Aldebaran methods and time response was compared too. NaoMo framework transformation was almost equal and the variation can be caused by the calculate process, but NaoMo was a 10% faster than NaoQi Library with the same calls.

Locomotion new methods, New locomotion methods were developed to facilitate developing behaviors and a more stable walking was design based of ZMP. Locomotion methods associated with NaoVi vision framework, as scanning method, were also created with aim to call repetitive actions by a high level methods to provide future developers with an abstraction layer.

Camera model study, Nao's principal perception system are two cameras situated forehead. Images acquired with Nao's cameras are computed to get an analysis of the scenario. Detection algorithms can locate objects at image space but it is completely unsatisfactory because robot are moving over an 3D scenario. Then, camera internal parameters called *intrinsic* values are needed. Pinhole model, the most common camera model, was studied and applied to Nao's cameras to get they internal parameters. Two utilities were used to get results that can be compared. The utilities use two different mathematical approaches to obtain pinhole camera model but both returned equivalent intrinsic matrix. Another empirical prove was done with aim to supervise the complete calibration method.

Stereo-vision study, Nowadays, stereo-vision system is common in robotics applications, specially in mobile robots. Navigation algorithm needs a depth estimation to avoid obstacles or detect goals. Stereo-vision techniques are usually used in static scenarios. RoboCup competition is a dynamic one but stereo-vision algorithms can be used with aim to get a robust localization, when lines and goals are considered beacons. A stereo-vision approach was tested in Nao robot with aim to obtain three-dimensional information.

Low-level computer vision algorithms, Image processing first stage commonly is based of doing a segmentation over the image. Image segmentation time can be reduced to dedicate more time to other algorithms as ball position detection. Two principal solutions are proposed and developed in the segmentation stage: LUT colour classifier approach combined with scanlines segmentation. A classical segmentation algorithm needs 16 ms at least to segment a whole image meanwhile scanlines segmentation can be performed with only 3ms.

High-level computer vision algorithms, Once image is computed using the multi-channel segmentation process, the extracted information can

be used to detect the game ball, goals and other Naos at game field. Some methods are proposed to determine the correct position at field. And a new visual sonar approach is proposed using the robot's inverse kinematic model combined with camera's inverse projection model as an object detection.

Behaviors and challenge, Some low-level behaviors were implemented to perform RoboCup soccer games during RoboCup 2009. Also a ball detection algorithm was developed to perform the technical challenge competition.

Publication, The article [58] will be published in Workshop in Physical Agents 2009 to be celebrated in Caceres.

All the proposed objectives in this Master Thesis were achieved, also some other approaches were developed at Nao robot, such as visual sonar and object detection methods, which were modified to be used at the Technical Challenge competition. Some Master Thesis results were published in an article [58] and others will be published in the future. This work can be seen as the first needed steps that must be done to perform another high-level computer vision and robotics methods. In the following years our study and work will be faced on localization and shared perception methods with aim to obtain an autonomous biped robot that can be successful in the RoboCup competition. One of the RoboCup aims is to be a platform to develop new robotics and computer vision techniques that can be exported to other robotic fields. Even some solutions presented here perhaps can be used in a closer future in domestic robots.

Bibliography

[1] *"RoboCup Standard Platform League (Nao) Rule Book"*, 2008 - 2009

[2] M. Veloso and M. Phillips, *"GTCMUnited08: Calibrated Motion, Modular Vision, and Accurate Behaviors*, *Technical Report, Carnegie Mellon University*, 2008

[3] G. Sen and D. Bailey, *"Discrete YUV Look-up Tables for Fast Colour Segmentation for Robotic Applications"*, *Canadian Conference on Electrical and Computer Engineering*, pp 963-968, 2008

[4] J. Fasola, *"Fast Goal Navigation Obstacle Avoidance Using a Dynamic Local Visual Model"*, *IEEE Latin American Robotics Symposium* pp 1-6, 2005

[5] R. S. Dahiya, M. Valle and G. Metta, *"Tactile Sensing Arrays for Humanoid Robots"*, *Research in Microelectronics and Electronics Conference*, pp 201-204, 2007

[6] M.A.Garcia, *"Hierarchical Clustering of 3D Objects and its Application to Minimum Distance Computation"*, *IEEE Int. Conf. on Robotics and Automation*, pp 5287-5292, 2004

[7] D. Gouaillier, V. Hugel and others, *"The Nao humanoid: a Combination of Performance an Affordability"*, *Computing Research Repository*, 2008

[8] Aldebaran Robotics, *"Nao Robot Reference Manual"*, *Internal Report*, 2009.

[9] *Open Embedded Linux Distribution"http://www.openembedded.org/"*

[10] W. Skarbek and A. Koschan, *"Colour Image Segmentation - A Survey "*, *Berlin University Technical Report*, 1994

[11] A. Mikdad, A. Hussain and others, *"Implementation of Inverse Perspective Mapping Algorithm for the Development of an Atomatic Lane Tracking System"*, *IEEE Tencon 2004*,Vol 1 pp 207-210, 2004

[12] D. Bullock and J. Zelek, *"Real-Time Tracking for Visual Interface Applications in Cluttered and Occluding Situations"*, *Image Vision Computing*, Vol 22, 2004

[13] J. Campbell, F. Sukthankar and others, *"A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision"*, *IEEE International Conference on Robotics and Atomation*, pp 3421-3427, 2005

[14] A. Pretto, E. Menegatti an others, *"A Visual Odometry Framework Robust to Motion Blur"*, *IEEE International Conference on Robotics and Atomation*, Accepted to publication, 2009

[15] S. K. Nayar, Y. Nakagawa, *"Shape from Focus"*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*,Vol 16 pp 824 - 831, 1994

[16] S. Yi and N. Ahuja, *"An Omnidirectional Stero Vision System Using a Single Camera"*, *IEEE International Conference on Pattern Recognition*, 2006

[17] S. Lenser and M. Veloso, *"Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision"*, *International Conference on Intelligent Robots and Systems*, 2006 *"Real-Time Tracking for Visual Interface Applications in Cluttered and Occluding Situations"*, *Image Vision Computing*, Vol 22, 2004

[18] J. Strom, G. Slavov and E. Chown, *"Omnidirectional Walking using ZMP and Preview Control for the NAO Humanoid Robot"*, *RoboCup Symposium 2009*, 2009

[19] T. Zarraonandia, J. de Lope and D. Maravall, *"Definition of Postural Schemes for Humanoid Robots"*, *Current Topics in Artificial Intelligence*, Vol 3040 pp 241-250, 2004

[20] A. Cherubini, F. Giannone and others, *"Policiy Gradient Learning for a Fumanoid Robot"*, *Robotics and Autonomous Systems*, Vol 57, 808-818, 2009

[21] K. Yin, k. Loken and M.V. Panne *"SIMBICON: Simple Biped Locomotion Control"*, *ACM International Conference on Computer Graphics and Interactive Techniques*, 2007

[22] R. Featherstone and D. Orin *"Robot Dynamics: Equations and Algorithms"*, *Proc. IEEE Int. Conference on Robotics and Automations*, pp 826-834, 2000

[23] M. Spont, S. Hutchinson and M. Vidyasagar, *"Robot Modeling and Control"*, *Ed. Wiley*, 2006

[24] M. Cui-hua, F. Xun, L. Cheng-rong and Z. Zhong-hui *"Kinematic Analysis Based on Screw Theory of a Humanoid Robot"*, *Journal of China University of Mining and Technology*, Vol 17 pp 49-52, 2007

[25] R. Manseur *"Robot Modeling and Kinematics"*, *Ed- Thomson*, 2006

[26] H. Yussof, M. Ohka, M. Yamano and Y. Nasu *"Navigation Strategy by Contact Sensing Interaction for a Biped Humanoid Robot"*, *International Journal of Advanced Robotic System*, Vol 5 pp 151-160, 2008

[27] G. Tan and P. Zhang, *"A new Method for computing key Kynamic Parameters of Dynamic Walking of Biped Robots"*, *Int. Conf. on Intelligent Processing Systems*, pp 28-31, 1997

[28] Honda electronic Resources *http://world.honda.com/ASIMO/*

[29] Omnivision, *"Advanced Information Preliminary Datasheet OV7670 CMOS VGA Camera Chip with OmniPixel Technology"*, *Internal DataSheet*, 2006.

[30] MicroJet Technology Co., *"Product Data Sheet Model 16A7"*, *Internal DataSheet*, 2008.

[31] A. Ryberg, A. Chistiansson, E. Eriksson adn Bengt Lennartson, *"A new Camera Model for Higher Accuracy Pose Calculations"*, *"IEEE Proceedings of International Symposium on Industrial Electronics"*, pp 2798-2802, 2006

[32] K. Kinoshita and M. Lindenbaum, *"Camera Model Selection Based on Geometric AIC"*, *IEEE Conference on Computer Vision and Pattern Recognition, 2000. Proceedings*, Vol 2 pp 514-519, 2000

[33] N. Sanchez and A. Perez, *"Analisis aplicado de metodos de calibracion de camaras para usos fotogrametricos"*, *Congreso Nacional de Topografia y Cartografia*, 2004

[34] Z. Zhang *"A Flexible New Technique for Camera Calibration"*, *Microsoft Research*, 2008

[35] H. Zhuang, Z.S. Roth and R. Sudhakar, *"Simultaneous Robot/World and Tool/Flange Calibration by Solving Homogeneous Transformation Equations of the Form AX = YB"*, *IEEE Transactions on Robotics and Automation*, Vol 10, 1994

[36] K. H. Strobl and G. Hirzinger, *"Optimal Hand-Eye Calibration‘"*, *Int. Conf. on Intelligent Robots and Systems*, pp 4647-4653, 2006

[37] J. Weng, P. Cohen, and M. Herniou, *"Camera calibration with distortion models and accuracy evaluation"*, *IEEE on Transactions on Pattern Analysis and Machine Intelligence*, 1992

[38] K. H. Strobl and G. Hirzinger, *"More Accurate Camera and Hand-Eye Calibrations with Unknown Grid Pattern Dimensions*, *IEEE International Conference on Robotics and Automation*,2008

[39] Z. Zhang, *"Flexible Camera Calibration by Viewing a Plane from Unknown Orientations"*, *Internationval Conference on Computer Vision*, pp 666-672, 1999

[40] C. Wang, *"Extrinsic Calibration of a Vision Sensor Mounted on a Robot"*, *IEEE Transactions on Robotics and Automation*, Vol 8 pp 161-175, 1992

[41] H. Zhuang and z. Qu, *"A New identification Jacobian Robotic Hand-Eye Calibration"*, *IEEE Transactions on Systems, Man and Cybernetics*, Vol 24 pp 1184-1287, 1994

[42] M. Jungel, J. Hoffmann and M. Lotzsch, *"A Real-Time Auto-Adjusting Vision System for Robotic Soccer"*, *LNCS RoboCup 2003 Springer*, Vol 2030 pp 124-255, 2004

[43] G. Liwn, K. Yamamoto and K. Kato, *"A Proposal of Generalized Horizon View Camera"*, *IAPR Conference on Machine Vision Application*, pp 61-64, 2007

[44] L. Lucchese and S.K. Mitra, *"Color Image Segmentation: A State-of-the-Art Survey"*, *INSA-A Image Processing on Vision Pattern Recognition*, pp 207-221, 2001

[45] U. Lydia, C.S Teh and G.W Ng, *"A Comparison of RGB and HSI Color Segmentation in Real-time Video Images: A Preliminary Study on Road Sign Detection"*, *International Symposium on Information Technology*, 2008

[46] A. R. Palma-Amestoy, P.A. Guerrero and others, *"Context-dependent Color Segmentation for Aibo Robots"*, *IEEE Latin American Robotics Symposium*, pp 128-136, 2006

[47] T. Rofer and M. Jungel, *"Fast and Robust Edge-based Localization in the Sony Four-legged Robot League"*, *LNIA: RoboCup2003*, pp 262-273, 2003

[48] C. Gonner, M. Rous and k. Kraiss, *"Real-time Adaptavie Colour Segmentation for the RoboCup Middle Size League"*, *LNCS: Robobocup2004*, pp 402-409, 2004

[49] D. Herrero and H. Martinez, *"Fast and Robust Recognition of Field Line Intersections"*, *IEEE Latin American Robotics Symposium*, pp 115-119, 2006

[50] L. Iocchi, *"Robust Color Segmentation Through Adaptive Color distribution Transformation"*, *LNCS: RoboCup2007*, pp 287-295, 2007

[51] T. Rofer, *"Region-Based Segmentation with Ambigous Color Classes and 2-D Motion Compensation"*, *LNCS: RoboCup2008*, pp 369-376, 2008

[52] E. Littmann and H. Ritter, *"Adaptive Color Segmentation - A Comparison of neural and Statistical Methods"*, *IEEE Transactions on Neural Networks*, Vol 8 pp 175-185, 1997

[53] Z. Lin, J. Jin and H. Talbot, *"Unseeded Region Growing for 3D Image Segmentation"*, *Pan-Sydney Workshop on visual Information Processing*, Vol 2 pp 232-237, 2001

[54] C. Hung and W. Lie, *"Region Growing Based on Extended Gradient Vector Flow Field Model for Multiple Objects Segmentation"*, *International Conference on Image Processing*, Vol 3 pp 74-77, 2001

[55] D. Comaniciu and P. Meer, *"Mean Shift: A Robust Approach Toward Feature Space Anaysis"*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 24 pp 603-619, 2002

[56] R. Alvarez, E. Millan and others, *"Multilevel Seed Region Growth Segmentation"*, *LNAI: MICAI 2005*, pp 359 - 368, 2005

[57] J. Bach and M. Jungel, *"Using pattern matching on a flexible, horizon-aligned grid for robotic vision"*, *LNCS Multilevel Seed Growing Segmentation Springer*, 2005

[58] J. Canas, D. Puig, T. Gonzalez and others, *"Visual Goal Detection for the RoboCup Standard Platform League"*, *Workshop of Physical Agents 2009*, 2009

[59] M. Valdes, D. Herrero and M. Martinez, *"Toward Automatic Camera Calibration Under Changing Lighting Conditions Through Fuzzy Rules"*, *Lecture Notes in Computer Science IWINAC*, pp 179-388, 2007

[60] J. B. MacQueen, *"Some Methods for classification and Analysis of Multivariate Observations"*, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Vol 1 pp 281-297, 1967

[61] M. J. Quinlan, C. L. Murch, T. G. Moore and others, *"The 2008 NUbots Team Report"*, *RoboCup Symposium 2008*, 2008

[62] C.E. Agero, A.L. Gonzalez, F. Martin and V. Matellan, *"Distributed Perception for a group of legged robots"*, *IEEE Internationa Symposium on Intelligent Signal Processing*, 2007

[63] P.A Vallejos J. Ruiz-del-Solar and A. Duvost, *"Cooperative Strategy using Dynamic Role Assignment and Potential Path Planning"*, *Journal: IEEE Latin American Robotics Symposium*, 2004

[64] RoboCup SPL Technical Committee, *"Technical Challenges for the RoboCup 2009 Standard Platform League Competition"*, 2009

[65] Y.K. Hwang, and N. Ahuja *"Gross motion planning - A survey"*, *ACM computing Survey*, Vol 24 pp 219-299, 1992.

[66] Colin G. Johnson, *"Robot motion planning - A survey"*, *Department of Mathmatics Report*, 1995.

[67] A. Sud, E. Andersen, S. Curtis, M.C. Lin and D. Manocha, *"Real-Time Path Planning in Dynamic Virtual Environments Using Multiagent Navigation Graphs"*, *IEEE Trans. Vis. Comput. Graph.*, Vol 14 pp 526-538, 2008

[68] R. A. Finkel and J. L. Bentley, *"Quad trees a data structure for retrieval on composite keys"*, *Acta Informatica*, Vol 4 Number 1 pp 1-9, 1974

[69] J. Baltes and J.Anderson, *"Flexible Binary Space Partitioning for Robotic Rescue"*, *Intelligent Robots and Systems*,Vol 4 pp 3144 - 314, 2003

[70] Y.K. Hwang and P.C. Chen, *"A Heuristic and Complete Planner for the Classical Movers Problem"*, *IEEE International Conference on Robotics and Automation*, Vol 1 pp 729-736, 1995

[71] J.L. Baxter E.K. Burke, J.M. Garibaldi and M. Norman, *"Multi-Robot Search and Rescue: A potential Field Based Approach"*, *Studies in Computational Intelligence (SCI)*, Vol 76 pp 916, 2008

[72] Y.Koren and J.Borestein, *"Potential field medhots and their inherent limitations for mobile robot navigation"*, *IEEE Conference on Robotics and Automation* pp 1398-1404, 1991

[73] J. Pearl, *"Heuristics: Intelligent Search Strategies for Computer Problem Solving"*, *Addison-Wesley*, 1984

[74] N. Sariff and N. Buniyamin, *"An Overview of Autonomous Mobile Robot Path Planning Algorithms"*, *IEEE 4th Student Conference on Rearch and Development*, pp 183-188, 2006

[75] E. W. Djkistra, *"A Note on Two Problems in Connexion with Graphs"*, *Numerische Mathematik* pp 269-271, 1959

List of Figures

Appendix A

Appendix - Nao General characteristics

The General Characteristics provided by the Aldebaran Robotics Nao manufacturer[8] are:

- Body characteristics
  - Height: 58 cm
  - Weight: 4.3 Kg
  - Body type: Technical plastic
- Energy
  - Charger: AC 90-230 volts/DC 24 volts adapter, CCCV output 25.2 V, 2 A max
  - Battery capacity: 45 min. autonomy
- Degrees of freedom
  - Head: 2
  - Arms: 5 in each arm
  - Pelvis: 1
  - Leg: 5 in each leg
  - Hands: 2 in each hand(0 in RoboCup edition)
- Audio
  - 2 loudspeakers. Diameter=36mm. Impedance=8 ohms. Sp level=87dB/w +/-3 dB. Freq range= up to 20 kHz. Input=2W
  - 4 microphones (2 in RoboCup edition). Sensitivity: -40 +/- 3 dB. Frequency range: 20Hz-20kHz. Signal/noise ratio= 58dBA.

- Network access
  - Wi-Fi (IEE 802.11g)
  - Ethernet connection
- Sensors
  - 32 x Hall effect sensors
  - 1 x gyrometer 2 axis
  - 1 x accelerometer 3 axis
  - 2 x bumpers located at the tip of each foot. These are simple ON/OFF switches. There is no difference between a push on the left or right foot.
  - Ultrasound: 2 emitters, 2 receivers. Frequency: 40kHz. Sensitivity: -86dB. Resolution: 9mm. Detection range: 0.2 - 1.2 m. Effective cone: 60.
  - 2 x I/R. (Academics only). Wavelength=940 nm. Emission angle=+/-60. Power=8 mW/sr
  - Camera: VGA 640x480, 30 fps. Focus range: 30 cm - infinity. Vision field: 58 on the diagonal
  - Capacitive sensor
- Motherboard
  - x86 AMD GEODE 500MHz CPU
  - 256 MB SDRAM / 1 GB flash memory
- Embedded software
  - Operating system: Embedded Linux (32 bit x86 ELF) using custom OpenEmbedded based distribution
  - Programming languages: C, C++, Python, Urbi

Appendix B

Appendix - Denavit-Hatenberg modeling technique

Denavit-Hatenberg (D-H) method is a robot kinematic modeling [23] technique first proposed by Denavit and Hartenberg. D-H robot modeling technique has become the standard in mathematical representation of robotic structures and is widely used in the robotics literature.
To analyze the motion of robot manipulators, reference frames are attached to each link starting from frame F_0, attached to the fixed link, all the way to frame F_n assuming the robot has n joints.
The D-H model is obtained by describing each link frame along the robotic chain with respect to the preceding link frame. The D-H modeling technique reduces the number of parameters necessary to represent F_i from F_{i-1} to only four parameters. Other robot modeling techniques like *RPY* (*Roll-Pitch-Yaw*) or *ZYZ Euler Angles* need minimum 6 non trivial parameters to represent the same frame. The D-H parameters denoted as a_i, d_i, α_i and ρ_i:

$\mathbf{a}_i$ is the length of the common perpendicular to axes z_{i-1} and z. a_i length is also known as *link length*.

$\mathbf{d}_i$ is the algebraic distance along axis z_{i-1} to the point where the common perpendicular to axis z_i is located. In the bibliography these parameters is also called the *link offset*.

$\boldsymbol{\alpha}_i$ is the angle (*link angle*) around x_i that vector z_i makes with vector z_{i-1}.

$\boldsymbol{\theta}_i$ is the angle around z_i that the common perpendicular makes with vector x_{i-1}. The angle around z_i is called *link twist*.

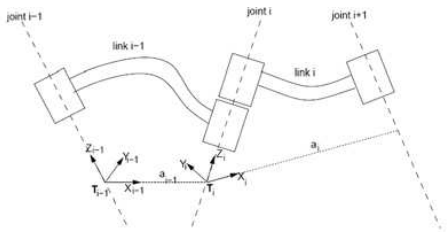

Figure B.1: Revolute Joints

B.1 Link Frame

The first step to develop the mathematical model of a robot is to assign the link frame. D-H model describe a process to follow formed by some rules and conventions. The rules are:

The z-vector is z_i**,** z_i of a link frame F_i is always on a joint axis.

The x-vector x_i**,** x_i of link frame F_i lies along the common perpendicular to axes z_{i-1} and z_i, and is oriented from z_{i-1} to z_i.

The origin , The link frame F_i origin is located at the intersection of the common perpendicular to axes z_{i-1} and z_i and joint axis z_i.

The direction , of the vector z_i is always chosen so that the resulting twist angle α_i is positive with the smallest possible magnitude.

B.2 Frame Location

The four DH parameters fully determine the location of a link frame. If we want determine some frame B from another frame A, which frame A is situated closer of base frame than frame B, by the four D-H parameters d, a, α and θ (as figure B.2). Starting from frame A, frame B can be found as the result of the sequence of four frame transformations described here.

1. From the origin of frame A, move the reference frame a distance d on the $\mathbf{z}_A$ axis. Note that d is algebraic and can be positive or negative depending on the position of the common perpendicular with respect to the origin of frame A. This step is a translation along $\mathbf{z}$ of a distance d.

2. The direction of vector $\mathbf{x}_B$ is determined by rotation vector $\mathbf{x}_A$ by an angle θ around $\mathbf{z}_A$. This is a rotation around $\mathbf{z}$ of angle θ.

3. Move the frame a distance a in the direction of vector $\mathbf{x}_B$. The position reached is the origin of frame B. In this step success a translation along $\mathbf{x}$ of a distance a.

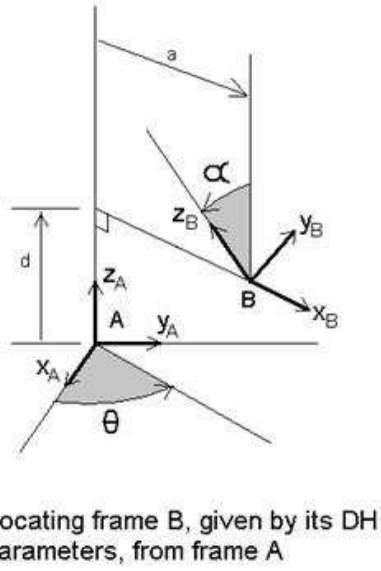

Figure B.2: Frame Location from DH parameters

4. Unit vector $\mathbf{z}_B$ is obtained from vector $\mathbf{z}_A$ by a rotation around $\mathbf{x}_B$ by an angle α. The last step is a rotation around $\mathbf{x}$ of angle α.

Appendix C

Appendix - Path Planning

Find-path problem, commonly known as Path Planning[65], is a well known in robotics and it plays an important role in navigation of autonomous mobile robots. The principal Path Planning objective is to arrive to a determined point *P1* to another point *P2* usually situated closer in the robot world space. It is important that path selected be the optimal without colliding with obstacles in the workspace area. These points could be in 2D or in 3D space and the complexity of the space on which the robot will navigate determine the overtaken strategy to generate a path in navigation. Path planning typically refers to the design of only geometric (or kinematic) specifications of the positions and orientations of robots. Then trajectory planning includes the design of the linear and angular velocities as well. In conclusion we can consider path planning as a subset of trajectory planning, where is not negligible in dynamic robots behavior. Trajectory planning is most important in biped robot walking motion because it stability must be keep in account.
Path Planning research of autonomous mobile robot has attracted attention since the 1970's. Recently in the past several years, researchers increased the special interest in that field induced by that robots are now applied in various application. Mobile robots application areas could be exploration, surveillance, industrial, armamentistic or domestic use. The wide areas like industrial and armamentistic areas is exploration area. In that area the robot complexity path planning generation in the navigation process change radically determined by the navigation media. That media could be aerial (AUV), aquatic media or terrain media. In our study case we are using a biped robot that navigate in terrain media.
In terrain media path planning algorithm are usually based on robot working space representation such as the Voronoi diagrams, regular occupancy grids, quad-trees and vertex graphs. That representation structures play an important role and greatly influence the computational complexity in real-time navigation process. There is a wide number of Path Planning algorithms that main objective is to manipulate the data structures to reduce the computational effort and in that way increase the decision flexibility of the path planning algorithm to

react in a dynamic environment as we are talking about. Some algorithms use that follow artificial intelligence methods [65, 66]:

1. Graph search algorithm
2. A* search algorithm or modified A* search algorithm
3. Genetic algorithm
4. Neural Networks
5. Heuristic methods for special situations
6. Stochastic and fuzzy logic methods

These above methods are not necessarily mutually exclusive, and a combination of them is often used in developing a motion planner. In sections above these methods are overview with they principal advantages and limitations.

Methods above can be classified by:

Completeness, the method can be *exact* if always find the best results or *heuristic* if not.Exact classification methods are computationally expensive in contrast witch heuristic ones. The Second one have the aim of find a solution in the shortest time. In many applications is considered better find a poor solution in a breve time interval, than a good o perfect solution in a long time interval.

Scope, we can talk about *local* or *global* depending on the input information used to obtain a decision and the classification method efficiency or limitations.

There are few principal different class planning:

Motion planning can be static or dynamic, it depends on whether the information change (is updated) or the environment is fixed along the robot navigation. In this case we have a dynamic environment.

Obstacles can be stationary in that we are talking about a time invariant problem. In our study case we have a movable-object problem because the other robots and the game ball change they situation every few seconds.

Also a there are a principal difference to make planning for one or multiple robot. Nowadays we are using a single path planning situation and multirobot or coordinate behavior will overtaken in future works.

C.1 Representation Motion Planning Approaches

An effective method for large problems is the generation of objects according to some principles and investigation of their properties. Object generation principles are called operators. Operators and the initial state define an object tree

called *search space*.
From the point of view of problem description we may consider some important requirements:

1. The way in which each space search object is represented.
2. Operator that we can apply to the space objects. This is the control strategies.

Numerous representation methods have been developed for motion planning. Some of them focused to resolve a particular planning problem and that methods have a limited applicability. In this section the mean motion planning approaches under study can be distributed in groups as:

1. Skeleton
2. Cell decomposition
3. Potential Field

C.1.1 Skeleton

The Skeleton method idea is to fit or reduce the objects in the workspace, usually that objects are in 2D space, to a one-dimensional lines. The method is also called the retraction, highway or roadmap approach. Ones the objects situated in the 2D becomes a lines in a 1D dimension the problem is transformed to a graph-searching problem. There are few well-known methods as Voronoi diagram[67] or visibility graph used for 2D space objects. Voronoi diagram method consists in a Eucliean space partition. If we consider S a 2D space with a set of n points P where each point *p(i)* is distributed in that space. For each point x in the S space can be determined the distance to each point *p(i)*. All point x that is closer to *p(i)* than another *p(j)* is part of called *Voronoi cell* or *Dirichlet cell V(i)*. Then, *V(i)* is formed by all points x that are closer to *p(i)* than another *p(j)* point. Also there are a set of points y that have the same distance to two or more points *p(i)*. This set of points are forming a line called *Voronoi tessellation*. In the special case that the space have only two points *p1* and *p2* forming then only two *Voronoi cells* we are talking about a *hyperplane* that separate the space in two affine subspaces. Is commonly extended the *Manhattan* distance or *Mahalanobis* distance used to reemplace the *Euclidean* distance but then is not guaranteed the *true tessellation*. This mean that is not guaranteed the S partition in two affine spaces.
Then path planning will be fixed to pass closer *Voronoi tessellation* point and far as possible of the *Voronoi cell* center. Voronoi diagrams method is commonly combined with potential field methods for local planning. This combination avoid damages at mobile robot when objects represented by *p(i)* have different aspect.

C.1.2 Cell decomposition

This approach has a simple principle. In that case all S Space is decomposed into a set of simple cells at the first step. Then we have to identify the cells than contain the start and goal points. The follow algorithm steps try to connect the this two points with free cells sequence. To find one way to arrive from point star to goal point is possible to follow two strategies:

Object-dependent Decomposition, in this case the boundaries of the object are used to create the cells boundaries. When the cells are created then an union operation merge them into one block. Of course the union of the free cells create the free space. This kind of decomposition method is not commonly used because this algorithm requires a computation effort. Specially when the polygon decomposed into a convex polygon, in that case the algorithm is classified as NP-hard.

Object-independent Decomposition, there are two main methods name Binary space partitioning method and quadtree method.

Quadtree Decompositions and Binary space partitioning

In this section will be explain the two most used object-independent decomposition methods. The first method called *Quadtree*[68] consist in a tree data structure when each internal node has up four child. Quadtree is not a really a tree because the Quadtree is not information independent structure, and to increase computation response, some nodes that have homogeneous data into it will not generate a new generation child. Quadtree can be classified as a *tries* using the space partition result. Quadtrees are most often used to partition a two dimensional space by recusively subdividing it into four quadrants or regions. The regions usually may be square or rectangular, but it can take another structural shape else. Some authors classify quadtrees by the data that type represents:

1. Region Quadtree
2. Edge Quadtree
3. Point Quadtree

Region Quadtree decompose the space in four equal quadrants then each quadrant contains information about one space region. As his node parent, each child node has four size equal quadrants or nodes. When an image pixel can obtain a value *0* or *1* and n depth node represent a $2^n x 2^n$ image space. Then the root node represents a block where not all pixel below share the same *0* or *1* value. When homogeneity value is share into all region the split algorithm stops, toward the split algorithm continue generating quadrant regions while regions are great than 1 pixel. Edge Quadtree is a data structural change to store lines. This approach was created with aim to reduce data volume stored when occupancy by obstacles and free spaces was great. The principal problem

of that approximation is the unbalanced trees. There is a Quadtree variant when we are working in 3D called *Octree*.
The second object-independent decomposition method called *Binary space partitioning* (BSP) is a method for recursively subdividing a space into convex sets by hyperplanes. Also the principal structure is a tree data known as *BSP tree*. Binary space partitioning is a generic process of recusively dividing a scene into two until the partitioning satisfies some requisites. The specific method of division depends on it final purpose; in [69] detection objective the partition algorithm can stop when region is simple enough for a simple heuristic or iterative treatment. BSP trees can be used with a great partition space number and we can create a four partition space as Quadtrees but Quadtrees are most useful in subdividing 2D spaces.

C.1.3 Potential Field

Potential Field approach is a completely different approach compare with Quadtrees and BSP. The Potential field is used not only to avoid obstacles locally but also to design a globally optimal path in the sense of path length for the classical mover's problem[70].
The classical mover's problem is the problem of finding a path for a rigid object between the start and the stop/target points while avoiding obstacles in the configuration Space, also known as C-Space. This problem solution is usually given as sequence of positions and orientations of the object. Mover's problem is more critical if the robot is autonomous and the problem have to be well defined to avoid damages. Although, that solution must be quite easy to be implemented and quite efficient to response in hard or soft real time. Some times mover's solution in 2D space with 3 DoF (Degrees of Freedom) is a brute search algorithm based on previous observations that minimize the computation analysis and can be quite fast because this is a reduced dimension scenario.
In our configuration space we have a 2D space and path planning can be resolved using a heuristic algorithm; a computationally efficient potential function is defined in terms of the boundary equations of polyhedral obstacles. The morphological structure of the free space is represented by a valleys of minimum potential (MPV)[70] with a local minimum value in target point. Whereas, obstacles are represented as a high value positions. Points inside an obstacle have greatest than points in the boundary. Then one point field is expressed as a sum of free space and obstacles values, that resultant force can denoted by R. R determines the subsequent direction and speed of travel. A path search algorithm must find the local minimum to obtain a local solution, also if we have completely global information or global information with known areas we can obtain a global solution. In RoboCup scenario we have an approximated case of global information when robot perception algorithm are share with other team robots [62, 71]. In the same line, shared perception or local perception combined with Potential Field Path Planning are used in [63] to switch dynamic roles between robots in a soccer game in RoboCup environment. The critical cases in this solutions will be when two or more obstacles are closer on to each

other and there are not a clear path to take. The easy case is when we have a wide free in the C-Space and we not have a possible collision. Then the optimal path is obtained by minimizing a weighted sum of the potential on the robot. We have not to forget that our path planning algorithm must be flexible as possible because when we have a more than one feasible path to take in a local solution the robot can need move go back to the latest position. It can be necessary if that local solution is not a global solution or bad solution when the robot perception change. We have to consider that we have a dynamic scenario and this local solution can be the worst solution in few seconds.

Of course potential field methods have path planning problems and it are studied by some researchers as [72]. Some methods based on potential field methods as *VFF* (*Virtual Force Field*) try to resolve that inherit problems:

- Trap Situations due to local minim (cyclic behavior).
- No passage between closely spaced obstacles.
- Oscillations in the presence of obstacles.
- Oscillation in narrow passages.

Trap Situations Due to Local Minim

A *trap-situation* may occur when the robot runs into a dead end. Trap can be resolved by heuristic or global recovery. Global perception research as [71] try to reduce this kind of situations. In RoboCup Soccer game that situations can not be possible because we not have dead end obstacles in the game field in this situations will not be take into account. Otherwise the solution of that problem is useful using *global path planner* method and can be worst if the use a approach based on an heuristic method.

No passage between closely spaced obstacles

With Potential Field Methods a repulsive forces of two obstacles *O1* and *O2* can be expressed by *F1* and *F2* respectively. When robot must take the path between that two objects, the sum of the repulsive forces *F1* and *F2* depending the force magnitude can take the robot away from the closer path (between two obstacles).

Oscillations in the presence of obstacles

One of the most significant limitation of Potential Field Method is the tendency of make unstable motion when we have a in-line set of obstacles. This is a great challenge to abort if we want to use the outside line fields to keep our robot game player the most time as possible in the field and avoid that the robot can go out driver by a bad perception as environment noise.

Oscillation in narrow passages

As the last PFM limitation, we can consider field lines as a corridor where robot must navigate in it. Also, this *corridor* in our case is not narrow to cause an oscillation in robot travel as we can experiment with PFM in other situations. Then we will not abort this kind of problems.

C.2 Search Methods

Given a description of our configuration space overtaken in chapter C.1 then we must choose an search algorithm to apply into the field representation used. As we mentioned above, we have multiple searching methods approaches to apply in this and other situations. Either, an exhaustive graph or tree search over the *search space* is possible if we consider not important the time needed for find a solution or the computational cost. In this case, always we will obtain the best solution if that solution exist. Else, the principal search method objective if to obtain the best or the better solution to the problem described some of them are commented in the following sections.

C.2.1 Graph search algorithm

Is well known that in graph theory the shortest path problem. It consist of finding a path between two nodes, that nodes are called *vertices*. In practice the nodes of the graph represent states and the algorithm search a path between two wished states.
To find that way we must follow the graph nodes avoiding a cycle to minimize the nodes visited. The two nodes are connected by weighted branches. The objective of the algorithm is to find a path between the start and stop nodes such the sum of the weights nodes are minim. Usually that branches are particular operators, then the branch sequence in minimum path is a set of operators to apply ordered in a concrete problem which have known solution. Most of the theoretical results concerning search strategies apply to trees in special types of graphs. On the other hand, many of the practice problems cannot modeled by cycle-less graph structures.
Given and weighted graph with a set of vertices V we can formalize a function f that give a real value of a path p, $f : G \rightarrow R$. This path p has a start vertex at v and final vertex at v'.

$$R = \sum_{p \in P} f(p) \tag{C.1}$$

The most important graph search algorithm that try to find a solution to the shortest path problem are overtaken in the follow subsections.

Strategies of this class differ in the amount of knowledge they use. We shall take an overview of strategies which do not use any information like depth-first or breadth-first and heuristic ones like best-first and A*. Graph oriented

strategies will be also over-viewed like Dijkstra's algorithm or Bellman-Ford algorithm.

Depth First and Breadth First Strategies

Depth First Strategy is a classic state-space search strategy. This method and some others are considered by some set of researchers a brute force strategies . The name emphasizes the order in which nodes are examined. A single path is investigated as long as its last element is not fount to be a goal or a terminal node. Search is always carried out starting from the last, most recently examined node which still has some un-inspected edges. The main problem of use *Depth First* is that it can be ineffective, even though the goal node is at a finite depth. Depth first search is usually supplemented by a depth limit control. So when the depth of a node exceeds a certain limit or a node satisfies some goal property, the depth first search backtracks. This implementation is also known as *Backtracking strategy.*
Breadth First Strategy is a brute force strategies but as opposed to the previously described strategies. This one examine the nodes in the levels of the same node depth assigns a higher priority to nodes of a lower depth. As the first algorithm the strategy will find a solution, if it exists. The main shortcomings of this algorithm compared with *Depth First Strategy is the memory wasted storing all nodes.*

Best-First Algorithm

Best-First is an heuristic search algorithm to allow information when making a decision about the next node to be examined. The heuristic function called *f* over node (or vertex v) *f(v)* is used as a criteria by the *Best-First.* The node *v'* with smaller heuristic function value *(v')* will be examined first. In [73] Judea Pearl described *Best-First* search as estimating the promise of node n by a "*heuristic evaluation function f(n) which, in general, may depend on the description of n, the description of the goal, the information gathered by the search up to that point, and most important, on any extra knowledge about the problem domain*". Over the years many modified *Best-First strategies* (BFS) are implemented and used in wide fields applications but the most important implementation is the A* algorithm [74].

A* Search Algorithm or Modified A* Search Algorithm

The standard search algorithm for the shortest path problem in a graph is A* algorithm. As mentioned above, the A* algorithm can be considered as the best first search algorithm that combines the advantages of greedy searches using a fitness function and uniform-cost. We can formalize as:

$$f(n) = h(n) + g(n) \tag{C.2}$$

where *f(n)* represents the heuristic function value for the node *n*, *h(n)* the fitness function of the remaining cost to get from node *n* to the goal node and *g(n)* the accumulated cost. During the search, the A* maintains two lists of nodes. The open list contains the nodes that have to be considered next and the closed list contains the nodes already visited. The algorithm itself consists of expanding the one node from the open list whose fitness function is minimal. Expanding a node means putting i into the closed list and inserts the neighbors into the open list and evaluating the fitness function. As happens before, the algorithm stops when the goal of node is expand. Is important the choice of a good heuristic algorithm in order to achieve both quality and efficiency of a search. If estimation is the real cost the shortest path is guaranteed found. But an under estimation can lead to an expansion of too many nodes. Either, when over estimation success the A* algorithm tends to expand nodes in the direct path to the goal before tying other nodes. The A* algorithm it has been widely used because it finds the minimum cost path and it will determine the existence of free path. The classical algorithm have a time consuming problem because it does not have a heuristic information to handle until it reach the goal. This will cause a lot of node being produced to find a short path and to avoid obstacles which will eventually make it run slower. In our case, that limitation is critical because we have a hardware platform with a small memory and slowest CPU, see appendix A for more details.

Dijkstra's Algorithm

Edsger Dijkstra[75] considered n nodes, all or some of them are connected by a branch. It branches are weighted and that weight is known. The problem was restricted and he considered that where at last one path between any two nodes. Then the problem was divided in two cases:

1. Construct the tree of minimum total length between the n nodes.
2. Find the path of minimum total length between two given nodes v_k and v_j.

If we denote the *initial node* as v and the *final* or *goal node* as v' Djkistra algorithm have the follow steps:

1. Assign to every node a distance value. Set to infinity for all nodes except the *initial node* v to zero.
2. Mark all nodes as unvisited. Set initial node as current, called v_k.
3. For *current node*, consider all its unvisited neighbors and calculate their distance (from the initial node). For example, if current node v_k has distance of 9, and an edge connecting it with another node v_j is 3, the distance to v_j through v_k will be 9+3=12. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.

4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.

5. Set the unvisited node with the smallest distance (from the initial node) as the next *current node* v_k and continue from step 3

The Dijkstra's algorithm published in 1959 [75] gives an exact method of finding the shortest distance connecting two points, but principal problem is that the algorithm does not work if any of the edges have a negative value. Because the author consider that you can have an edge of negative length. If we are talking about distances then obviously Dijkstra was in true because this is impossible, but if we consider a situation when edge (or branch) is a cost operation. Then the algorithm is not useful if any operation value is negative. Even, of course we can normalize the operation cost from 0 to a value K if it is possible. Other factor to take on account is that the algorithm minimize the operation cost between two nodes but the not he number of operations that solution have.

Another algorithm were proposed and applied in a large range problems. Some proposed methods are focused on real-time response as Genetic Algorithm, Neural Networks and some Fuzzy Logic implementations.