Abstract

RoboCup is an annual international robotic soccer competition. One of its most popular divisions is the Four-Legged League, whereby each team consists of four Sony AIBO robots and researchers focus solely on software development over a standard robotic platform. To compete successfully each robot needs to address a variety of problems: visual object recognition, legged locomotion, localization, and team coordination. This thesis focuses on the problem of visual object recognition and suggests a uniform approach for recognising the key objects in the RoboCup 2007 field: the two goals, the two beacons (colored landmarks), and the ball. The proposed Image Processor module processes the color-segmented camera image and delivers the type, as well as an estimate of the distance and the angle with respect to the robot, of each recognized object in the current field of view. Our approach is based on a number of procedures which are used uniformly for all three types of objects: horizontal and vertical scanning of the entire image, identification of large colored areas through a finite state machine, clustering of colored areas through histograms, formation of a bounding box indicating possible presence of an object, and customised filtering for removing implausible indications. Our approach is compared against the image processors of German Team 2004 and SPQR-Legged 2006 and is shown to be equally good or better in many cases. The proposed approach has been used successfully by Team Kouretes of the Technical University of Crete during various RoboCup events.

Περίληψη

«Οπτική Αναγνώριση Αντικειμένων για το Πρωτάθλημα Τετραπόδων του Διαγωνισμού Ρομποτικού Ποδοσφαίρου RoboCup 2007»

Το RoboCup είναι ένας ετήσιος διεθνής διαγωνισμός ρομποτικού ποδοσφαίρου. Μία από τις δημοφιλέστερες κατηγορίες του RoboCup είναι το πρωτάθλημα τετραπόδων, όπου κάθε ομάδα αποτελείται από τέσσερα ρομπότ Sony AIBO και οι ερευνητές επικεντρώνονται αποκλειστικά στην ανάπτυξη λογισμικού για τη συγκεκριμένη ρομποτική πλατφόρμα. Προκειμένου να διαγωνισθεί επιτυχώς, κάθε ρομπότ καλείται να επιλύσει μια σειρά προβλημάτων: οπτική αναγνώριση αντικειμένων, μετακίνηση με βηματισμό, εντοπισμός θέσης στο γήπεδο και ομαδική συντονισμένη συμπεριφορά. Η παρούσα διπλωματική εργασία επικεντρώνεται στο πρόβλημα της οπτικής αναγνώρισης αντικειμένων και προτείνει μια ενοποιημένη προσέγγιση για την αναγνώριση των βασικών αντικειμένων στο γήπεδο του RoboCup 2007: τα δύο τέρματα, τα δύο beacons (χρωματιστά κολωνάκια) και η μπάλα. Το μοντέλο που προτείνεται για επεξεργασία εικόνας επεξεργάζεται την χρωματικά τμηματοποιημένη εικόνα της κάμερας και δίνει ως έξοδο τον τύπο, καθώς και μία εκτίμηση της απόστασης και της γωνίας σε σχέση με το ρομπότ, για κάθε αντικείμενο που αναγνωρίζει μέσα στο οπτικό πεδίο τη δεδομένη χρονική στιγμή. Η προσέγγισή μας βασίζεται σε μια σειρά από διεργασίες οι οποίες χρησιμοποιούνται σχεδόν αυτούσιες και για τους τρεις τύπους αντικειμένων: οριζόντια και κάθετη σάρωση ολόκληρης της εικόνας, προσδιορισμός μεγάλων χρωματικών περιοχών μέσω ενός πεπερασμένου αυτομάτου, συγκέντρωση των χρωματικών περιοχών με τη βοήθεια ιστογραμμάτων, σχηματισμός πλαισίου που οριοθετεί και υποδηλώνει ενδεχόμενη παρουσία αντικειμένου και προσαρμοσμένο φιλτράρισμα προς αποφυγή αδόκιμων ενδείξεων αντικειμένων. Η προσέγγισή μας συγκρίνεται με την επεξεργασία εικόνας που χρησιμοποιούν οι ομάδες German Team 2004 και SPQR-Legged 2006 και εμφανίζεται εξίσου καλή ή και καλύτερη σε πολλές περιπτώσεις. Η προτεινόμενη προσέγγιση έχει χρησιμοποιηθεί με επιτυχία από την ομάδα Κουρήτες του Πολυτεχνείου Κρήτης κατά τη διάρκεια διαφόρων διοργανώσεων σχετικών με το RoboCup.

Acknowledgements

At this point, I consider it essential to thank the supervisor of my thesis, Assistant Professor Michail G. Lagoudakis for his guidance and help. I would also like to thank Petros Patelis (team Kouretes 2007 member) for his contribution to beacon recognition, when it was most needed. Many thanks to the other Kouretes 2007 team members: Chrisavgi and Alexandros, for adapting existing image processors to recognize the 2007 beacons; Giorgos, Giorgos, and Andreas for their excellent cooperation within the team.

Readers will hopefully excuse me to continue the rest of this page in my native language, Greek ...

Εκτός από τους παραπάνω ανθρώπους, θέλω να ευχαριστήσω και κάποιους ανθρώπους που η σκέψη τους και μόνο με κάνει να χαμογελάω και να αισιοδοξώ. Πολλά ευχαριστώ, λοιπόν, στην οικογένειά μου για την ασύγκριτη αγάπη και την υποστήριξή τους, αλλά και στους φίλους μου, την Ευτυχία, την Αλίκη, το Νίκο, την Αδερφή, τον Κώστα, τη Θοδώρα και την Κατερινούλα, τη Μαρία και τη Δώρα, τον Καλαμπόκα, το Σπύρο, τη Μερόπη, τον Παπαζάχο και τον Θανάση, τον Άλκη, το Γιαννάκη, τον Πάνο, το Στέλιο, την Κατερίνα και τον Άρη, την Ελένη, την Τζουλιάνα και το Δημήτρη.

Ιδιαίτερα “ευχαριστώ” στον Πέτρο που ήταν πάντα εκεί, παρόλο που βρισκόταν τόσο μακριά.

Τέλος, ευχαριστώ για τη συγκυρία εκείνη που με έφερε στα “όμορφα Χανιά μας” για να ζήσω 6,5 χρόνια γεμάτα απίστευτες, δυνατές στιγμές και συγκινήσεις.

Table of Contents

1 Introduction

RoboCup is an annual international robotic competition. It consists of four different divisions, one of which is the RoboCupSoccer. This division is related to competitive soccer between robotic teams. RoboCupSoccer is further divided in leagues, according to the size and type of the robots participating. The Four-Legged League is one of its most popular divisions. In this league, each team consists of four four-legged Sony AIBO robots which operate and play soccer autonomously within the field. The research objective of this league is the development of software solutions over a standard robotic platform.

There is a variety of problems to be solved in order to assure a successful performance of the players within the field. The robot is expected to walk by moving its four legs in a synchronized manner, to know its position within the field, to coordinate with the other robots, and to correctly perceive the objects around the field. Thus, it gives rise to fields of research such as legged locomotion, localization, coordination, and visual object recognition.

This thesis focuses on the problem of visual object recognition. The key objects in the RoboCup 2007 field are the two goals, the two beacons (colored landmarks), and the ball. Each object has its own appearance characteristics which discriminate it from the other objects. The most important of these characteristics is the color of the object upon which lies the core of image processing and object recognition.

The Image Processor module, which is proposed by this thesis, first processes the color-segmented camera image. The entire image is scanned pixel by pixel, both horizontally and vertically. Large areas of the same color are then determined through a finite state machine, which permits a specific tolerance of non-colored pixels within the colored areas. These areas are likely to be part of an object of interest. In order to figure this out, the colored areas are clustered through histograms, which expose the density of the colored areas and give an indication of their extent. Such indications help in finding out the dimensions of an object within the image. Then, after a series of transformations using the robot head pose, the object dimensions within the image, the camera parameters, and the real world dimensions of the objects, an estimate of the distance and the orientation to each object is computed. Some checks are then performed, related to the relationship between the estimated dimensions and distance of the objects, and the actual ones. Those checks can reject implausible indications. If the acquired information passes these tests, the type and an estimate of the distance and the angle with respect to the robot of each recognized object in the current field of view, is delivered.

Our approach is compared against the image processors of German Team 2004 and SPQR-Legged 2006 by means of statistics in successful object cognition, image processing time performance, as well as the way in which the objects are perceived by the different image processors. The results of this comparison show that our Image Processor seems to be equally good or better in many cases.

We should also mention that the proposed approach has been used successfully by Team Kouretes of the Technical University of Crete during various RoboCup events.

All the details of this thesis work are presented in the chapters that follow. At this point, it is useful to present the report' s outline.

In Chapter 2, information about the RoboCup competition can be found. Some historical details are given, as well as an overview of the divisions of the RoboCup competition. One of these divisions is the RoboCupSoccer which involves soccer games and consists of different leagues. The league of interest here is the RoboCup four-legged league, thus it is described in detail. Information is given about this league' s rules, the field in which the league' s games are performed, the players who participate (AIBO robots), as well as the participating teams.

In Chapter 3, the problem this project is dealing with is thoroughly stated. That is, the objects to be recognized are presented in detail, the environment, in which the problem is stated, is described, and some general principles about the problem are also listed.

Chapter 4 reviews the work of four RoboCup four-legged league teams: German Team, rUNSWift, Cerberus, and SPQRLegged team. These teams' approach to the specific problem has in many ways helped our comprehension of the problem and the evolution of our solution.

Chapter 5 is the one that describes the approach to the problem as stated in Chapter 3. Chapter 5 is divided into four sections: sections 5.1, 5.2, and 5.3 deal with the recognition of the goals, the beacons, and the ball, respectively. The recognition process is described for each object in detail. Then, section 5.4 gives technical information about the implementation of the our approach.

In Chapter 6, the proposed approach is evaluated and the results are discussed. Statistics of the object recognition performance are presented in section 6.1 and a comparison with other RoboCup four-legged teams is conducted in section 6.2. This comparison includes statistics of successful object recognition, average image processing time and, qualitative aspects of recognized objects.

Finally, Chapter 7 concludes the report. This last chapter presents the achievements of the present work, as well as suggestions for further research.

2 RoboCup Four-Legged League

This chapter provides some general information about the RoboCup competition and its divisions. Most of the attention is driven towards RoboCupSoccer and, in particular, on the four-legged league. This league's rules, field, players (Sony AIBOs) and teams are about to be described.

2.1 RoboCup

RoboCup (Robot Soccer World Cup) [1] is an international robotics competition. It was founded in 1993 and has been taking participants in a different country once a year, since 1997 [3]. Every year RoboCup take a further steps in developing robots that act fully autonomously within a specific environment where a standard problem must be addressed. This robotics competition intends to encourage research in Artificial Intelligence and Robotics as well as enhance the education and experience of the competitors in these fields.

The main divisions of RoboCup are: RoboCupRescue, Robocup@Home, RobocupJunior and RoboCupSoccer.

RoboCupRescue [1] is focusing on the development of robotic agents designed to assist rescue teams in the case of a disaster within an unstructured environment where human access is almost impossible. RoboCupRescue consists of two different leagues: RoboCupRescue Robot League and RoboCupRescue Simulation League.

RoboCup@Home [1] deals with robots that develop their abilities within a home-like environment with the intention of helping people with the common housework.

RobocupJunior [1] is highly associated with K-12 education, as it is directed towards young students who are given the opportunity to develop their skills in fields such as robotics and electronics. The participants are also taught how to cooperate and act within a team, acquiring a lot of experience.

RoboCupSoccer [1] represents the main RoboCup division and has to do with competitive soccer games using different types of robots. The robots compete in teams within a clearly defined environment giving their “coaches” the opportunity to enhance their technical skills through hardware and software. Each RoboCupSoccer game is also a spectacular event for the people watching the game. This RoboCup division is further divided into five different leagues:

- Simulation league:
  This is one of the oldest RoboCupSoccer leagues. During the games of this league, the participating robots are software entities created and programmed within a computer (Figure 2.1). They are independent and play soccer on a virtual field. The game consists of 5-minute halves.

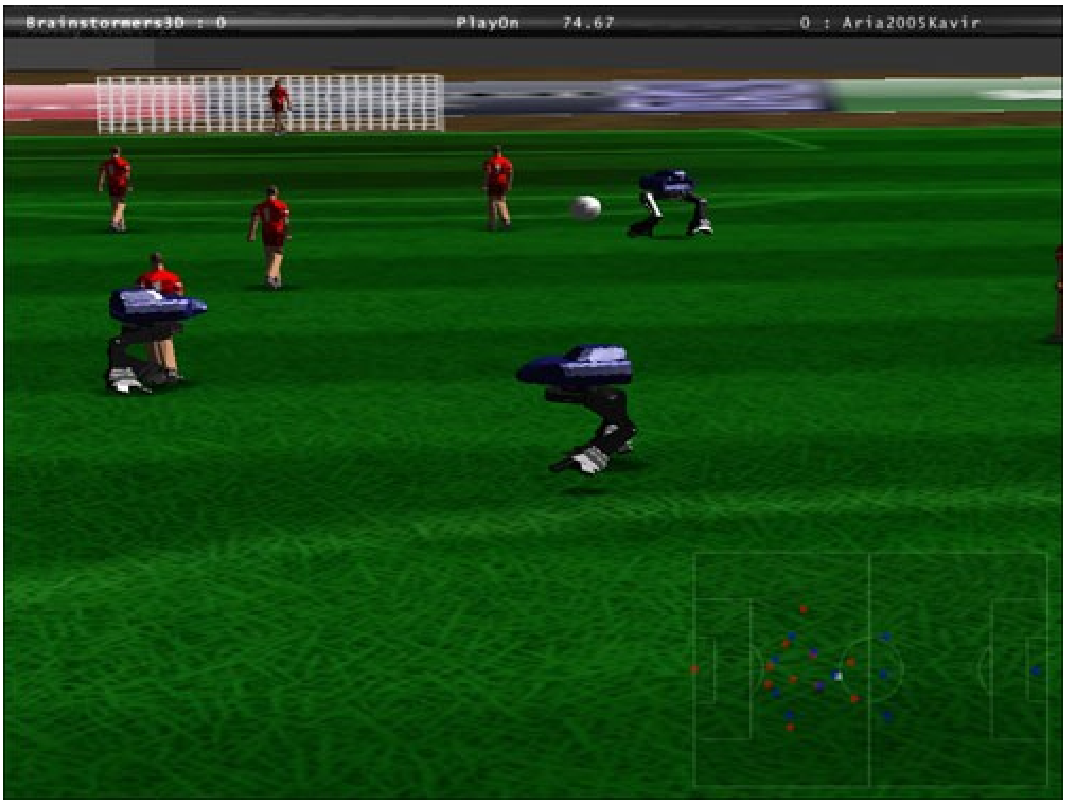


Figure 2.1 – Snapshot from the RoboCupSoccer simulation league [24] .

- Small-size robot league:
  In this league, small cylindric robots of about 18cm diameter play soccer on a field quite bigger than a ping-pong field with an orange golf ball (Figure 2.2). Each team consists of up to 5 players and the game consists of 10-minute halves.

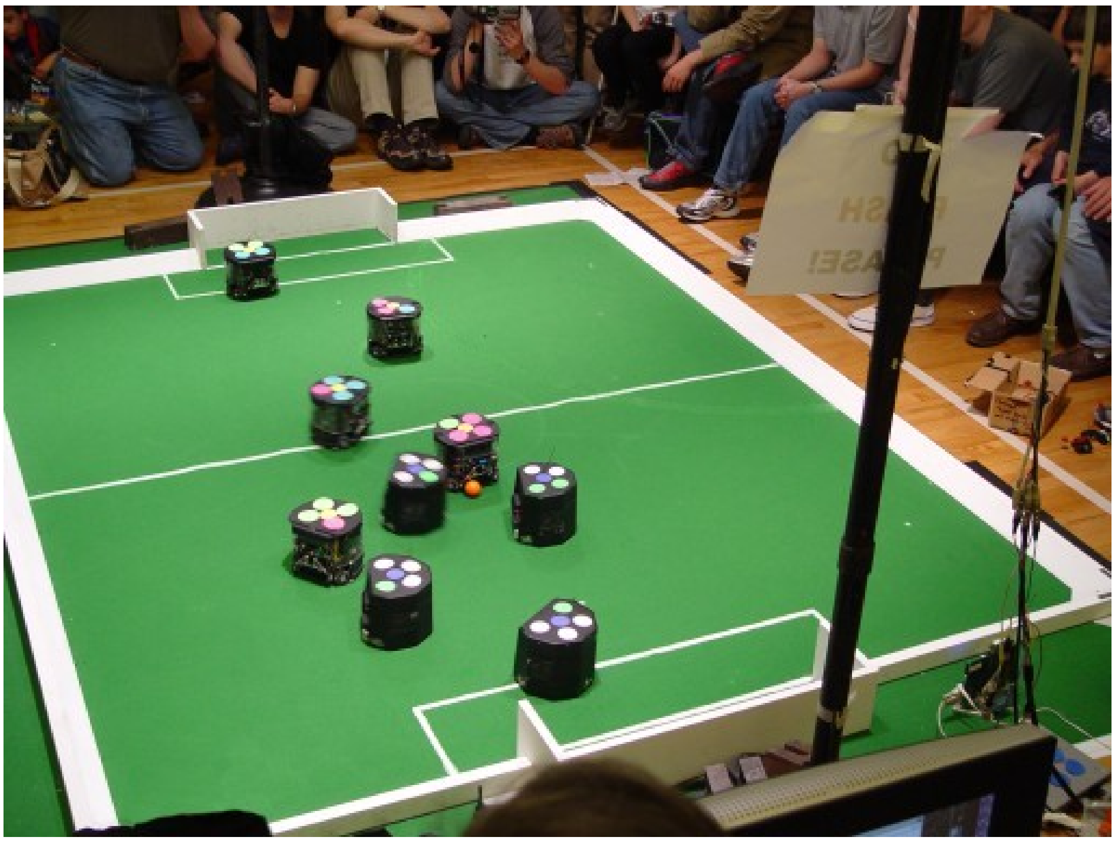


Figure 2.2 – The RoboCupSoccer Small-size robot league [25].

- Middle-size robot league:
  Cylindric robots of about 50cm diameter play soccer on a field of 12x8m with an orange soccer ball (Figure 2.3). Each team consists of up to 4 players and the game consists of 10-minute halves.

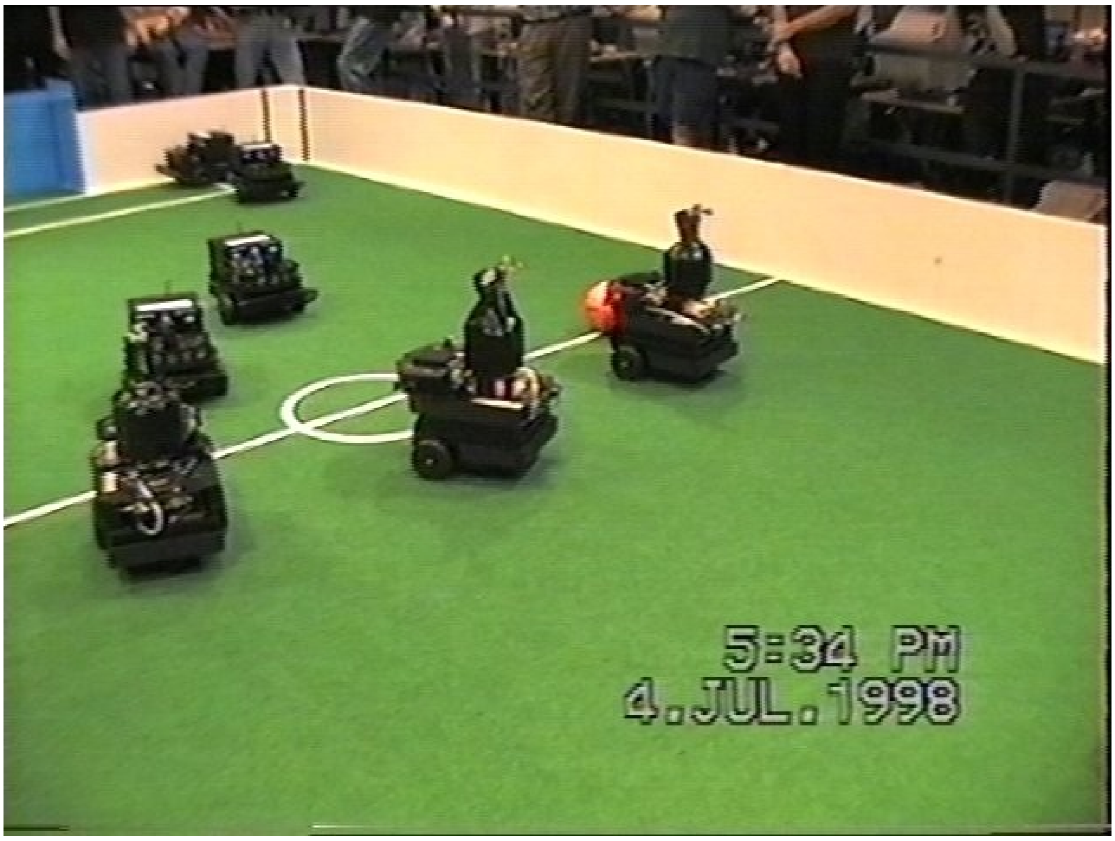


Figure 2.3 – The RoboCupSoccer Middle-size robot league [26].

- Humanoid league:
  This league was first introduced in 2002. The league' s players are autonomous biped humanoid robots (Figure 2.4) and play penalty kicks, 1 vs. 1 and 2 vs. 2 matches.

- Four-legged robot league:
  In this last league, four-legged entertainment robots play soccer on a field of 6x4m with an orange plastic ball (Figure 2.4). Each team consists of to 4 players and the game consists of 10-minute halves. This league is described in detail in the following sections of this chapter.

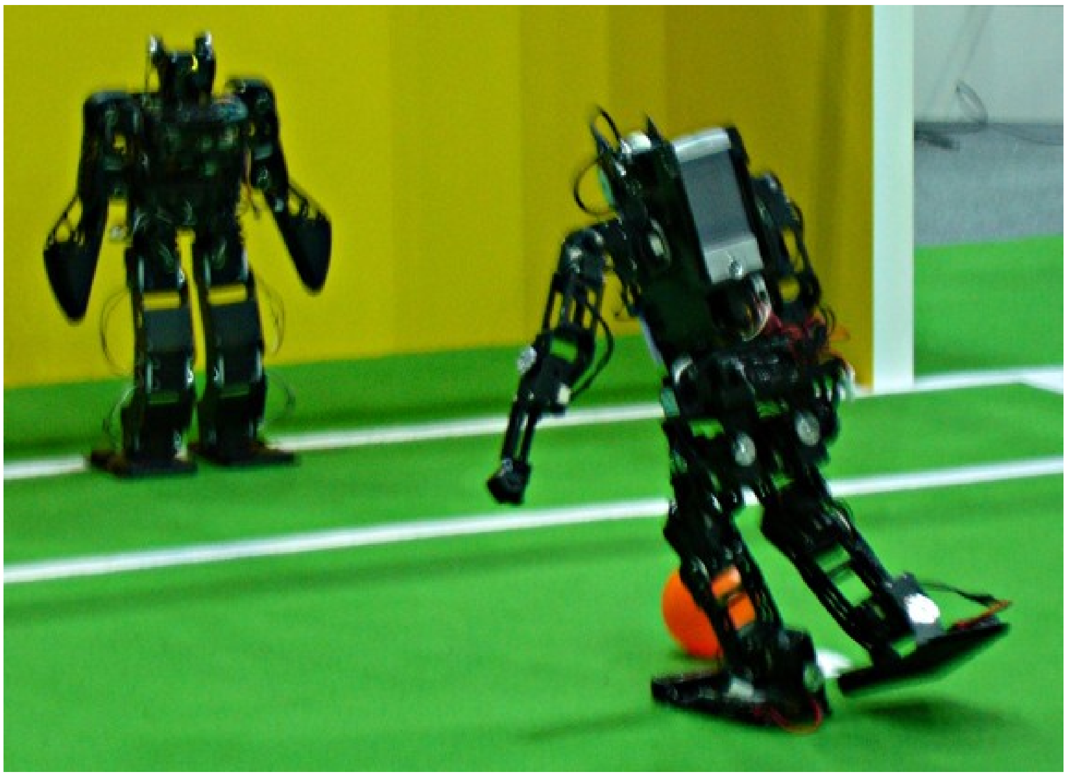


Figure 2.4 – The RoboCupSoccer humanoid league [27].

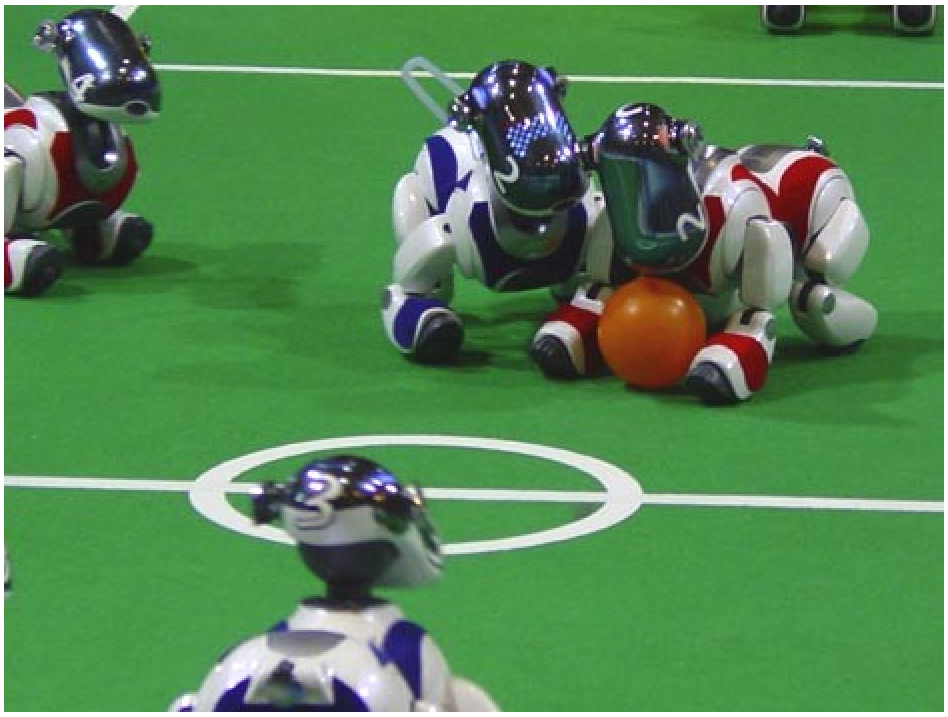


Figure 2.5 – The RoboCupSoccer four-legged league [28].

As one could guess by their names, the main difference among the leagues is the size and type of the robots taking part.

Besides education and entertainment, the RoboCup project intends to advance robotics research towards real-world applications. RoboCup' s ultimate goal is ambitions:
“By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup” [2].

2.2 RoboCup Four-Legged League Rules

Each RoboCup soccer league has its own rules and regulations. In the Four-Legged League each team consists of four four-legged SONY AIBO entertainment robots. One of them plays as a goalkeeper and the rest three share the roles of the attacker and defender, according to the way they are programmed to play (for example, two defenders and one attacker). All robots compete in on a 4 x 6 meter field.

Within the field the difference in color among the surroundings of the robot is of great importance and determines its visual perception. The floor is green, the lines that define the areas of the field are white, the ball is orange, the players can be either blue or red, there is a blue and a yellow goal and there are also beacons, that is markers that help the robots to be localised within the field, whose number and colors have varied during the years of the competition.

The game is conducted by a human referee who is in charge of notifying the two halves of the match – ten minutes each, and also to penalize the robots that do not play according to the rules. Disregard to the rules, such as illegally pushing a robot, leaving the field or illegal defence, comes with a penalty of 30 seconds' exclusion from the game. Usually, there are also two assistant referees whose job is to place the robots manually to the correct positions in and out of the field.

During the game, the only external control is applied to the robots is the game controller, which uses the wireless communication to indicate the start, pause, goals, penalties and the end of the game. By all other means, the robots operate fully autonomously.

2.3 RoboCup Four-Legged League Field

Concerning the RoboCup Four-Legged League field, there has been an evolution in its appearance during the years of competitions. Each team consisted of 3 robots including the goal-keeper and there were 6 beacons, that were used for the localization of the robot in the field, as shown in Figure 2.6 [4], until the 2003 competition of the year [4] [5] [6] [7] [8].

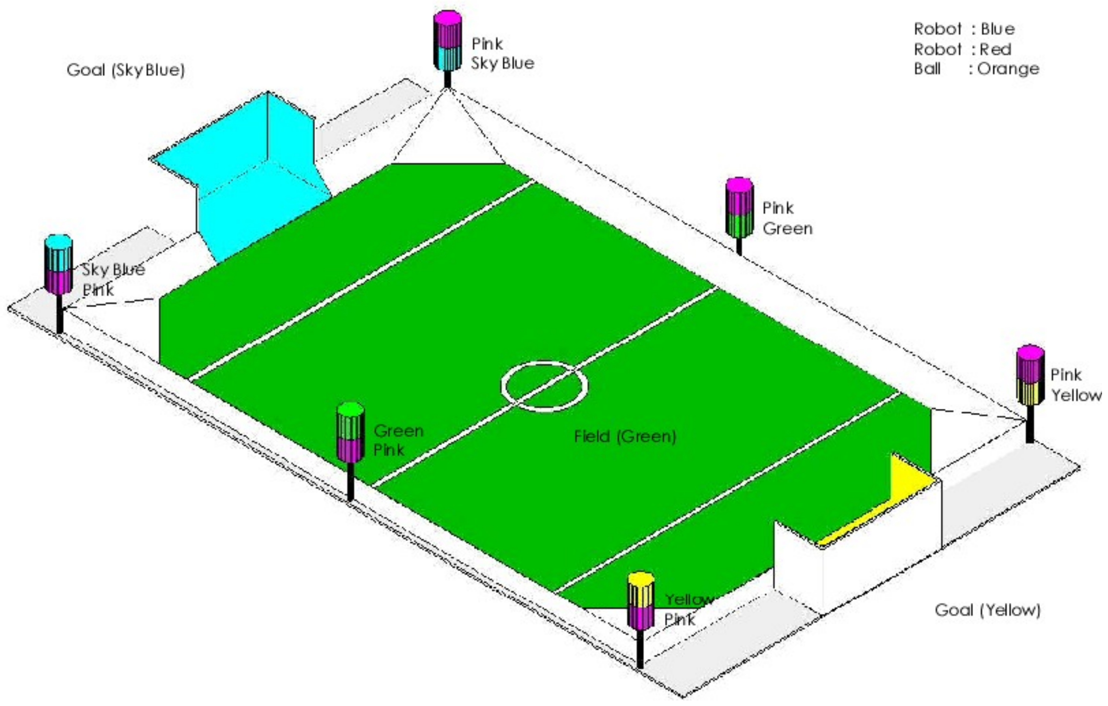


Figure 2.6 - The four-legged league field in 2000.[4]

The goals differed from 1999 to 2007 as it is shown in the pictures of Figure 2.7: in 2000 the goal loses its upper edge whereas in 2001 the goal's floor becomes green. The goal appearance remains like this until the year 2007 when it changes again, as it is described below.

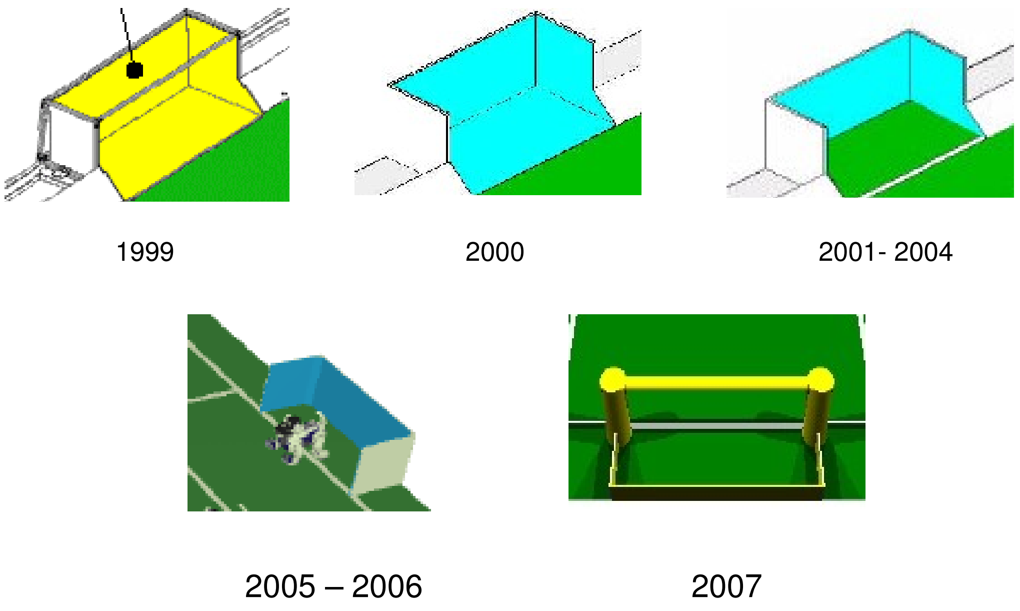


Figure 2.7 - Goals evolution during the years 1999 to 2007 [4] [5] [6] [11] [12].

In 2003 [8] there was a change of field lines concerning the penalty area as we can see in Figure 2.8. There was also a white area added below the colored areas of each beacon.

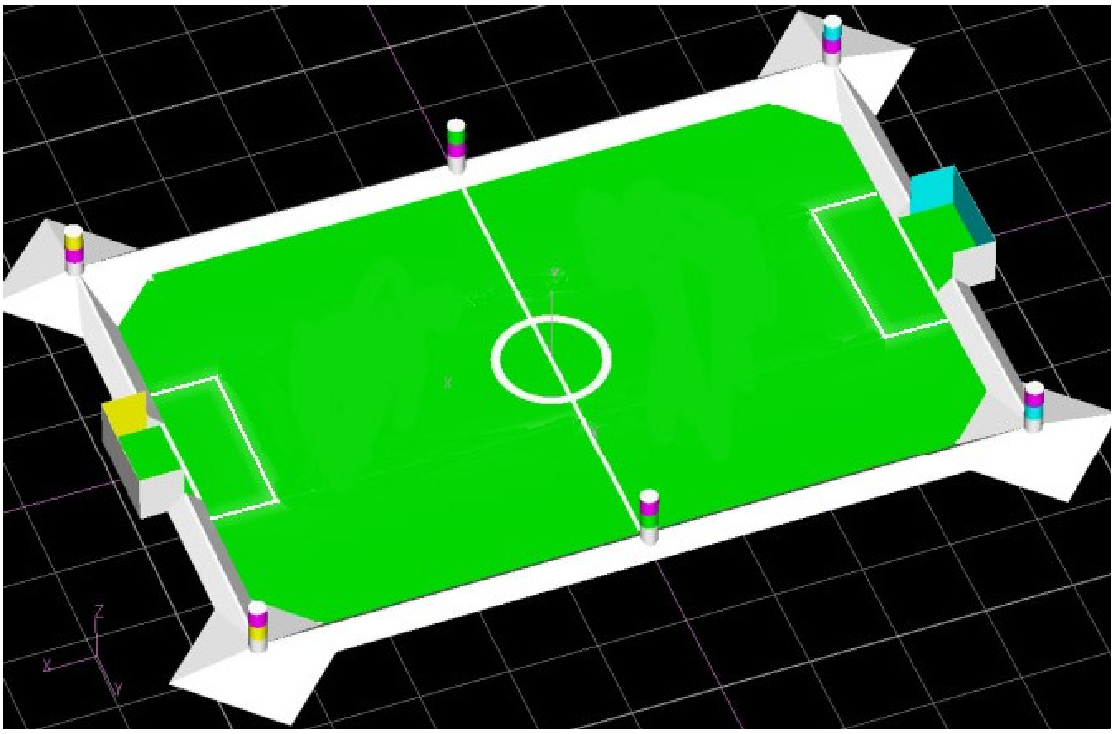


Figure 2.8 - Lines' changes in 2003 field. [8]

In the 2004 competition [9] there were several important changes. The number of each team’s players increased from three to four. The field beacons are now four instead of six and located at the corners of the field. The beacon colors are only three this year: yellow, blue and pink, instead of four (yellow, blue, pink and green) as it used to be in the previous competitions. The beacon's combinations this year are: pink up/yellow down, yellow up/pink down, pink up/sky blue down and sky blue up/pink down. These combinations were preserved until the 2006's competition. In addition, in 2004 there was a reduction in the size of the center circle of the field. We can confirm those changes by looking at Figure 2.9 [9].

The beacons’ position as well as the field’s white surrounding were changed for the RoboCup 2005. Each beacon was placed in the middle of the field’s half at both sides according to its color, as shown in Figure 2.10 [10].

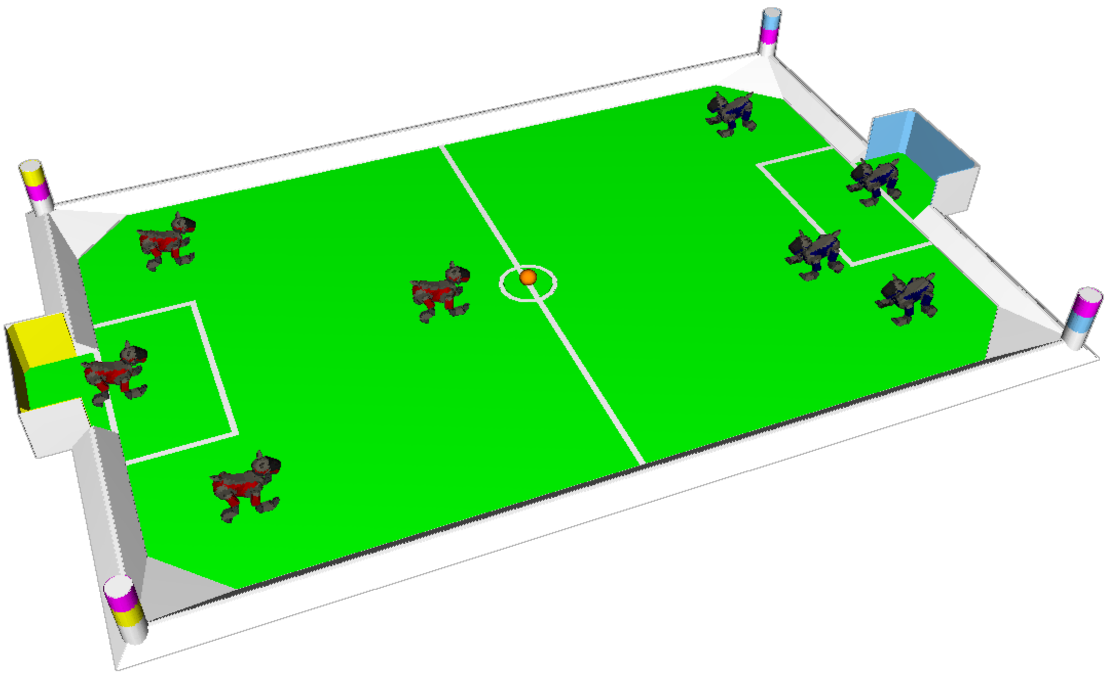


Figure 2.9 - Foul-Legged League in 2004. [9]

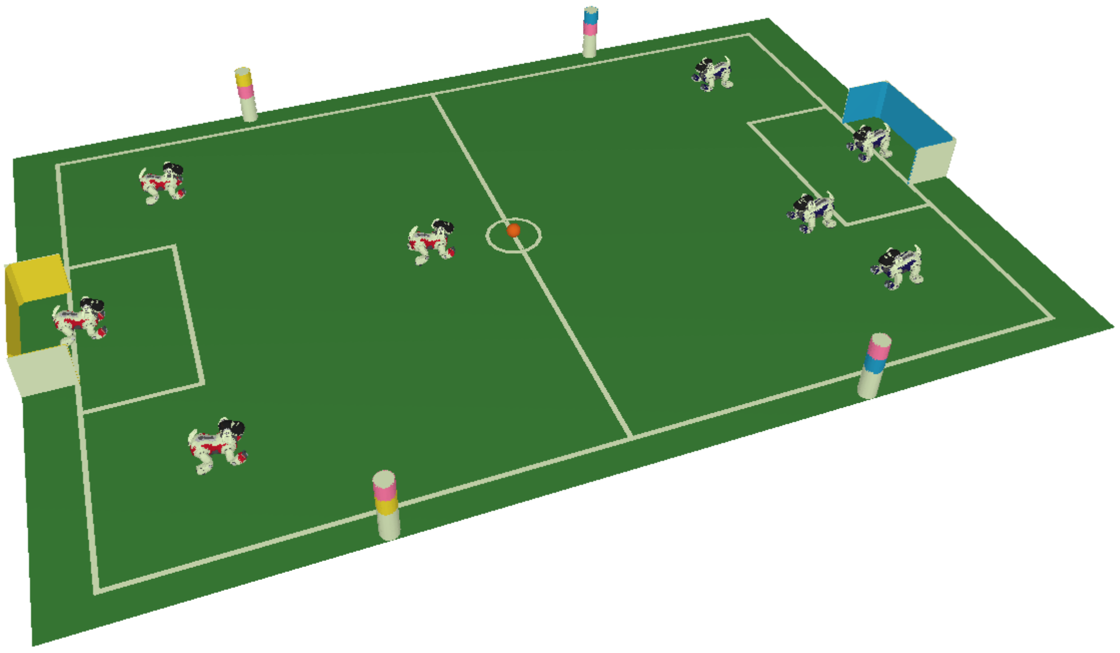


Figure 2.10 - Four-legged league field in 2005. [10]

Finally, in the RoboCup 4-Legged League 2007 [12] in comparison to the previous competitions [10] [11] the beacons of the field have been reduced from four to two as shown in Figure 2.11 [12] when compared to Figure 2.10 [10]. In addition, the structure of the goals has changed. These two specific changes will also compose the subject of following chapters.

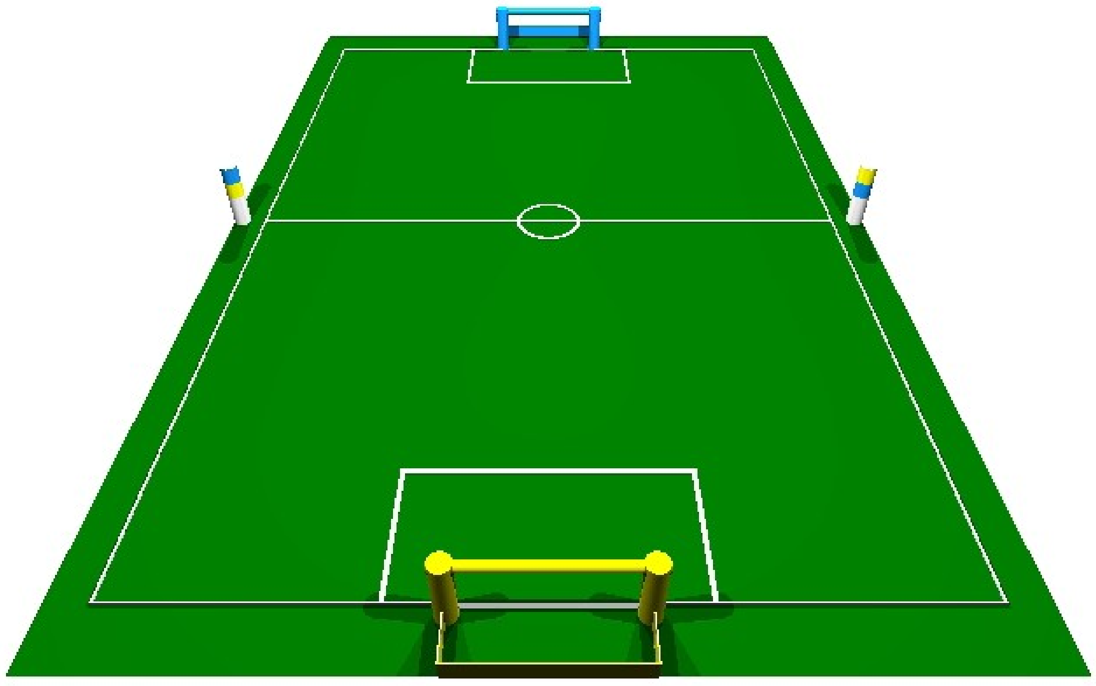


Figure 2.11 - Four-Legged League field in 2007. [12]

2.4 RoboCup Four-Legged League Players

As we have already mentioned, RoboCup four-legged league players are the AIBO entertainment robots by SONY. The name “AIBO” (Artificial Intelligence roBOt) comes from the Japanese word that means “companion” or “friend”. AIBOs have been produced and released by Sony Corporation in 1999, but their production ceased in March 2006.

Regarding AIBO' s technical characteristics, it is run by a 576 MHz, 64-bit RISC processor with 64 MB RAM memory. It supports 802.11b wireless LAN (2,4 GHz) and also has a 64-chord midi synthesizer. As far as programmability is concerned, the robot functions according to the code that is saved in a memory stick. This stick can be accessed by a reader/writer in the body of the robot. In order to operate autonomously, the robot also has one lithium-ion battery which gives him 1,5 hours of operating time. Including the memory stick and the battery, an AIBO weighs about 1.6 Kg and its dimensions are approximately 180 (w) x 278 (h) x 319 (d) mm.

There are also some external features that should be mentioned. Every AIBO has its own color CCD camera of 160x208 pixels resolution, 30 frames per second, 56.9° wide and 45.2° high. It also possesses a great variety of sensors. More precisely, it has got three infra-red distance sensors, five touch sensors (one on the head, one under the jaw and three on the back), a paw sensor on each paw, two audio sensors on the ears for stereo audio, four movement sensors (three accelerometers and one vibration sensor) and a temperature sensor. Last but not least, there is a battery meter, as well. The exact position of the aforementioned features can be seen in Figures 2.12 and 2.13 [19] .

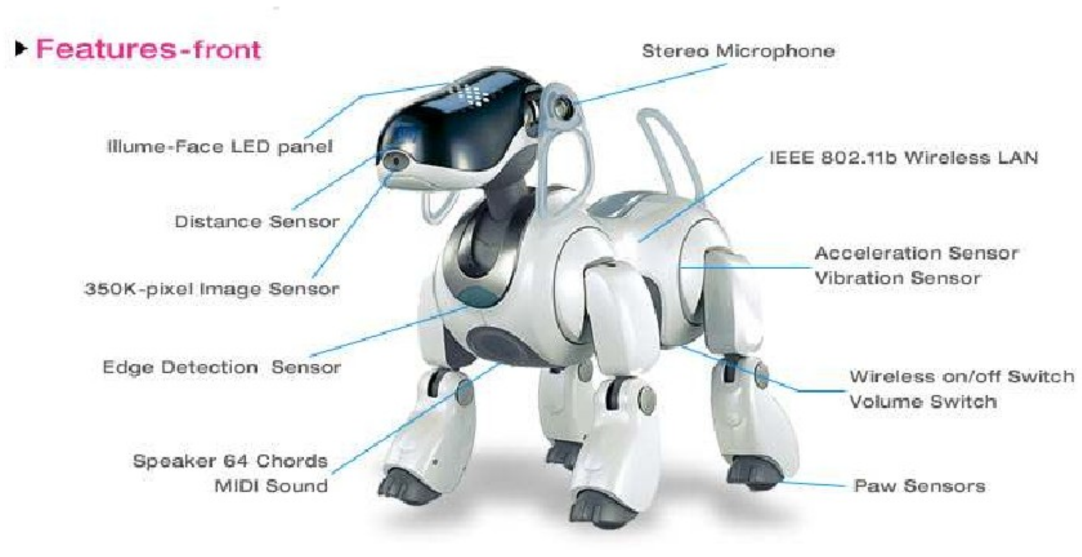


Figure 2.12 – AIBO' s features (front side) [19] .

Concerning the AIBO' s movement, there are 20 degrees of freedom in total which allow it to move and walk. They are distributed among the four legs (three degrees of freedom in each leg), the head (three degrees of freedom), the mouth (one degree of freedom), the tail (two degrees of freedom) and the two ears (one degree of freedom each). In addition, the robot carries 26 LEDs, a head LED display panel and a speaker (500mW). These features are also depicted in Figures 2.12 and 2.13 [19] .

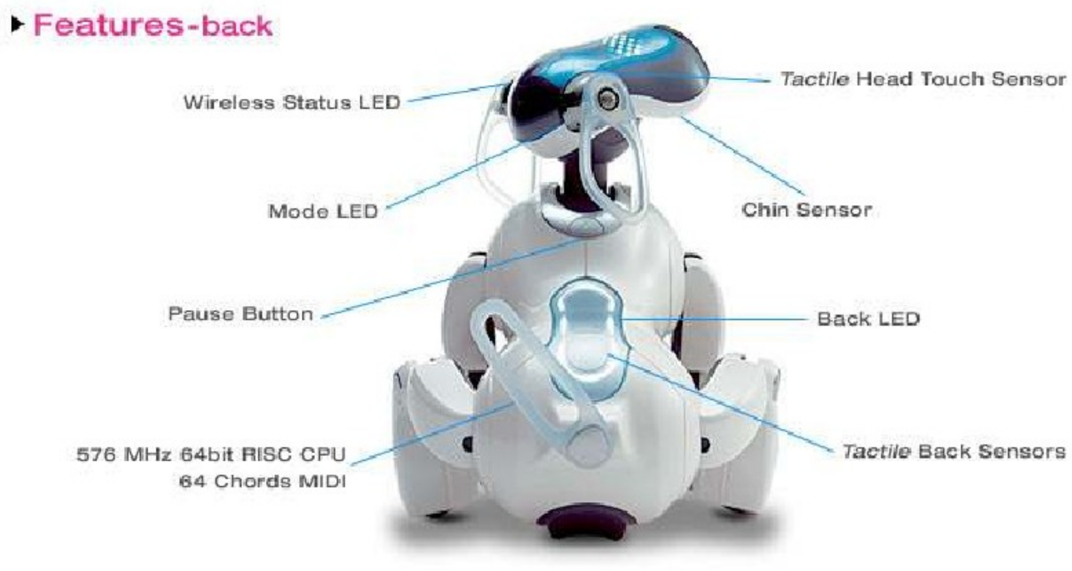


Figure 2.13 – AIBO' s features (back side) [19] .

To conclude, AIBO' s ability to be fully programmed, its great number and variety of motors and sensors, as well as its relatively low price (about €2000 in Europe) made AIBO very popular for both entertainment and educational purposes all around the world.

2.5 RoboCup Four-Legged League Teams

During the years, there have been several teams from all over the world participating in the four-legged league of the RoboCup competition. However, in this text we are focusing on the RoboCup 2006 teams, whose work we have studied in order to acquire knowledge that would help as to confront our own problem.

The participating teams in the RoboCup 2006 four-legged league were the following:

- German Team, collaboration of four different German Universities and laboratories
- Nubots, University of New Castle in Australia

- Cerberus, Bogazici University in Turkey
- Northern Bites, Bowdoin College in U.S.A.
- SPQR, University of Rome “La Sapienza” in Italy
- Dutch AIBO Team 2006, collaboration of several institutes of Netherlands
- Jolly Pochie, collaboration of two different Universities of Japan
- rUNSWift, University of New South Wales and National ICT in Australia
- UPENNALIZERS, University of Pennsylvania in U.S.A.
- UT Austin Villa 2006, University of Texas at Austin, U.S.A.
- Wright Eagle, University of Science and Technology of China
- ARAIBO, The University of Tokyo and Chuo University in Japan
- ASURA, Kyushu and Fukuoka Institutes of Technology in Japan
- BabyTigers DASH, Osaka City University in Japan
- Eagle Knights, “Instituto Tecnológico Autónomo de México” (ITAM)
- Hamburg Dog Bots, “Universitat Hamburg, Fachbereich Informatik, Arbeitbereich Technische Informatiksysteme“ in Germany
- Impossibles, Sharif University of Technology in Iran
- Mi-Pal Team Griffith, Griffith University and Institute for Integrated and Intelligent Systems (IIIS) in Australia
- Microsoft Hellhounds, Robotics Research Institute Information Technology Universitat Dortmund in Germany
- SpelBots, Spelman College of Atlanda in U.S.A.
- TeamChaos, collaboration of Spanish and Swedish Universities
- Twaves, Tokai and Tamagawa Universities in Japan
- UChile1, Universidad de Chile
- Kouretes, Technical University of Crete in Greece
- IsocRob-4LL, Instituto Superior Tecnico in Portugal
- sharPKUngfu, Peking University in China
- TsinghuaHephaestus, Tsinghua University in China

3 Problem Statement

This chapter describes the problem that the present work deals with. It is necessary to state here the objective, environment and essential specifications of the problem.

The present thesis copes with the problem of object recognition within the 2007 RoboCup four-legged league field by the Sony AIBO robot. The objects of interest here are the two goals, the two beacons and the ball. The robot needs to recognise these objects in order to know its position within the field, as well as its distance from these objects and play soccer according to the way it is programmed to. In order to address this recognition, the raw material to be processed is the video stream from the robot' s camera combined with the pose of the robot at each moment. This material is further color segmented in order to provide the image to which the processing is going to apply. Especially, the pose of the robot' s head, where the camera is placed, is taken into account, so as to conclude the relationship between the camera image, the real world and the orientation of the robot.

3.1 Field objects

In this section the objects of the field that should be recognised by the robot are described. Characteristics such as the color, appearance and dimensions are here separately mentioned for each one of the field objects.

Goals

The robot should be able to recognise the two goals of the field. The goal color can be either blue or yellow (Figure 3.1). Each goal is composed by pieces whose dimensions are specified (see Figure 3.2 [12]). Two cylindrical pieces of 30cm each, stand for the vertical goalposts and support the upper

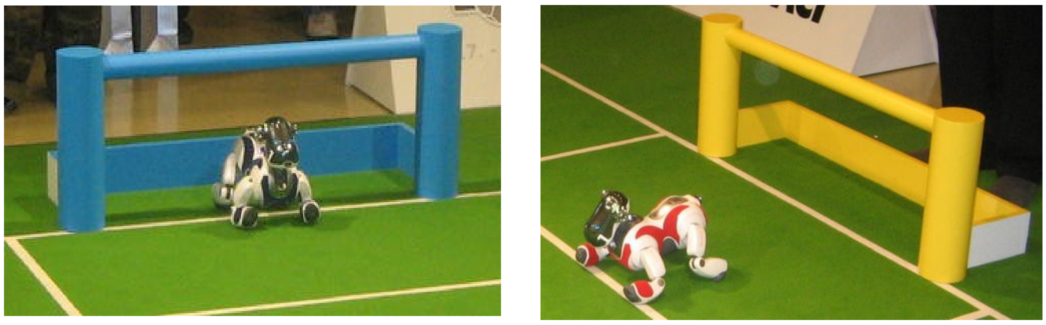
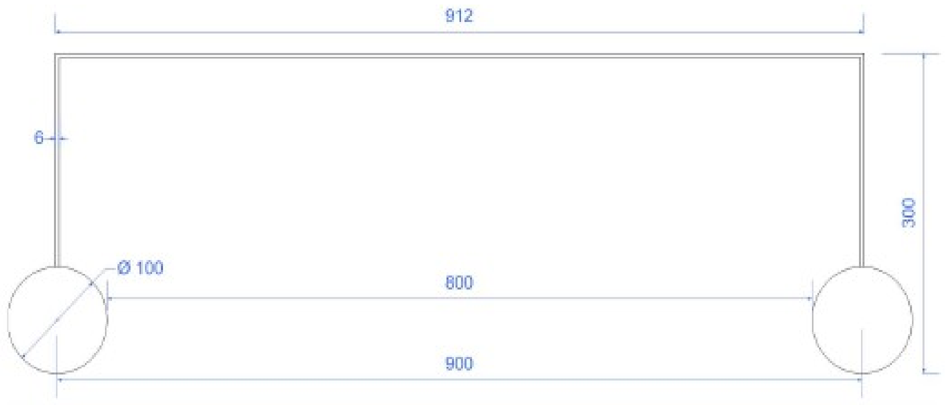


Figure 3.1 – The blue and yellow goal of the 2007 RoboCup four-legged league field.

horizontal goalpost which has a length of 80cm. There is also another part of the goal which is connected with the lower part of the vertical goalposts and delimits the region of the goal on the field surface.



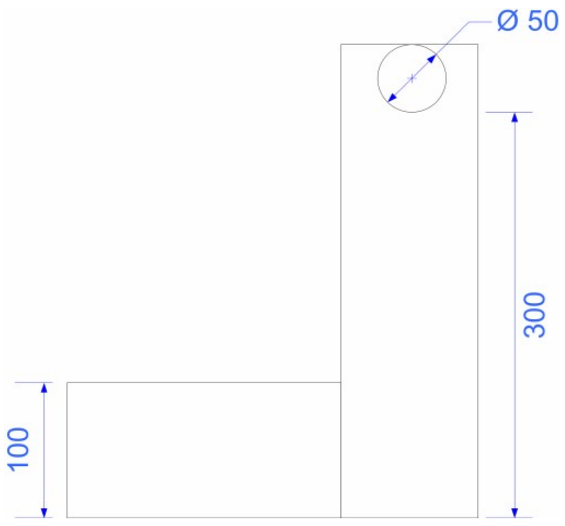


Figure 3.2 – The dimensions of the goal parts. The goal is shown from above (top sketch) and from the side (bottom sketch) [12].

Except for their obvious use for scoring, the goals also serve as landmarks for the localization of the robot within the field. Another couple of landmarks in the field are the beacons that are described below.

Beacons

The beacons of the 2007 RoboCup four-legged field are cylindrical wooden pieces of 40cm height, which are painted white, blue and yellow in two different combinations (from top to bottom): blue-yellow-white and yellow-blue-white (see Figure 3.3). They are placed at fixed positions, either side of the field middle line, and therefore can facilitate the localization process if recognized correctly.

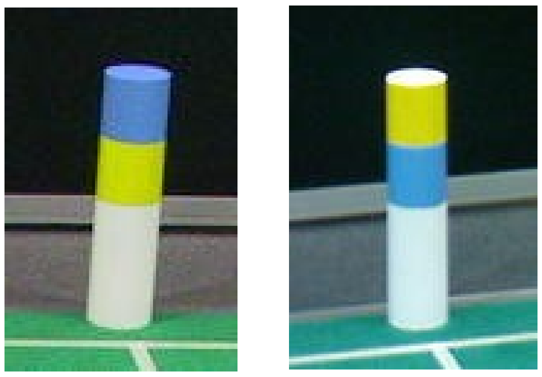


Figure 3.3 – The blue-yellow and the yellow-blue beacon of the 2007 RoboCup four-legged league field.

Starting from the bottom, each beacon consists of a white part 20cm high and then, depending on its type, either a yellow part of 10cm high followed by a blue part of equal height or vice versa (see Figure 3.4 [12]).

The exact position of both the goals and the beacons within the field is going to be the subject of the following section.

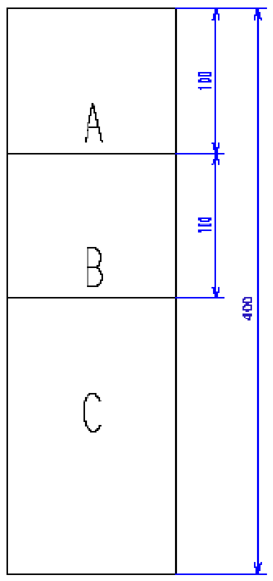


Figure 3.4 – The dimensions and the color proportions of the beacons in the 2007 RoboCup four-legged league field: C is the white part, whereas A and B can be either blue or yellow, depending on the beacon type [12].

Ball

The ball that the robot should recognise within the 2007 RoboCup four-legged league field is an orange ball of 10cm radius, made of plastic. It is depicted in Figure 3.5. Note the reflection effect at the top of the ball due to its smooth surface.

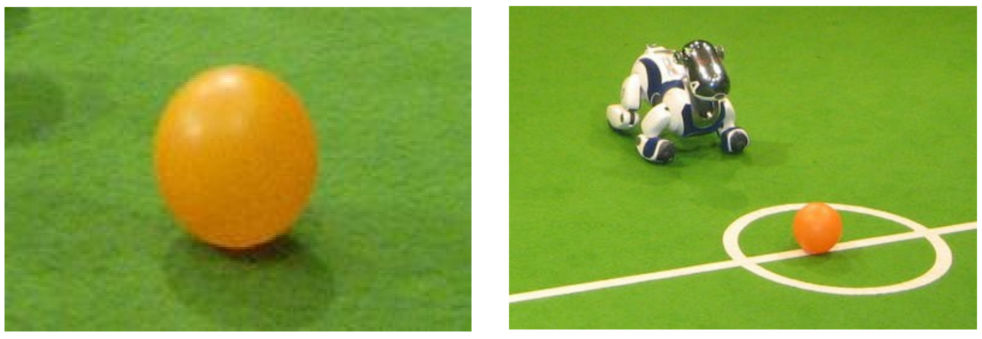


Figure 3.5 – The ball of the 2007 RoboCup four-legged league.

3.2 Environment

Except for the general appearance of the field objects to be recognised it is also important to mention the exact position of each object within the field, as well as the environment within which the object recognition should take place.

Each goal is placed in the middle of one of the small sides of the field as it is shown in Figure 3.6 [12]. On the other hand, each beacon is placed in the middle of the large sides of the field. The position of the goals and beacons are fixed. For example, if a robot stands in front of the blue goal with its head towards the yellow goal, then the yellow-blue beacon should be on its left and the blue-yellow beacon should be in its right. Finally, the ball can roll in any place inside (or outside) the field area.

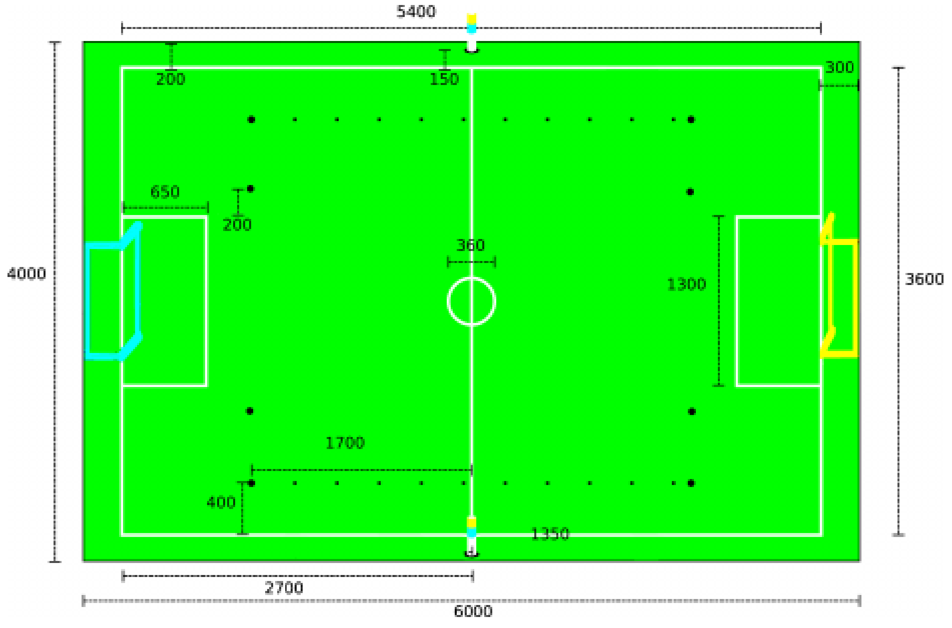


Figure 3.6 – The 2007 RoboCup four-legged field: the field dimensions (shown in mm), the field objects' position and the field lines [12].

Except for the field objects, that have already been mentioned, there are also field lines in white color (see Figure 3.5), the robot players in blue or red uniform, depending on the team that they belong to, and, also, the human referees that are usually dressed in black. In addition, there is no protective surrounding around the field area, so the robots' camera may as well perceive the area outside the field. As far as the lighting conditions are concerned, only ceiling lights are used, resulting in lighting of about 1000 lux.

Now that the specifications of the problem are clear, it is useful to describe the existing approaches to the problem and choose which characteristics are to be adopted and which should be rejected. This will be the subject of the following chapter.

4 Related Work

During this chapter the work of four representative RoboCup Four-Legged teams in the area of object recognition within the field is reviewed. Their work was the most influential to our work. It is for the German Team, the rUNSWift team, the Cerberus team and the SPQRLegged team. Here is described how they process the robot camera images and finally reach recognition of the goals, the beacons and the ball.

4.1 German Team

The German Team (GT) [15] is a collaboration of the Humboldt University of Berlin, the University of Bremen and the Technical University of Darmstadt. It has been participating in RoboCup four-legged league since 2001. German team' s open source code as well as its detailed documentation have been a useful starting point for many other teams that followed.

According to German Teams' s work [16] for the 2004 RoboCup competition, the robot' s perception relies on processing of the images coming from its color camera. This process consists of several steps. First of all, the horizon of the image is calculated with regard to the camera's parameters

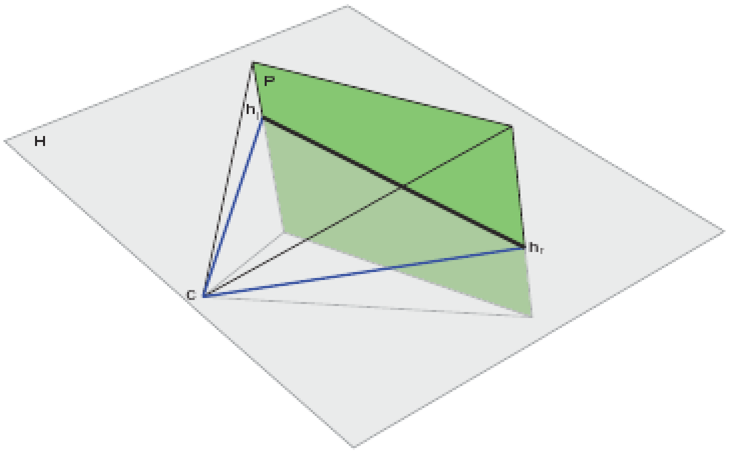


Figure 4.1 – Image horizon construction. [16]

(resolution and angle) and the rotation of the camera relatively to the robot's body. In Figure 4.1, the horizon is defined by the points hl and hr. These are the points of intersection between the plane H, which is parallel to the ground at the height of the point C, and the plane P, which is the plane where the real world' s objects are projected by the robot's lens and is perpendicular to the optical axis of the robot. The point C is the center of this projection as well as the focal point of the camera lens.

Once the horizon has been constructed, processing of the image begins. The image is scanned by scan lines which are constructed to be parallel (starting from the left, ending at the right of the picture) and perpendicular (starting from the top, ending at the bottom of the picture) to the horizon. The density of these lines is not uniform. The scan lines become denser as we approach the horizon line, where the objects to recognize become smaller, therefore higher resolution is needed. The German team uses three different grids of scan lines depending on the kind of objects to be detected.

Each one of the scan lines scans the image pixel by pixel and categorizes the pixels into specific color classes. A continuous appearance of pixels of the same color class states a possible indication of a field object. For example, orange pixels indicate a possible ball, sky blue pixels indicate either a goal or a part of a beacon. Thus, there are several tests, regarding the dimensions or the position of the possible objects, that follow and verify their kind. These tests differ in regard to the kind of object to be recognized, so we are going to explain the procedure for each object in detail.

In order to detect a goal (2004 type), the German Team first uses a grid of scan lines of a fixed density around and parallel to the horizon line. The image is scanned for sequences of goal colors (i.e. continuous yellow or sky blue pixels) from the left to the right and if any indications for a goal have been found during this horizontal scan, a vertical scan follows, from the top to the bottom, perpendicular to the horizon. During the scan, particular small gaps of pixels of another color or unclassified ones are accepted. However, when a large gap has been scanned, then the border of the goal has been found and is the last pixel to have a goal color. The size of the goal, is determined by the distance between the goal borders in each direction. Information about the exact position of the four edges of the goal is provided to the *PerceptCollection* module which is the next step of the robot' s vision.

For beacon detection (2004 type), the German team uses another grid of scan lines, which is also constructed parallel to the horizon line. However, the density of these scan lines increases as they get close to the horizon line. These scan lines are looking for pink pixels. When found, they are clustered and compose a horizontal indication of the pink color; a new scan line. From that scan line, three or four other scan lines are constructed to be perpendicular to it and scan vertically for the color (yellow or sky blue) of the other half of the beacon. After that, different sanity and comparison checks are performed in order to verify the existence of a beacon and, if they come true, the center of the pink area is calculated. From this starting point, the image is scanned in all four directions so the beacons' borders are detected. In addition, two more couples of horizontal scan lines, one for the pink and one for the other (yellow or sky blue area), as well as three vertical (i.e. perpendicular to the horizon) scan lines also scan the image, so as to reach the final accurate result about the beacon's edges. This information is further sent to the *PerceptCollection* [16].

Regarding the ball, the German team used the main grid of scan lines, which is constructed so as to scan around and below the horizon. The scan lines search for the largest orange part of the image and, when found, its center is calculated. The center is the starting point of eight new scan lines that scan horizontally, vertically and in diagonal in order to find the edge of the ball. Some further color checks are performed to avoid false ball perceptions and, then, the ending points of the scan are saved in a list. The center and radius of the ball are found using the Levenberg Marquardt [16] method taking into account points from the list.

4.2 rUNSWift

The rUNSWift RoboCup four-legged league team is based at the University of New South Wales School of Computer Science and Engineering in Sydney, Australia, and first participated in the RoboCup competition in 1999 [21] . This team has introduced really innovative ideas to RoboCup research which contributed to an amazing performance during several competitions.

One of their innovations is the solution they have given to the problem of variations in lighting conditions due to shadows, reflections or non-uniformity of the lighting source within the field . Thus, they have introduced the difference between neighbouring pixels as a relative color value, instead of using the absolute color value of the pixel, transforming the static color classification into a dynamic one.

Another important approach of the rUNSWift team is their sub-sampling of the image. They are stating that the important information concerning the RoboCup four-legged league lies upon the edges of the field' s objects. Aggregating a great deal of pictures taken by the robot' s camera, showed that this kind of information is concentrated mostly around the horizon line within the image. So, this fact led the rUNSWift to use grids of scan lines which are constructed based on and around the horizon line. A view of those grids is shown at the snapshots of Figure 4.2 [20].

The object recognition phase begins with feature detection. During this phase, the image is scanned with the use of the aforementioned grids and each pixel' s value is compared with its previous one in the scan line. If a rapid value transition occurs, that means that an object feature has been detected. For example, if the color value changes from green to orange then a ball feature has been detected.

The features detected during the previous phase are grouped together in order to create the complete object currently being detected. This procedure is going to be described in detail depending on the object that is to be detected.

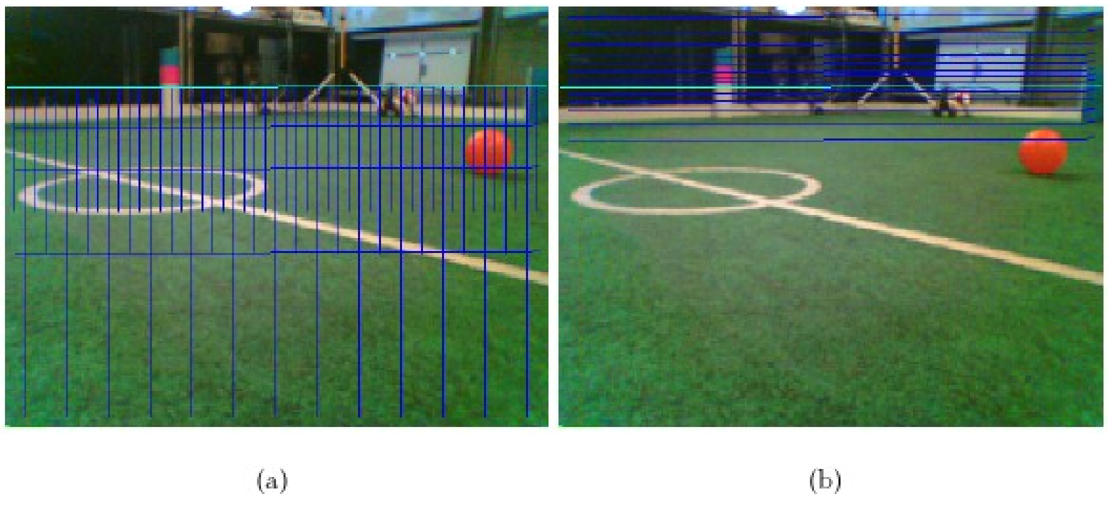


Figure 4.2 – The rUNSWift team scan line grids: (a) This grid is used to detect the ball, field, lines and obstacles. (b) A grid of scan lines used to detect landmarks. Both grids scan lines are constructed to be either parallel or perpendicular to the horizon [20].

Concerning the goals (2004 type), the blue and yellow features that have been previously detected are of interest. The features of the same color that overlap, are merged to create goal features. Then, the centroid of the box that surrounds the goal features is calculated, so as to have an estimation for the centroid of the goal. The goal' s height comes from the construction of one line for each goal feature which is perpendicular to the horizon and searches for the height of the goal feature. The maximum line' s length stands for the height of the goal.

The rest of the goal properties, such as the distance between the goal and the robot, are calculated using transformations, as well as previously found values (height), the position and size of the goal within the image, the robot' s pose and camera parameters. In addition, several checks are performed onto the acquired information about the goal, concerning the shape, the relative position to the horizon and the color of the surrounding pixels, so that the reliability of goal recognition is assured.

Beacon recognition (2004 type) relies on the pink features that have been detected before. The features belonging to consecutive scan lines that overlap, are grouped to create beacon features. Each one of the beacon features is likely to belong to one of the four types of beacons (2004 field), so an eight-point asterisk scan is performed around each beacon feature (above and below it) so as to figure out this type. Then, the height and centroid of the pink features are calculated and, according to that height, also the height of the whole beacon is estimated.

Similarly to the goals, the rest of the beacon properties, such as the distance between the beacon and the robot, are calculated using transformations, as well as previously found values (height), the position and size of the goal within the image, the robot' s pose and camera parameters. In order to prove the acquired information' s validity, the beacon centroid' s position with respect to the horizon is checked.

Ball recognition takes place after the two previous types of objects have been recognized within the image. The previously detected edge features are used to compose the outline of the ball, according to a generalisation of Siegel's repeated median line fitting algorithm to circles [20]. These features are combined in all possible triplets in order to create a circle using the equation of the circle:

$$(x-a)^2+(y-b)^2=r^2$$

A circle is determined by each triplet, having as center the intersection point of the middle perpendiculars of the circle chords formed by the triplet. The radius is calculated by taking the median of the various radius estimations, *ri*, using the equation:

$$r=\operatorname*{med}_i r_i$$

The important ball properties are calculated similarly to the goals and beacons, using transformations, as well as previously found values, the position and size of the ball within the image, the robot' s pose and camera parameters. In order to verify that the acquired information is valid, checks concerning the relative position to the horizon, the radius size and the number of orange pixels within the perceived ball area are performed.

4.3 Cerberus

Cerberus is a RoboCup four-legged league team which was based at the Bogazici University of Istanbul in Turkey. This team has been participating in the RoboCup competition since 2001 and has released several interesting approaches to object recognition.

Regarding object recognition, first of all, the regions of interest that may contain an object need to be determined. The Cerberus team applies a region growing algorithm on the raw image [23] . This algorithm processes all the pixels of the image with the intention of finding out regions of the same color and merges them, if they are adjacent to each other, by labeling each pixel with the appropriate region label. These regions are then blobbed to form large regions that are likely to contain a specific field object (candidate regions).

The image, which has been previously processed, should also be rotated according to the robot' s head pose. The Cerberus team has also introduced the *bounding octagon of a region* [23] . This approach intended to reduce the cost of the process, since rotation is an expensive function and octagons are more easily rotated than rectangular boxes.

In order to detect an object within the field, a number of checks is performed on each candidate region. First of all, the neighbouring regions of the current region should meet the specifications of the field colors. For instance, the bottom neighbouring region of the ball should be green. Furthermore, the regions' size is taken into account and compared with the actual size and dimensions' ratio of those of corresponding object. In particular, regarding ball recognition, the following equation is used in order to calculate the radius in cases where the ball is partially present within the image:

$$r = \frac{h}{2} + \frac{s \times s}{8 \times h}$$

In this equation *r* stands for the radius of the ball, *h* is the height and *s* is the width of the candidate region.

4.4 SPQRLegged

The team, whose work helped us the most to develop our own solution was the SPQRLegged of the University of Rome “La Sapienza” in Italy. That's why we are going to analyze their approach to object recognition in detail. We have to mention that the description that follows derives from the team's brief documentation [18] , but mostly from the SPQRLegged code [17] itself.

The vision process begins by scanning the whole image of the camera trying to find regions of the same color. This process consists of two phases: first, the image is scanned for different particles of the same color (eg. blue) and then the particles of the same color (blue) are blobbed in order to create a region where most of the pixels belong to that particular color class (the color class of blue). The latest gives an indication that an object of the corresponding color (the blue goal) may exist at that part of the image. So, when the image process comes to the recognition of this particular object (the blue goal), it is enough to scan only the aforementioned region. This region is a rectangular region within the image frame which is candidate to contain a specific object.

Once the blobbing of candidate regions is finished, each region is scanned for an object according to the color that the region contains. We are going to see this process in detail.

Regarding the goals (2004 field), the colors of interest are the blue and the yellow, so only blue or yellow candidate regions are to be scanned. First of all, it is essential to define the horizon line within the image frame. The horizon is constructed in the same way as it has been described for the German Team. Furthermore, *vertical line* is constructed so as to be perpendicular to the horizon line.

The recognition of goals is implemented in four phases: *calculation of horizontal scan lines, horizontal scan for goals, calculation of vertical scan lines and vertical scan for goals.*

Calculation of horizontal scan lines

During this phase of the process, the horizontal scan lines are constructed parallel to the horizon and scan the rectangular candidate region. First of all, two lines are constructed, *scanline* and *endline* [17] so as to be parallel to the horizon line and perpendicular to the *vertical line.* Those lines' precise position depends on the sign of the angle between the horizon line and the horizontal side of the candidate region. In Figures 4.3 and 4.4 we can see the construction of these lines when the angle between the horizon line and the horizontal side of the candidate region is positive and negative, respectively.

The rest of the horizontal scan lines are going to be constructed between the *scanline* and the *endLine* parallel to them. For each scan line to be constructed, we already know that it should be parallel to the horizon, so we also need the coordinates of one point. These coordinates are the result of increasing the coordinates of *scanline'* s beginning point by a steady step for each new scan line. Each horizontal scan line has to scan only the candidate region. The beginning and ending points of a scan line are defined by the intersection points of the scan line with the rectangular candidate region.

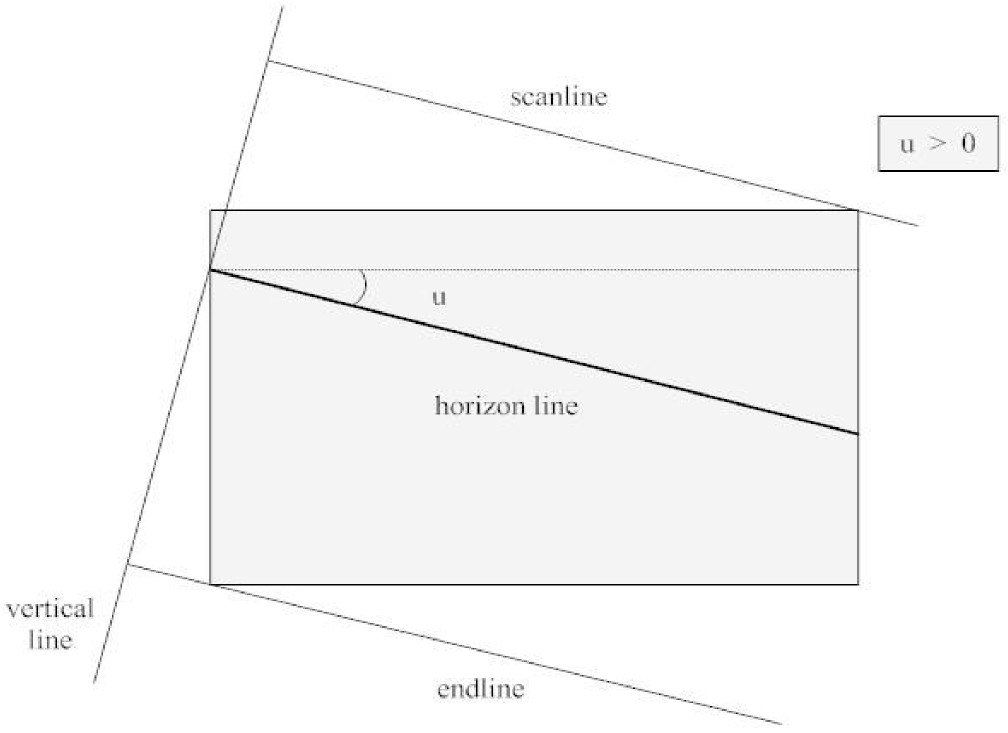


Figure 4.3 – Construction of vertical line, scanline and endline when the angle between the horizon line and the horizontal side of the candidate region is positive.

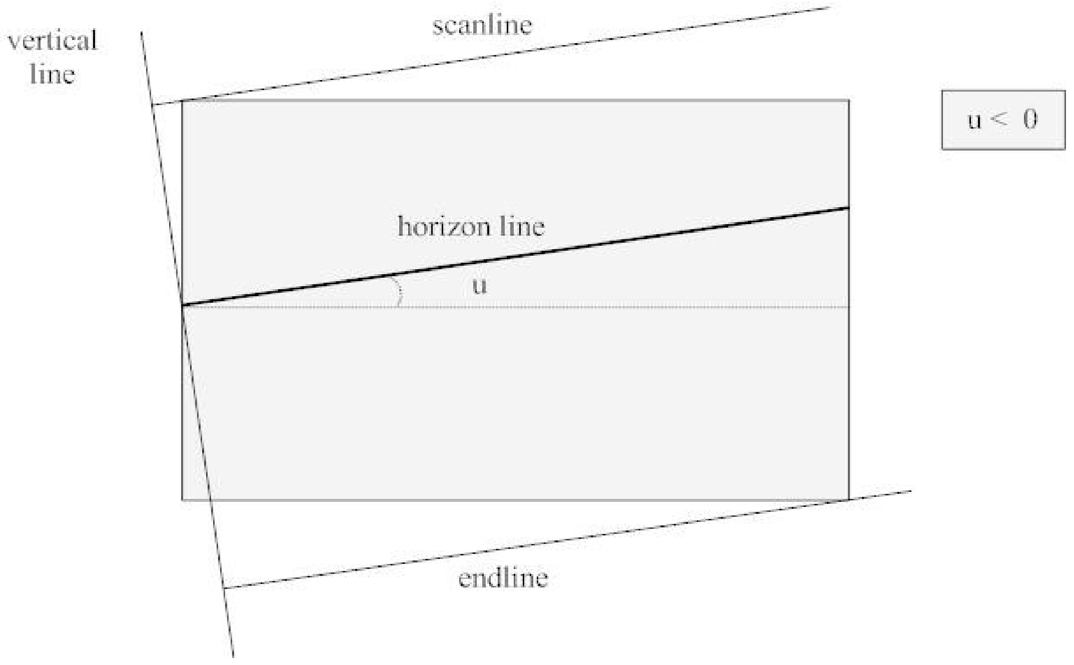


Figure 4.4 – Construction of vertical line, scanline and endline when the angle between the horizon line and the horizontal side of the candidate region is negative.

Horizontal scan for goals

Once they are constructed, the horizontal scan lines can scan the candidate region pixel by pixel and check whether one pixel belongs to a color class of interest or not. Since the colors of interest for the goals are yellow and blue, the following process is performed twice.

A state machine of four states is used in order to find the parts of consecutive pixels of the same color. This machine begins in the state *No Goal Color Found,* where the scan line has not met blue or yellow pixels, yet. Once the scan line finds three consecutive pixels of the same color, the state changes to *Found Begin Of Goal Color* where the machine remains as the scan continues to meet pixels of the same color. If the scan finds three consecutive pixels of a different color, the state of the machine becomes *Found End Of Goal Color.* If six consecutive non-goal color pixels appear the

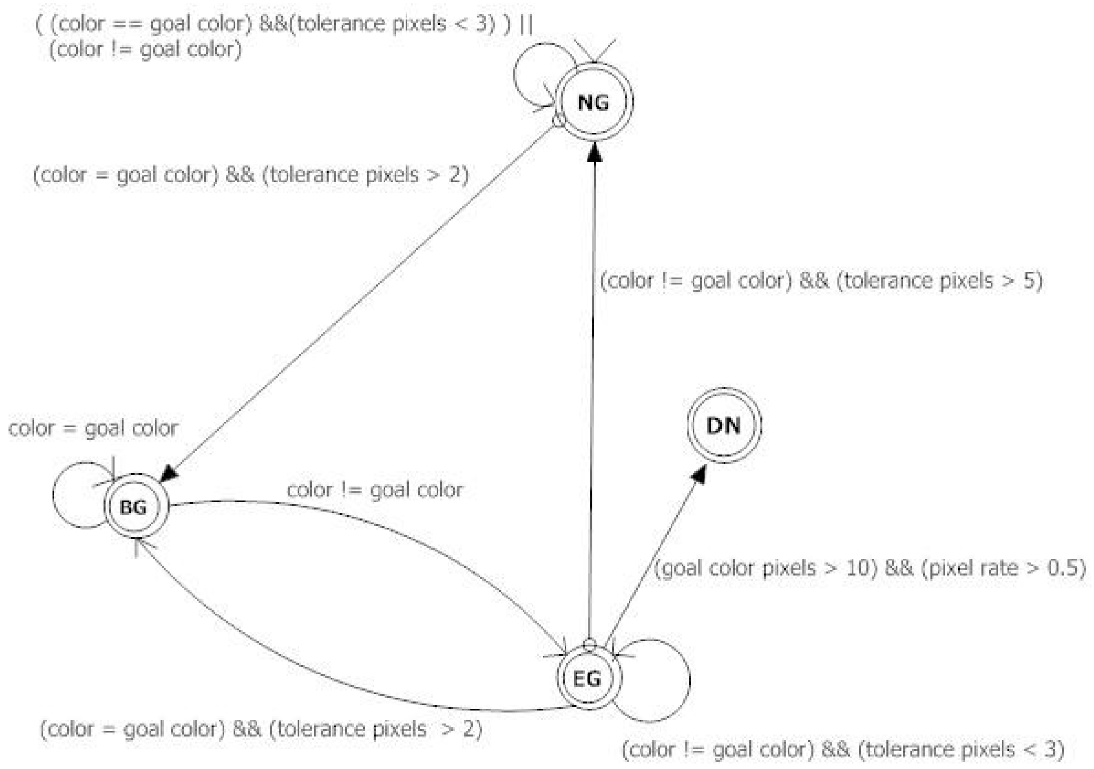

((color == goal color) &&(tolerance pixels < 3)) ||
(color != goal color)
NG
(color = goal color) && (tolerance pixels > 2)
(color != goal color) && (tolerance pixels > 5)
DN
color = goal color
BG
color != goal color
(goal color pixels > 10) && (pixel rate > 0.5)
EG
(color = goal color) && (tolerance pixels > 2)
(color != goal color) && (tolerance pixels < 3)

Figure 4.5 - State Machine used during the horizontal scan for goals (SPQRLegged2006).

machine returns to *No Goal Color Found.* If three consecutive goal color pixels appear, the machine goes to *Found Begin Of Goal Color.* If a small ratio of goal pixels is found, that clearly states the absence of a goal, and the machine goes to the state *Do Nothing.* We can see the structure of this state machine in Figure 4.5. In conclusion, the changing of a state depends on different checks concerning the alternation of the color classes that the pixels belong to, as well as the value of different pixel counters.

The state machine works through the end of a scan line and then continues to the next line until all horizontal scan lines are finished. When a scan line has finished the scan of the whole region, large parts of pixels belonging to the same color class are defined with the use of a function that is called either when the machine leaves the *Found End Of Goal Color* state or when the scan line comes to its end and the machine' s state is still *Found End Of Goal Color*. In order to determine a large part, this function needs to have a range of goal color (i. e. consecutive pixels of goal color interrupted by some non-goal color pixels of tolerance) greater than ten pixels.

All large parts found on that line are added to the structure that determines clusters [17] . This structure collects the large parts of every scan line that finishes the scan and clusters the large parts' edge points with regard to the horizontal axis. We could better see that in Figure 4.6. The clustering of the large parts intends to create the whole indication where the goal is extended within the image frame horizontally.

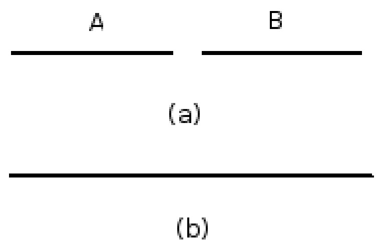


Figure 4.6 – Large parts clustering horizontally: (a) Large parts A and B on the same scan line, before clustering. (b) Large parts A and B after clustering.

If there are only three or four scan lines that contain large parts of goal color, these parts are also clustered into a second structure which collects the parts of the goal that are not occluded by other objects within the field, i.e. the free part of the goal. Then, one *goal indication* [17] is added for each goal cluster; also, the left and right points as well as the center of the goal indication are estimated.

In regard to the determination of the free part of the goal, that only happens when there is one or two goal clusters for the free part of the goal. If there is only one cluster, the center and angle of the free part are estimated. If there are two of them, it is stated that the free part will be only seen completely if the right and left points of both of the clusters are not on the border of the image. The two clusters are then compared in size, so the smaller and larger parts of the free part of the goal are determined. The free part of the goal is only determined in two cases: i) if the right and left points of both small and large parts of the goal's free part are not on border of the image or ii) if the right and left points of the large part of the goal' s free part are on border, whereas those of the small part are not on border of the image.

Finally, in case that the free part of the goal has been determined, needed information about the goals free part is given to the next level of vision, *Obstacles Percept* [16].

Calculation of vertical scan lines

The vertical scan lines are constructed to be perpendicular to the horizon [17] . Their exact position depends on the goal indications that were defined in the previous phase of the process. So, for each goal indication three vertical lines are constructed: one at the center of the goal indication, one at the left side and one at the right side of the indication. As we can see in Figure 4.7, the first vertical line goes through the middle of the goal indication, the second one crosses the middle of the left half of the goal indication, whereas the third scan line crosses the middle of the right half of the goal indication.

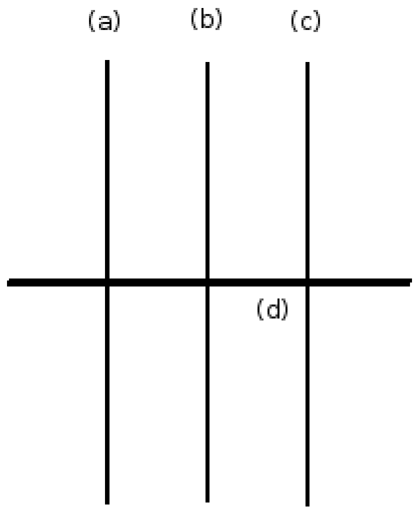


Figure 4.7 – Construction of vertical scan lines on a goal indication: (a) Vertical scan line at the left part of the goal indication. (b) Vertical scan line at the center of the goal indication. (c) Vertical scan line at the right part of the goal indication. (d) One of the horizontal goal indications.

In case that the free part of the goal has been defined during the previous phase, an extra vertical scan line is constructed for each goal cluster for the free part.

Vertical scan for goals

During this phase, the process is quite the same as in the *horizontal scan for goals.* However, the information here derives from the vertical lines scan for goals.

The state machine that is used for this phase (Figure 4.8) is similar to the one we described for the *horizontal scan for goals.* The only differences are that here the *Do Nothing* state is replaced by the *Determined Goal* [17] state, with the corresponding change of the transitive condition.

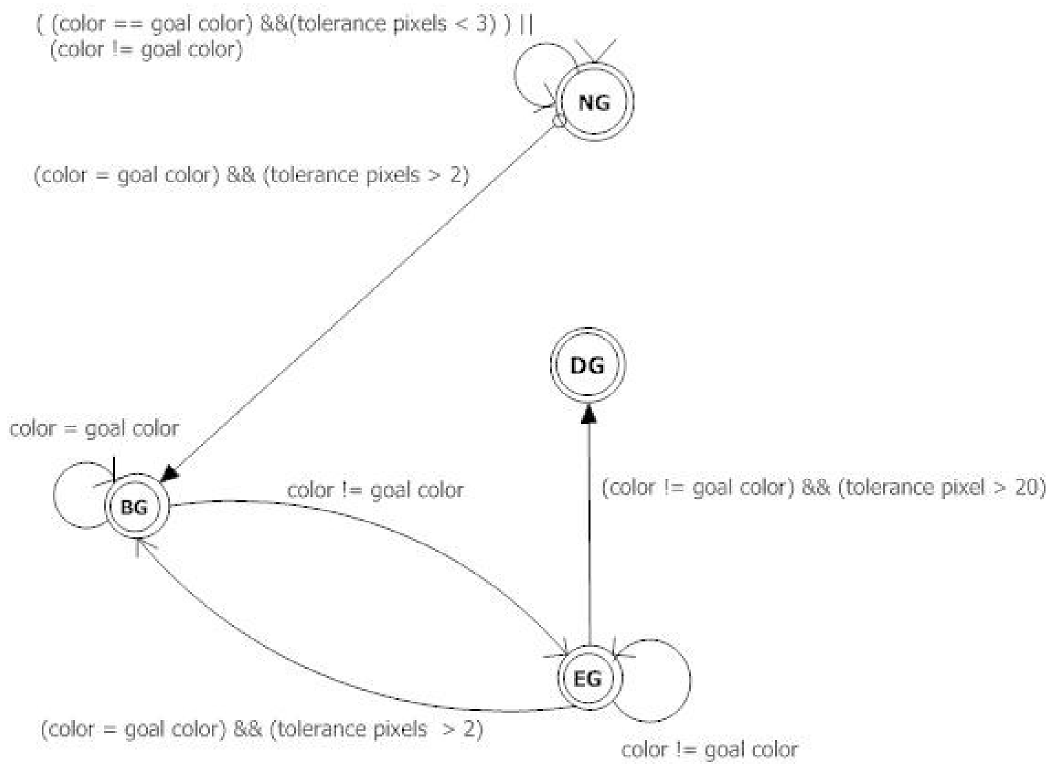

((color == goal color) &&(tolerance pixels < 3)) ||
(color != goal color)
NG
(color = goal color) && (tolerance pixels > 2)
DG
color = goal color
BG
color != goal color
(color != goal color) && (tolerance pixel > 20)
EG
(color = goal color) && (tolerance pixels > 2)
color != goal color

Figure 4.8 - State Machine used during the vertical scan for goals (SPQRLegged2006).

During the vertical scan, the corresponding state machine also intends to find large parts of goal color. Once the machine has worked for every scan line and if there are large parts found from vertical scan, there is a check for white and pink pixels performed in order to see if the scanned area is actually a beacon instead of a goal.

The large parts obtained during this phase are saved within an array, so, when the vertical scan is finished we have information about both the horizontal and the vertical extension of the goal within the image, by the horizontal goal indication and the corresponding array of large parts in vertical. That is to say that for every *goal indication* [17] the information that we have about the goal can be depicted as in Figure 4.9.

If no large parts are found during the vertical scan, then the free part for the goal that has probably been found during the corresponding horizontal scan on that scan line is canceled.

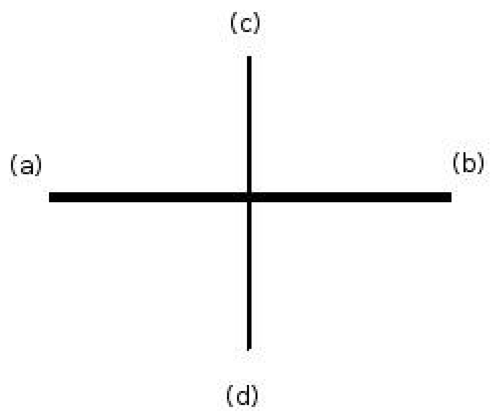


Figure 4.9 – A goal indication after both horizontal and vertical scan: (a) Left of horizontal goal indication, (b) right of horizontal goal indication, (c) top point of the first large part found on that vertical scan line and (d) bottom point of the last large part found on that vertical scan line.

After both scanning procedures, all goal indications are added together to create the goal with the use of the *Conditional Boundary* [17] . This structure is defined through a set of axes which are first initialized and then the edge points of the goal indications are added until the whole goal takes shape. For example, looking at Figure 4.10, if the point P is an edge point of a goal indication and needs to be added, then its x coordinate (px) will be

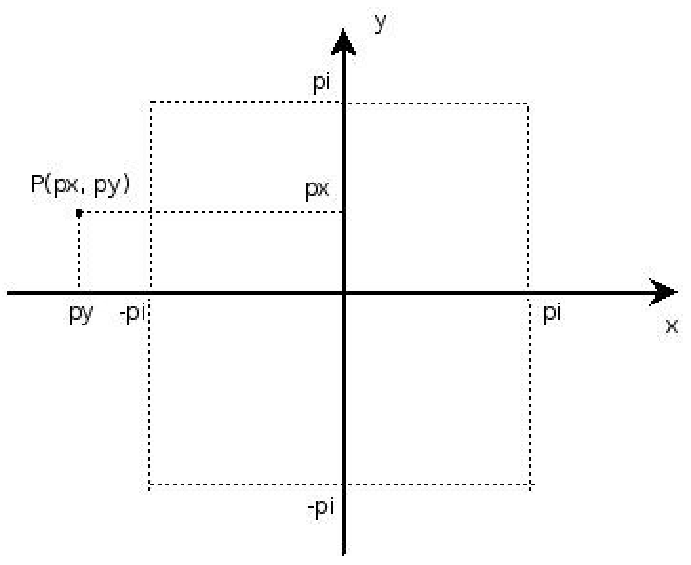


Figure 4.10 – Conditional boundary. The point P is the edge point of a goal indication and needs to be included to the boundary.

added to -pi. Its y coordinate is already smaller than the existing boundary' s y coordinate so it remains equal to pi. So, the resulting boundary is better shown in Figure 4.11.

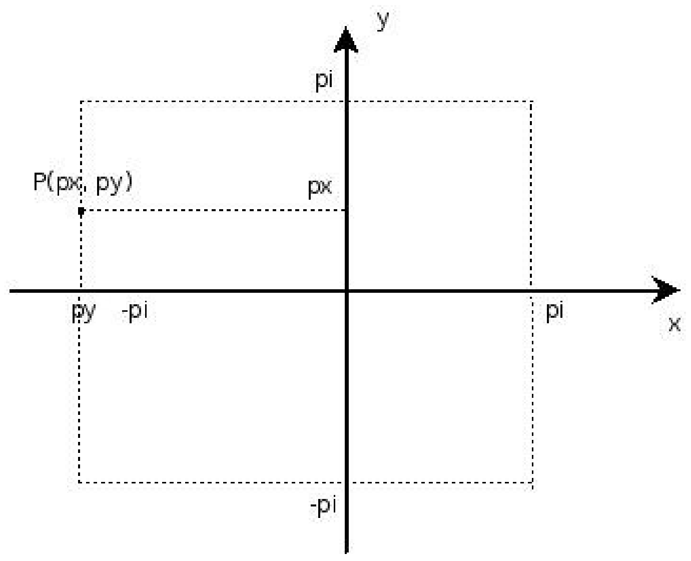


Figure 4.11 - Conditional boundary. The point P has been included to the boundary.

Once all the goal indications are added to the boundary, the box that contains the goal within the image is complete and the goal corners coordinates can be easily found by the boundary' s limit coordinates. This information is then sent to the next step of the vision, *landmarksPerept* and *obstaclesPercept* [16].

Concerning the beacons and the ball, SPQRLegged has been using the corresponding German Team' s modules that we have already described in previous section. The only difference is that the SPQRLegged applies these modules only on the candidate regions and not to the whole image frame as the German Team does.

4.5 Observations and Conclusions

So far we have described the work of four different teams, related to visual object recognition for the RoboCup 2006 four-legged league. We have focused more on the SPQRLegged and the German Team. Their work has been a starting point for us and helped us address our own problem. However, there are also some issues that need to be further discussed and improved.

As mentioned earlier, regarding the image processing, the SPQRLegged first finds the candidate regions to be scanned and then the scanning procedure takes place only in regard to these regions and not the whole image. On the other hand, the German Team uses the whole image frame but the grids of scan lines that they use are extended mostly around the horizon, since it is more likely to find the objects of interest there, as they state.

These approaches have worked well enough for the previous goal' s compact appearance. But, concerning the new 2007 goals, we realised that some problems may occur. The SPQRLegged approach is likely to find four separate candidate regions for the goal (one for each of the two horizontal and the two vertical goal posts), so the process that will follow will not make any sense. Also, the German Team' s approach was likely to confuse the goals with the beacons, because the vertical goal posts were partly similar with the new beacons.

In addition, we were concerned about the number of scan lines that should be used. The SPQRLegged has been using ten horizontal scan lines within the candidate region, whereas the German Team uses fourteen of those lines for the entire image. Both teams use vertical scan lines only when goal indications have been found during the horizontal scan. This, of course, reduces the image processing time a lot, because the vertical scan can sometimes be avoided (for example, in cases that there is really no indication of an object of interest within the image).

Nevertheless, the goals' new appearance make it more difficult to recognize, because, as we said, it is not compact. So, we expect that the vertical goal posts are more likely to be reached by horizontal scan lines whereas the horizontal posts would be mostly crossed by vertical scan lines. Furthermore, the horizontal posts occupy more area within the image frame than the vertical ones, so, in comparison it is even more important for them to be correctly defined. Thus, the increase of the vertical scan lines becomes essential.

Therefore, we decided to increase the number of both the horizontal and the vertical lines, trying to reach a trade-off between correct object recognition and satisfying image processing time, as well.

Moreover, tolerance in pixels that do not belong in the desired color class had to be reduced in cases such as the goal recognition, where the regions of homogeneous color are now smaller. Last, counters for pink pixels had to be removed since the corresponding color class no longer exists within the field.

We are about to describe such changes, as well as our whole approach in detail in the following chapter.

5 Our Approach

In this chapter, the solution which has been given to the problem of object recognition within the 2007 RoboCup competition field is going to be presented. We are going to describe three approaches, one for each type of field object (goals, beacons and ball), and indicate both the similarities and the essential differences among them.

The snapshots that are used from this chapter and on are taken while Kouretes Image Viewer is running. Kouretes Image Viewer is a tool used to run and debug the source code. It is a variation of the corresponding SPQRLegged 2006 tool [17] .

Taking a look at Snapshot 5.1, one can note that the first window shows the raw image, as it comes from the robot' s camera. The second window shows the color-segmented picture as well as whatever we would like to print for debugging. The second window is the image that is processed in order to perform the object recognition. The horizon line is also drawn in this window. Every pixel of the image is classified into a color class. This occurs according to a *Color Table ,* which is also loaded and determines the coordinate limitation of every color, following the HSV color space (fourth window) [18] . The third one is the perception window, where the recognized objects are depicted as well as the horizon line. We are going to focus on such objects and the specific window in following snapshots.

5.1 Goal Recognition

In this section, we are focusing on how the robot is expected to recognize the new goals (RoboCup 2007) correctly and transmit the corresponding information to the next level of the robot' s cognition.

In order to achieve that, we decided to scan the whole image frame, both horizontally and vertically, with respect to the horizon so as to get information about the color and the position of pixels of interest within the image. Then this information is further processed in order to yield the final result.

First, the goal colors are defined according to the team the robot belongs to. Then, the *horizon line* and *the vertical line* are constructed in the same way as it has been described for the German Team. These lines are the base for constructing the horizontal and the vertical scan lines.

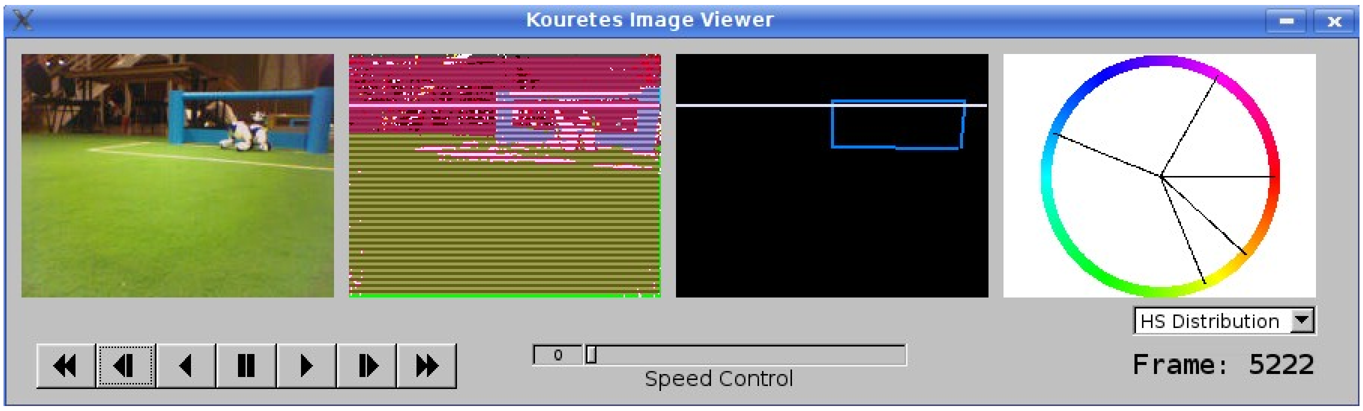
The whole process of goal recognition consists of five different phases: *calculation of horizontal scan lines, horizontal scan for goals, calculation of vertical scan lines, vertical scan for goals* and *goal determination.*

Calculation of horizontal scan lines

During this phase of the process, the horizontal scan lines are constructed. They are parallel to the horizon and cover the whole image frame. We use 40 horizontal scan lines which are constructed to be equally distanced from each other.

Apart from the scanning area and the number of the scan lines, the construction process is the same as the corresponding one of the SPQRLegged team [17] . Two lines are constructed, *scanline* and *endline* which are parallel to the horizon line and perpendicular to the *vertical line.* Those lines' precise position depends on the sign of the angle between the horizon line and the horizontal side of the image frame. In Figures 4.2 and 4.3 (see Chapter 4) we can see the construction of these lines keeping in mind that in our case the rectangular area of the Figures is the whole image frame instead of the SPQRLegged candidate region.

As before, the rest of the horizontal scan lines are going to be constructed between the *scanline* and the *endLine* and will be parallel to them. In Snapshot 5.1 we can see how the horizontal scan lines are constructed on the image frame. When the aforementioned angle is equal to zero, the horizontal scan lines are also parallel to the horizontal side of the image.



Snapshot 5.1 – Kouretes Image Viewer: in the second window we can see the horizontal scan lines which are parallel to the horizon. The angle between the horizon line and the horizontal side of the image frame is zero.

Horizontal scan for goals

During this phase, the horizontal scan lines scan the image pixel by pixel and check whether one pixel belongs to a color class of interest or not. Since the colors of interest for the goals are yellow and blue, the process that is described below takes place once for each color.

The state machine that we use consists of two states, *No Goal Color Found* (NG) and *Goal Color Found* (G), and aims to find consecutive pixels of the same color. This machine begins from the state *No Goal Color Found,* where the scan line either has not met blue or yellow pixels yet or, the ones found are less than the *horizontal threshold* in sequence. Once the scan line finds as many consecutive pixels of the same color as the threshold, the state changes to *Goal Color Found,* where the machine remains as long as the scan continues to meet pixels of the same color or less than the threshold consecutive pixels of non-goal color. If the scan hits the horizontal threshold in consecutive pixels of a different color, the state comes back to *No Goal Color Found* state. We can better see the structure of this state machine in Figure 5.1.

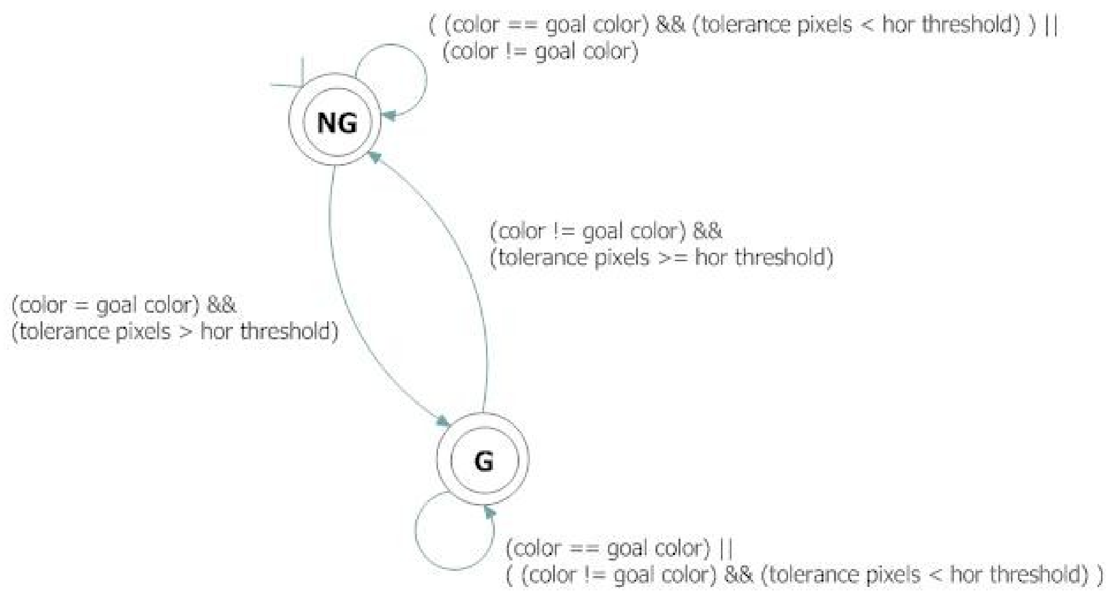
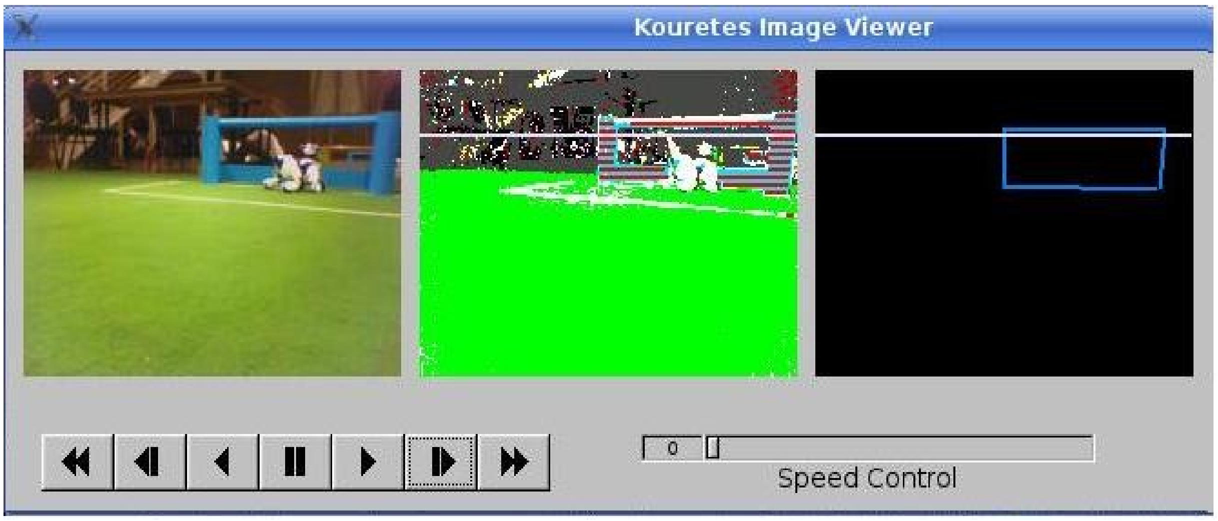

((color == goal color) && (tolerance pixels < hor threshold)) ||
(color != goal color)
NG
(color != goal color) &&
(tolerance pixels >= hor threshold)
(color = goal color) &&
(tolerance pixels > hor threshold)
G
(color == goal color) ||
((color != goal color) && (tolerance pixels < hor threshold))

Figure 5.1 – State machine for the horizontal scan for goals (Kouretes2007).

The state machine works through the end of each scan line until all pixels are processed. Once the scan line has finished the scan of the whole region, large parts of pixels belonging to the same color class are defined with the use of a function that is called either when the machine leaves the *Goal Color Found* state or when the scan line comes to its end and the machine' s state is still *Goal Color Found*. In order to determine a large part, this function takes a range of goal color pixels (that is, consecutive pixels of goal color possibly interrupted by some non-goal color pixels) marking the entrance and the exit to the state *Goal color Found*. We can see how the large parts of the goal are forming during the horizontal scan in Snapshot 5.2.

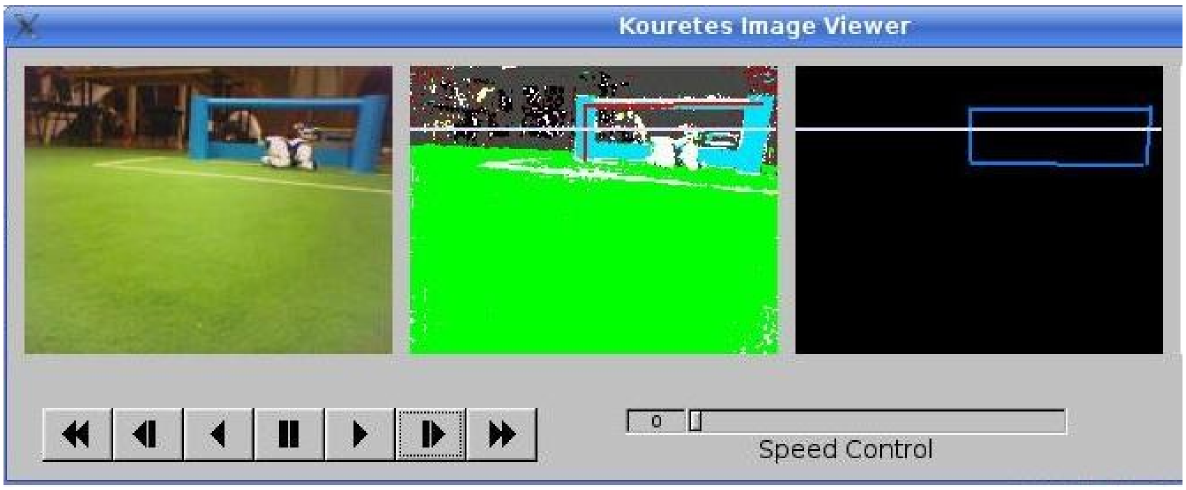


Snapshot 5.2 – Second window: the large parts of the blue goal formed by horizontal scan, shown in red color.

All large parts found along the horizontal scan lines are added to a data structure that determines clusters. The clustering of large parts occurs in the same way as described for the SPQRLegged team (see Figure 4.5).

Having formed the goal clusters from the large parts, we now create two *horizontal goal indications*: one indication to show the extent of the goal along the horizontal scan lines (*horizontal goal indication along*) and one showing extent of the goal across the horizontal scan lines (*horizontal goal indication across*). As one could expect, the *horizontal goal indication along* begins at leftmost point of the cluster and ends at the rightmost one. On the other hand, the *horizontal goal indication across* begins and ends at the most extreme scan lines containing large parts. For example, it could extend from the second till the tenth horizontal scan line.

Both the *horizontal goal indication along* and the *horizontal goal indication across* the horizontal scan lines are depicted in Snapshot 5.3.



Snapshot 5.3 – Second window: the horizontal goal indications (along and across) of the blue goal formed by horizontal scan, shown in red color.

The large parts, are also added to another data structure, the *horizontal histogram.* This is a structure where all the large parts are stacked together on top of each other to expose their density along the horizontal axis. The histogram is based on a discretization of the horizontal axis into 270 intervals covering the entire viewing angle. Each interval' s value is incremented every time this particular interval belongs to a large part (it lies between the starting and the ending point). When all the points are inserted, the histogram can give us information about the part of the image where most of the large parts exist.

Regarding the horizontal scan, if two peaks are found in the histogram, that states the existence of the two vertical goal posts and the horizontal indication is then extended between these two peaks. If only one peak is found in the histogram, then there is a single vertical goal post within the image. In that case, it is checked whether the goal color parts extend horizontally until the border of the image and the horizontal goal indication' s other limit would, then, be the corresponding image border.

The way in which the horizontal histogram is formed, is depicted in Figure 5.2.

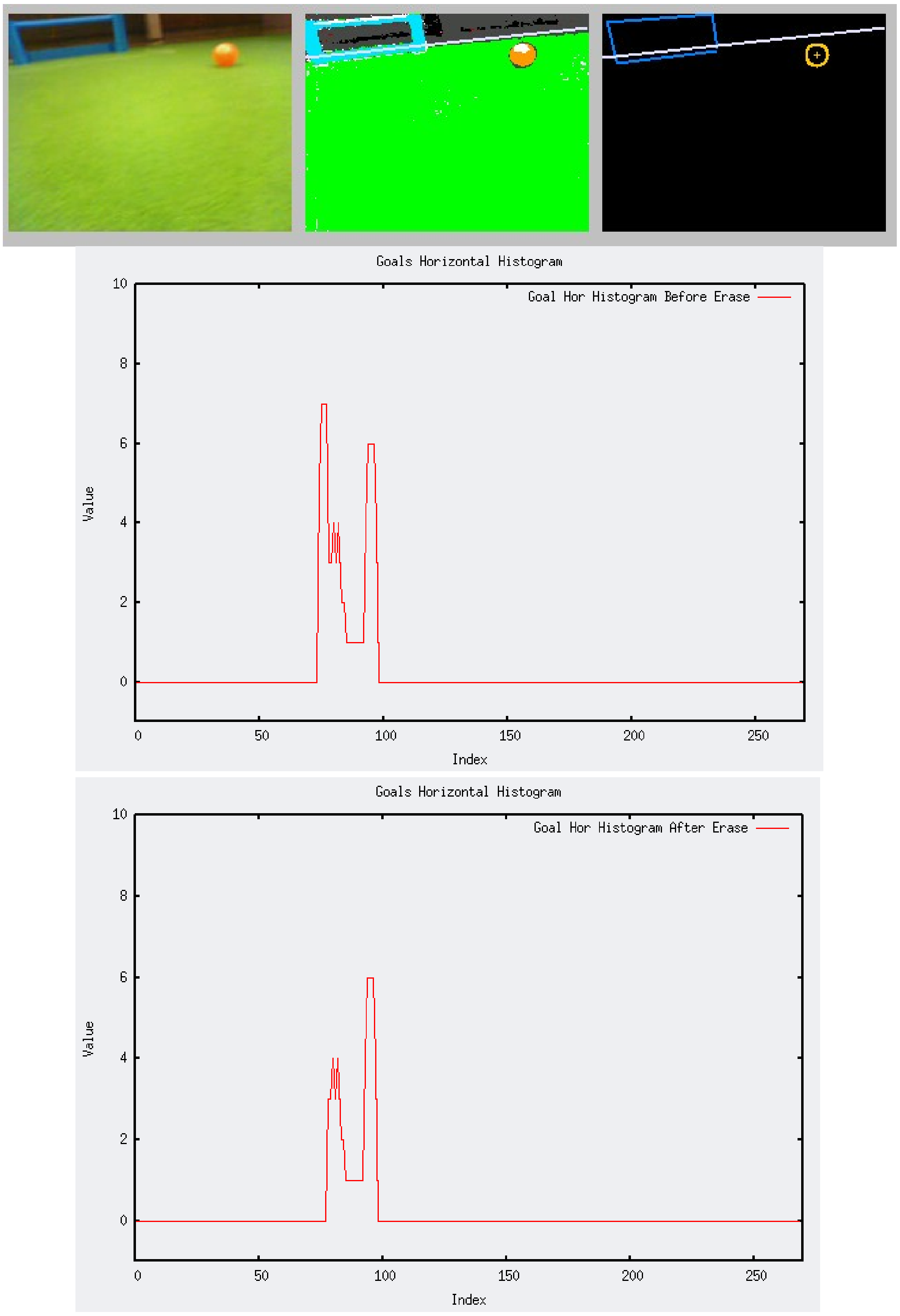


Figure 5.2 – Goal horizontal histogram: the first diagram shows how the histogram' s value converts with the large parts' edge points (index) during the scan for goals of the

color segmented image (second window of Kouretes image viewer of top image). Note the two peaks of the histogram, which refer to the two vertical goal posts. The second histogram is formed after the first peak is erased from the histogram, so as to find the second maximum of the histogram, which refers to the other vertical goal post.

All information acquired by the horizontal scan of the image is saved and will be further processed during the *goal determination* phase of the process. We are now ready to describe the next step of the process, the *calculation of vertical scan lines.*

Calculation of vertical scan lines

The vertical scan lines are constructed so as to be perpendicular to the horizon and cover the whole image frame. Their construction is similar to the one of the horizontal scan lines. We use 50 vertical scan lines which are constructed to be equally distanced from each other.

The construction procedure begins. Two lines are constructed, *scanline* and *endline* which are parallel to the *vertical line* and perpendicular to the horizon line*.* Those lines' precise position depends on the sign of the angle between the horizon line and the horizontal side of the image frame. In Figures 5.3 and 5.4 we can see the construction of these lines onto the whole image frame.

The rest of the vertical scan lines are going to be constructed between the *scanline* and the *endLine* and will be parallel to them. In Snapshot 5.4 we can see how the vertical scan lines are constructed on the image frame. When the aforementioned angle is equal to zero, the vertical scan lines are also parallel to the vertical side of the image frame.

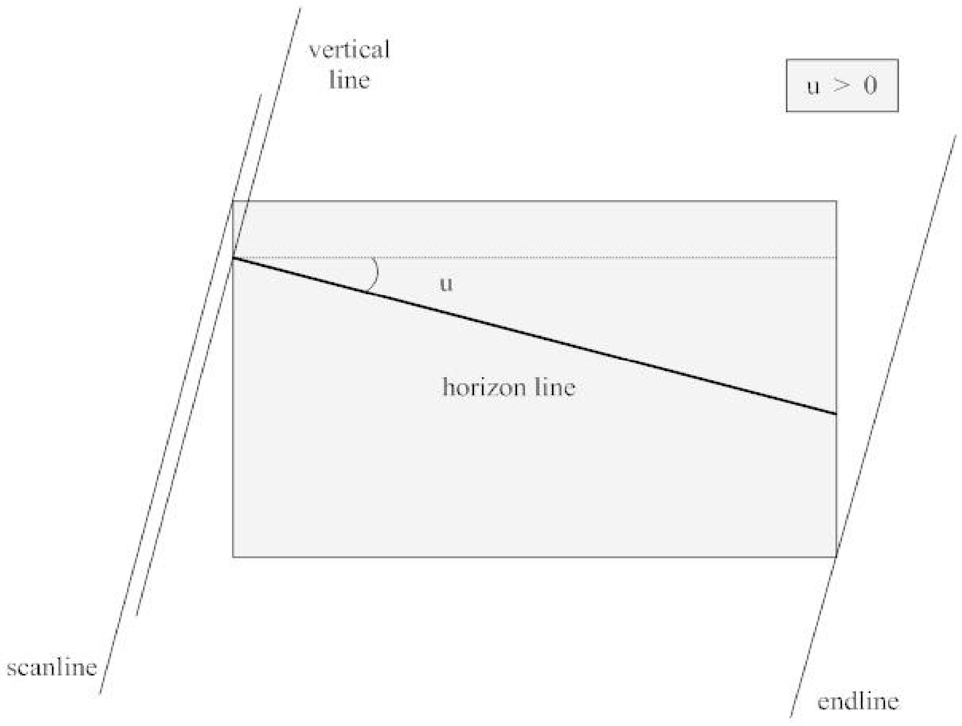


Figure 5.3 – Construction of vertical line, scanline and endline when the angle between the horizon line and the horizontal side of the image frame is positive.

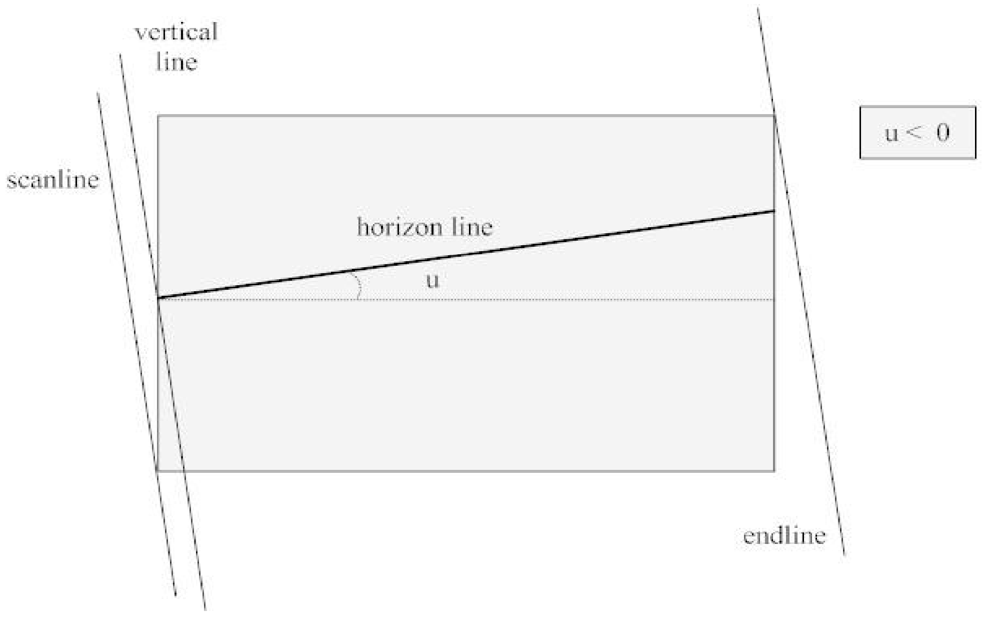
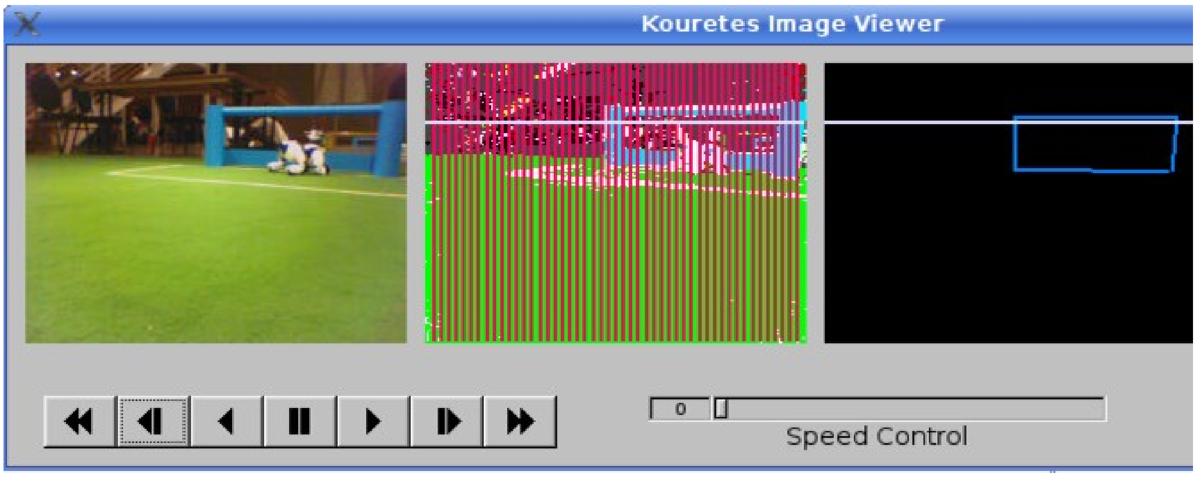


Figure 5.4 – Construction of vertical line, scanline and endline when the angle between the horizon line and the horizontal side of the image frame is negative.



Snapshot 5.4 – In the second window we can see the vertical scan lines which are perpendicular to the horizon. The angle between the horizon line and the horizontal side of the image frame is positive.

The vertical scan lines are going to scan the whole image according to what is described in the next phase.

Vertical scan for goals

The vertical scan for goals uses the same state machine as the horizontal one. The only difference is that the tolerance in goal or non-goal color here is the *vertical threshold,* which is smaller than the horizontal one. This choice makes sense, since the upper horizontal goal post is thinner than the vertical goal posts.

The state machine for the vertical of the image is shown in Figure 5.5.

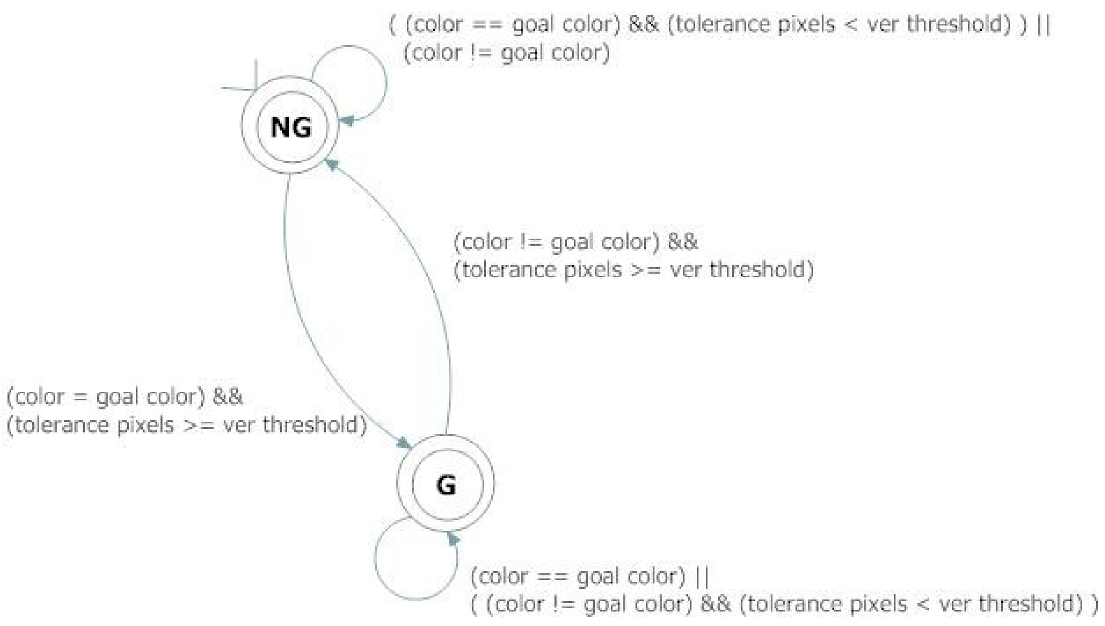
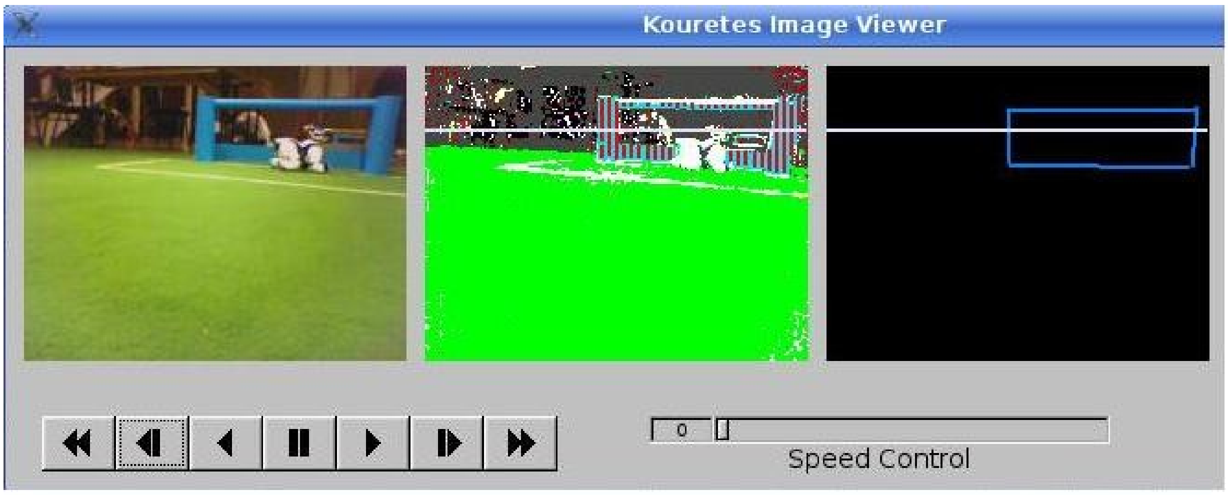

((color == goal color) && (tolerance pixels < ver threshold)) ||
(color != goal color)
NG
(color != goal color) &&
(tolerance pixels >= ver threshold)
(color = goal color) &&
(tolerance pixels >= ver threshold)
G
(color == goal color) ||
((color != goal color) && (tolerance pixels < ver threshold))

Figure 5.5 – State machine for the vertical scan for goals (Kouretes2007).

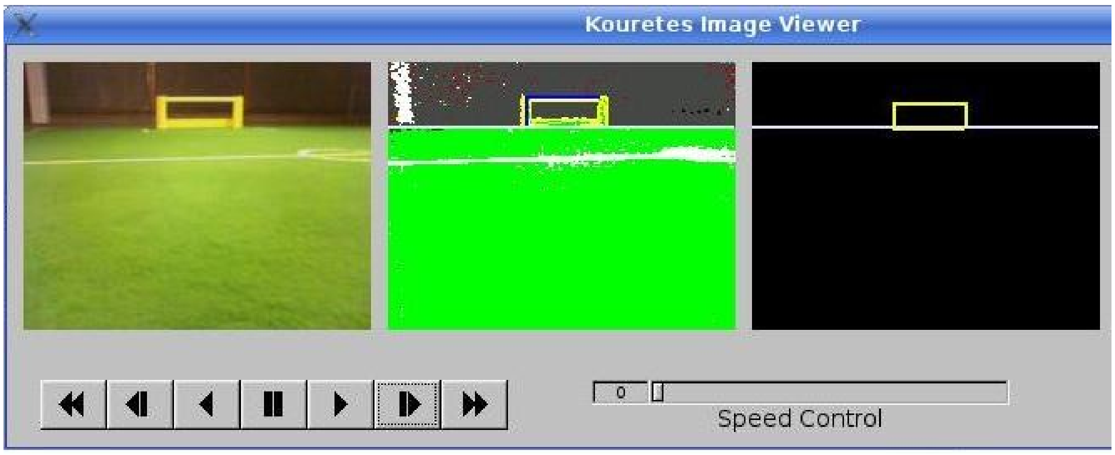
As the state machine runs for a particular scan line, the large parts are determined using the same function used during the horizontal scan. This function needs to have a range of goal color pixels greater than the *vertical large-parts threshold* in order to determine a large part. We can see how the large parts of the goal are forming during the vertical scan in Snapshot 5.5.



Snapshot 5.5 – Second window: the large parts of the blue goal are forming after the vertical scan, shown in red color.

The procedure continues similarly to the horizontal scan. The large parts found along a particular line are added to the structure that determines clusters and two *vertical goal indications* are created: one indication to show how the goal extends within the image along with the vertical scan lines (*vertical goal indication along*) and one showing the goal extension across the vertical scan lines (*vertical goal indication across*) within the image frame.

Both the vertical goal indication along and the vertical goal indication across the vertical scan lines are depicted in Snapshot 5.6.



Snapshot 5.6 – Second window: the vertical goal indications (along and across) of the yellow goal formed by vertical scan, shown in blue color.

In addition, the *vertical histogram* is created in accordance with the horizontal one. The large parts, are added to another data structure, the *vertical histogram.* This is a structure where all the large parts are stacked together on top of each other to expose their density along the vertical axis. The histogram is based on a discretization of the vertical axis into 90 intervals covering the entire viewing angle. Each interval' s value is incremented every time this particular interval belongs to a large part (it lies between the starting and the ending point). When all the points are inserted, the histogram can give us information about the part of the image where most of the large parts exist.

Regarding the vertical scan, if two peaks are found in the histogram, that states the existence of the two horizontal goal posts and the vertical indication is then extended between these two peaks. If only one peak is found in the histogram, then there is a single horizontal goal post within the image. In that case, it is checked whether the goal color parts extend vertically until the border of the image and the vertical goal indication' s other limit would, then, be the corresponding image border.

The way in which the horizontal histogram is formed, is depicted in Figure 5.6.

All information acquired by the vertical scan of the image is saved and is further processed, combined with the corresponding information out of the horizontal scan, during the *goal determination* phase of the process.

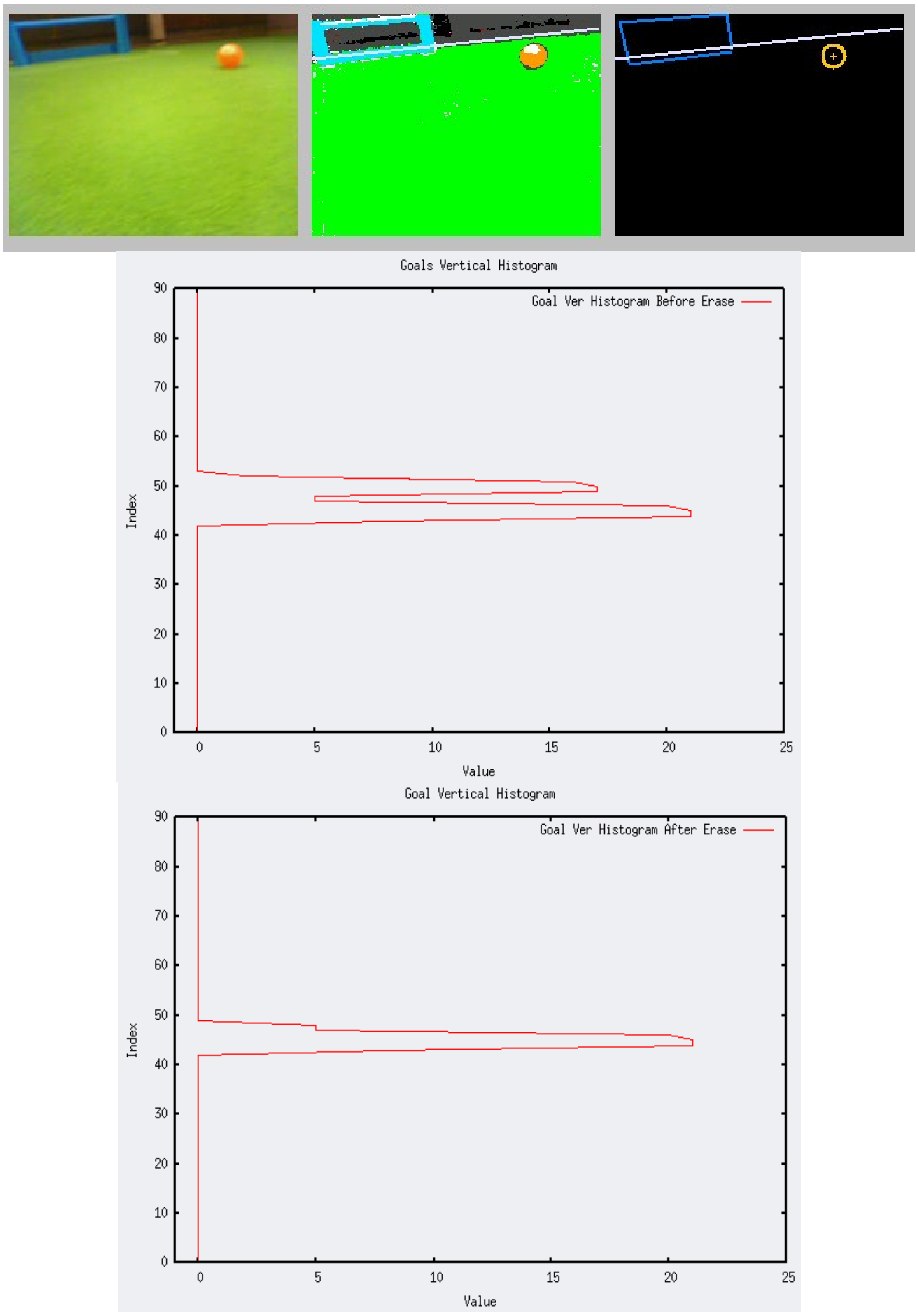


Figure 5.6 – Goal vertical histogram: the first diagram shows how the histogram' s value converts with the large parts' edge points (index) during the scan for goals of the color segmented image (second window of Kouretes image viewer, top image). Note the two peaks of the histogram, which refer to the two horizontal goal posts. The

second diagram is formed after the first peak is erased from the histogram, so as to find the second maximum of the histogram, which refers to the other horizontal goal post.

Goal determination

At this moment, we have information about four different indications for the existence of a goal: two of them come from the horizontal scan and the other two from the vertical scan. We are going to see how we decide which information is preferable to use and how the final goal is formed, as well as the different checks that the acquired information should successfully pass in order to refer to a recognized goal.

First of all, we handle the cases where one goal indication' s maximum and minimum points, acquired by a specific scan, are identical. Then, under some circumstances, one of these points is replaced by the corresponding one acquired by the other scan. For example, if the horizontal scan has resulted into a *horizontal goal indication* along whose minimum and maximum points are identical, but their corresponding (vertical goal indication across) minimum and maximum points, acquired by the vertical scan, are not identical, then one of the *horizontal goal indication* along minimum or maximum point is replaced by the corresponding from the vertical scan, so they are not identical any more.

After this replacement has occurred and in order to form the whole goal, the *goal boundary* is formed. This is a *Conditional Boundary* [17] (see also Figures 4.9 and 4.10) whose function has already been described for the SPQRLegged2006. In our case, horizontal goal indication's along and vertical indication's across edge points are added to the boundary (see section 4.4) in order to form the horizontal dimension of the boundary, whereas horizontal goal indication's across and vertical indication's along ones are added to form the vertical dimension of the boundary.

After the boundary has been formed and we know how the goal is extended within the image, we also want to pass this information through several filters in order to reject the formed goal if it doesn't fit some essential specifications.

First, we check the confidence that the formed goal information can provide us. If all four goal indications (horizontal-along, horizontal-across, vertical-along, vertical-across) have been created through information that derives from a number of scan lines that is below a defined threshold, then the goal boundary that has been formed through these indications is rejected.

Then, we verify whether the goal boundary' s dimensions result in a reasonable perceived distance. This is achieved by taking into account the distance in which the robot sees the perceived goal boundary. This distance can be calculated through some geometrical transformations that use both the perceived and the actual dimensions of the goal. Thus, if this distance exceeds some specific threshold (negative or larger than the field size), we cannot use this goal boundary as information to form the goal, so it is again rejected.

Another check that is performed, refers to the ratio between the two goal dimensions which should be stable. Since we know the actual goal dimensions, we can calculate the ratio between them and compare this ratio to the goal boundary' s dimension ratio. Attention is paid also to the fact that the goal can be occluded by the image frame borders. In that case, the dimension ratio is not real, since a part of the goal is not present within the frame. So, the dimension ratio filter is only used when at least one side of both the horizontal and the vertical indication is not on the border of the image. Under these circumstances, if the dimension ratio abstains a lot more than a specific tolerance from the actual ratio, then the found goal boundary is rejected.

If the found goal boundary has successfully passed all the aforementioned filters, that means that we can consider it as a perceived goal. So, information about the goal color, its center and its angle, with respect to the robot, can be easily calculated and sent to the next step of cognition, *landmarks percept* and *obstacles percept* [16].

5.2 Beacon Recognition

In this section, we are focusing on how the robot recognizes the new beacons (RoboCup 2007) correctly and transmits the corresponding information to the next level of the robot' s cognition. As we have seen in the second chapter, the beacons have been reduced from four to two and their color combinations have also changed, for the 2007 RoboCup competition.

As described for the goal recognition, we decided to scan the whole image frame, both horizontally and vertically with respect to the horizon, so as to get more accurate information about the color and the position of pixels of interest within the image. Then this information is further processed in order to yield the final result.

First, the *horizon line* and the *vertical line* are constructed in the same way as it has been described for the German Team. These lines are the base for constructing the horizontal and vertical scan lines.

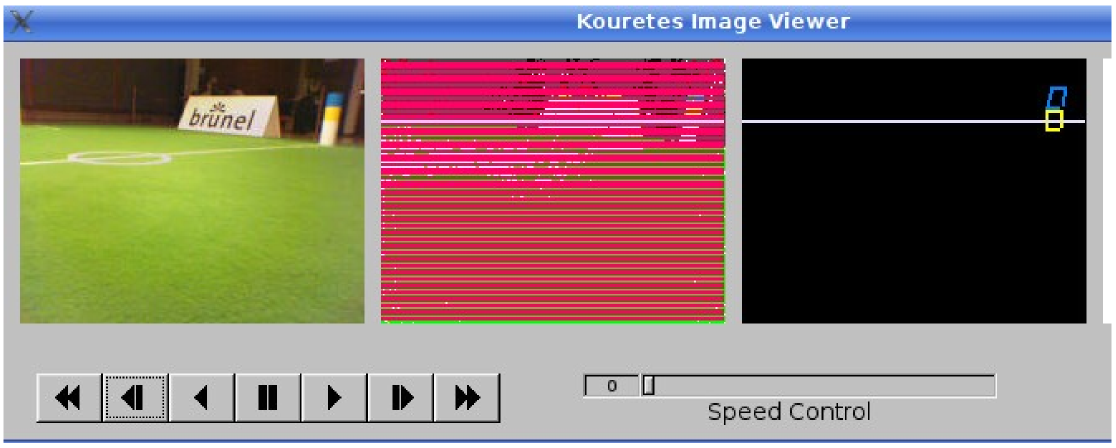
The whole process of beacon recognition consists of five different phases: *calculation of horizontal scan lines, horizontal scan for beacons, calculation of vertical scan lines, vertical scan for beacons* and *beacon determination.*

Calculation of horizontal scan lines

During this phase of the process, the horizontal scan lines are constructed. They are parallel to the horizon and cover the whole image frame. We use 60 horizontal scan lines which are constructed to be equally distanced from each other. This relatively big number of horizontal scan lines is used in order to reduce the vertical ones as we will see in the corresponding section.

The process is exactly the same as the corresponding one for the goals (see also section 5.1). Two lines are constructed, *scanline* and *endline* which are parallel to the horizon line and perpendicular to the *vertical line.* Those lines' precise position depends on the sign of the angle between the horizon line and the horizontal side of the image frame. In Figures 4.2 and 4.3 we can see the construction of these lines keeping in mind that in our case the rectangular area of the Figures is the whole image frame instead of the SPQRLegged candidate region.

As before, the rest of the horizontal scan lines are going to be constructed between the *scanline* and the *endLine* and will be parallel to them. In Snapshot 5.7, we can see how the horizontal scan lines are constructed on the image frame. When the aforementioned angle is equal to zero, the horizontal scan lines are also parallel to the horizontal side of the image.



Snapshot 5.7 – Kouretes Image Viewer: in the second window we can see the horizontal scan lines used for beacons. The angle between the horizon line and the horizontal side of the image frame is, here, equal to zero.

Horizontal scan for beacons

This phase of the process is quite similar to the corresponding phase for the goals. The horizontal scan lines scan the image pixel by pixel and check whether one pixel belongs to a color class of interest or not. The colors of interest for the beacons are, again, yellow and blue, so the process that is described below takes place twice, once for each color.

The state machine that we use consists of two states, *No Beacon Color Found* (NB) and *Beacon Color Found* (B)*,* and aims to find consecutive pixels of the same color. This machine begins from the state *No Beacon Color Found,* where the scan line either has not met blue or yellow pixels yet or, the ones found are less than the *horizontal threshold*, in sequence. Once the scan line finds as many consecutive pixels of the same color as the specific threshold, the state changes to *Beacon Color Found,* where the machine remains as long as the scan continues to meet pixels of the same color or the consecutive pixels of non-beacon color met are less than the threshold. If the scan finds consecutive pixels of a different color equal in number to the threshold, the state comes back to *No Beacon Color Found* state. We can better see the structure of this state machine in Figure 5.7.

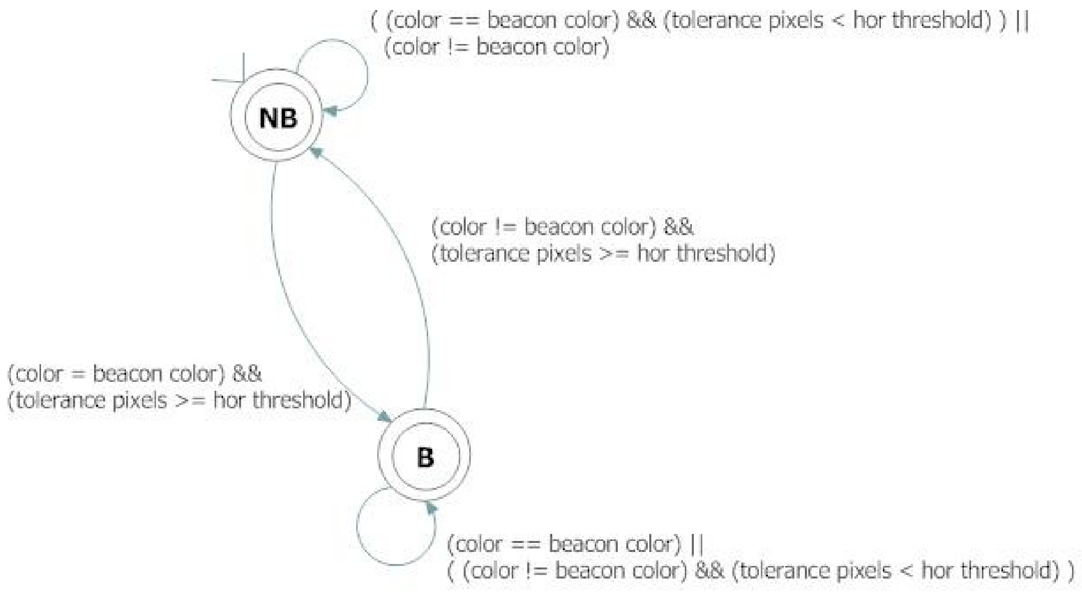
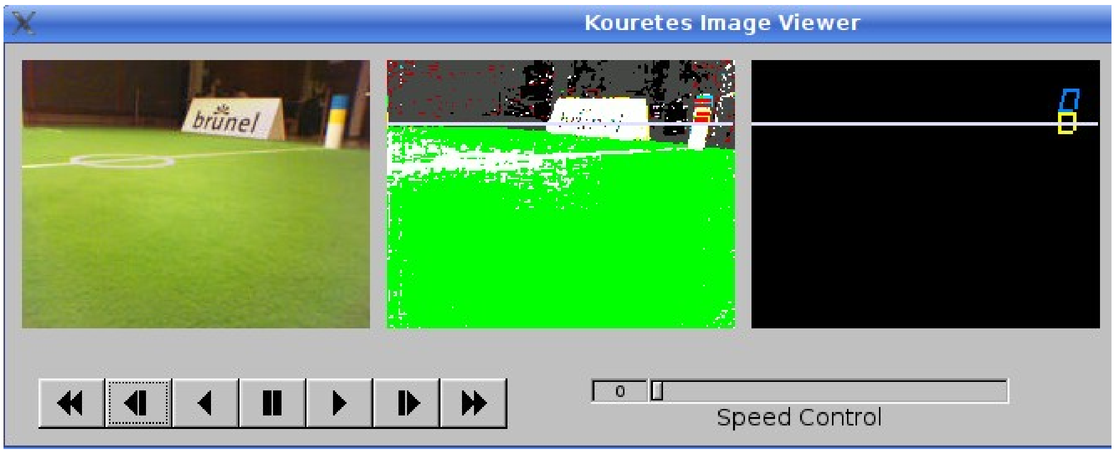

((color == beacon color) && (tolerance pixels < hor threshold)) ||
(color != beacon color)
NB
(color != beacon color) &&
(tolerance pixels >= hor threshold)
(color = beacon color) &&
(tolerance pixels >= hor threshold)
B
(color == beacon color) ||
((color != beacon color) && (tolerance pixels < hor threshold))

Figure 5.7 – State machine for the horizontal scan for beacons (Kouretes2007).

The state machine works through the end of each scan line until all pixels are processed. Once the scan line has finished the scan of the whole region, large parts of pixels belonging to the same color class are defined with the use of a function that is called either when the machine leaves the *Beacon Color Found* state or when the scan line comes to its end and the machine' s state is still *Beacon Color Found*. In order to determine a large part, this function takes a range of beacon color pixels (that is, consecutive pixels of beacon color possibly interrupted by some non-beacon color pixels). We can see how the large parts of the beacon are forming during the horizontal scan in Snapshot 5.8.



Snapshot 5.8 – Second window: the large parts of the blue-yellow beacon formed by horizontal scan, shown in red color.

All large parts found along the horizontal scan lines are added to a data structure, the *horizontal histogram.* As we also saw during the goal recognition, this is a structure where all the large parts are stacked together on top of each other to expose their density along the horizontal axis. The histogram is based on a discretization of the horizontal axis into 270 intervals covering the entire viewing angle. Each interval' s value is incremented every time this particular interval belongs to a large part (it lies between the starting and the ending point). When all the points are inserted, the histogram can give us information about the part of the image where most of the large parts exist.

Regarding the horizontal scan, we are looking for the point of the image where the product of the values of both histograms (one for each beacon color) is the maximum. This point is considered to be the center of the *horizontal beacon indication*. From this point, two scans begin, in opposite directions in order to find the limits of the beacon indication, horizontally. The first points, in both sides of the beacon indication center, where the aforementioned product becomes zero, are the limits of the *horizontal beacon indication*.

The way in which the horizontal histogram is formed, is depicted in Figure 5.8. We can also see how the *horizontal beacon indication* extends between the two vertical red lines in Snapshot 5.9.

Another condition that should be met refers to the relationship between the two colors' histograms. The two histograms' value for the same point of the histogram should not abstain more than a defined threshold, because the colored areas on each beacon are of the same size. This helps us to assure that the two colors will be the one above the other, as it is for the beacons.

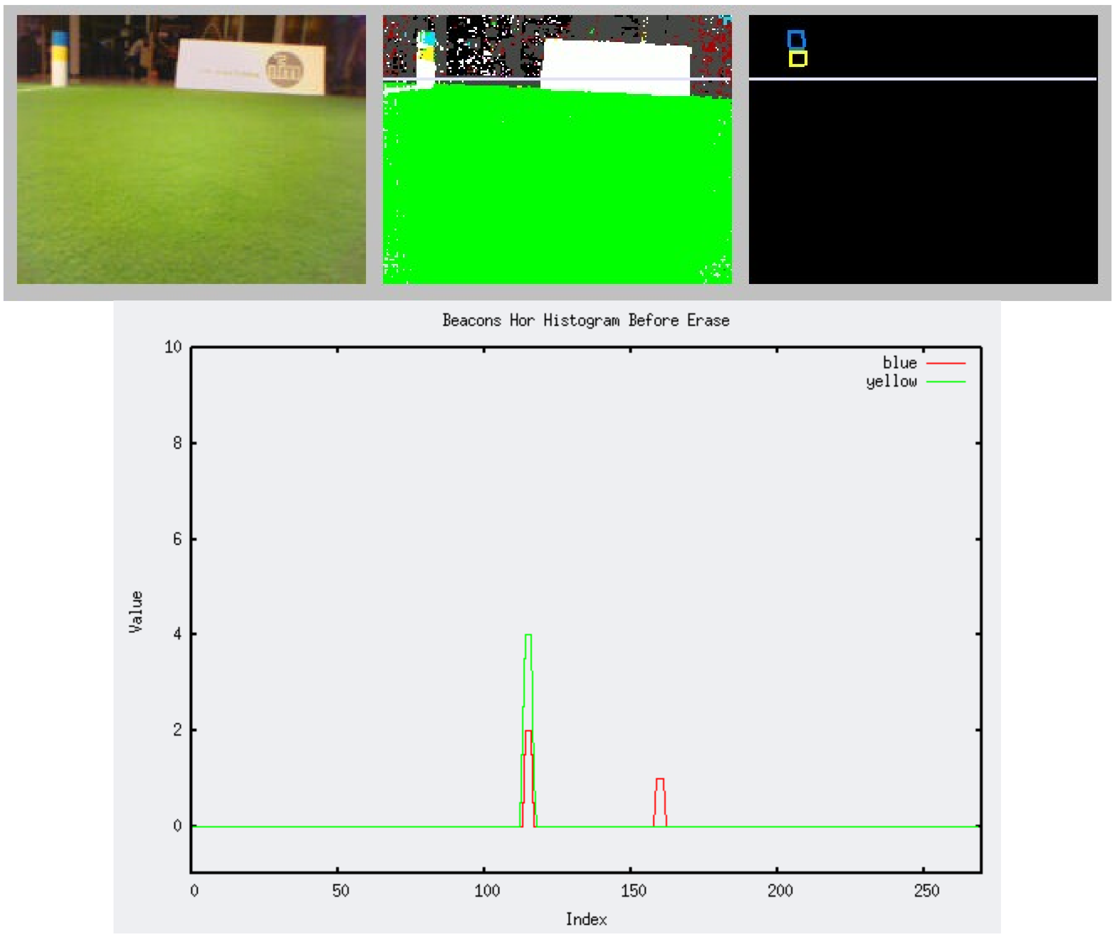
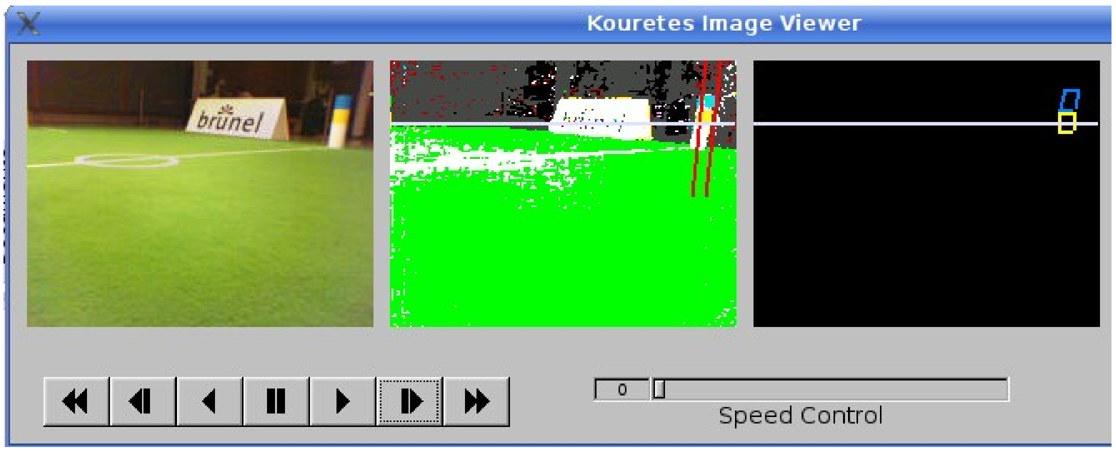


Figure 5.8 – Beacon horizontal histogram: the diagram shows how the histogram' s value converts with the large parts' edge points (index) for both beacon colors, during the scan for beacons of the color segmented image (second window of Kouretes image viewer, top image). Note the two peaks of the histogram which refer to the two beacon colors.



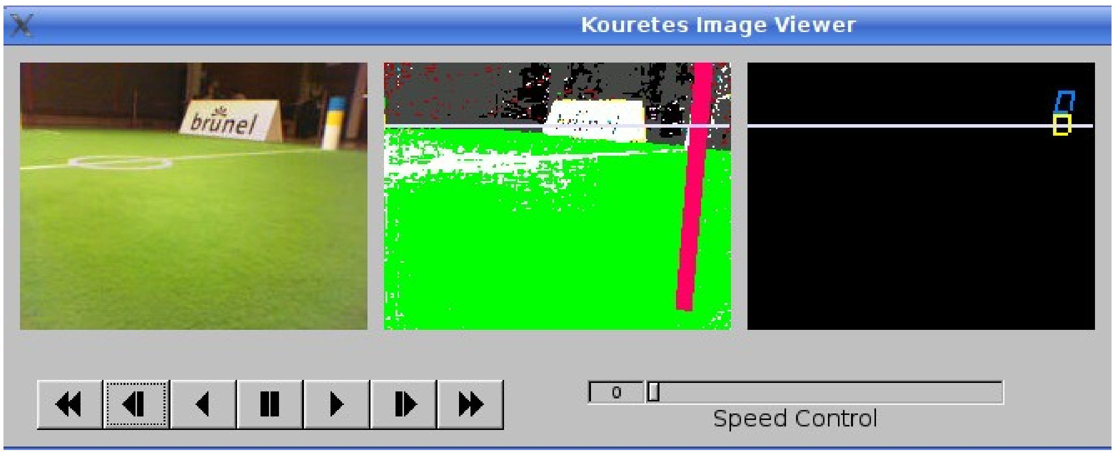
Snapshot 5.9 – Second window: the horizontal beacon indication of the blue-yellow beacon extends horizontally and in parallel to the horizon between the two vertical red lines.

All information acquired by the horizontal scan of the image is saved and will be further processed during the *beacon determination* phase of the process. We are now ready to describe the next step of the process, the *calculation of vertical scan lines.*

Calculation of vertical scan lines

The way that the vertical scan lines for the beacons are constructed differs from the one that is used for the goal recognition. The vertical scan takes place only within the horizontal beacon indication that has been previously defined. Only 7 vertical scan lines are constructed in order to scan through the indication.

The vertical scan lines are constructed so as to be perpendicular to the horizon. The distance between two different scan lines is constant and they cover the entire horizontal indication. The length of each scan line is fixed, above and below the *horizon line,* sufficient to include any beacon. We can see how the lines are formed in Snapshot 5.10. Since the beacon is far from the robot' s camera, the vertical scan lines seem are very close to each other.



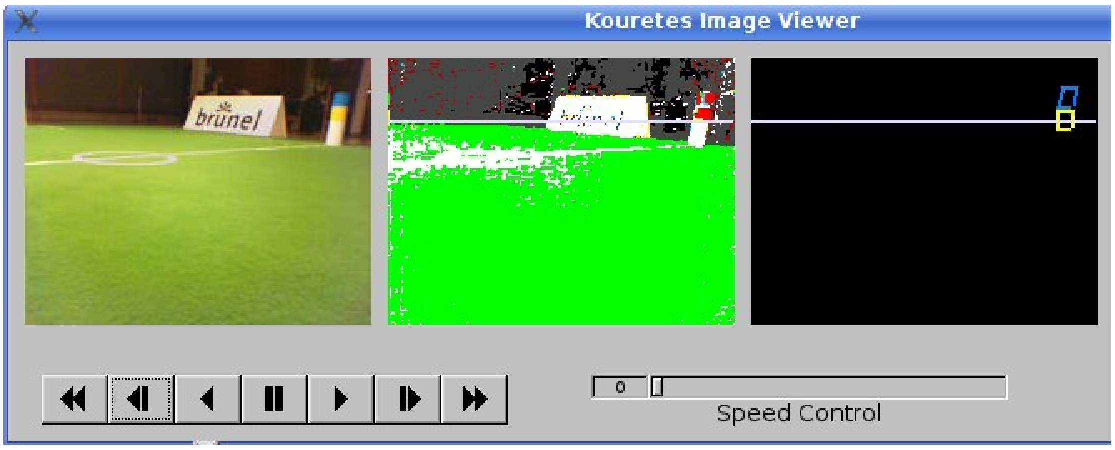
Snapshot 5.10 – In the second window we can see the vertical scan lines for the beacons which are perpendicular to the horizontal indication.

<u>Vertical scan for beacons</u>

The vertical scan for beacons uses exactly the same state machine as the horizontal one. The corresponding threshold (*vertical threshold*) are equal to the horizontal ones. The state machine for the vertical scan of the image is shown in Figure 5.7.

As the state machine runs for a particular scan line, the large parts are determined using the same function used during the horizontal scan. This function needs to have a range of beacon color pixels greater than the *vertical large-parts threshold* in order to determine a large part. The vertical large-parts threshold is equal to the horizontal one. We can see how the large parts of the beacon are forming during the vertical scan in Snapshot 5.11.

The procedure continues similarly to the horizontal scan for beacons. All large parts found on a particular line are added to a data structure, the *vertical histogram. T*his is a structure where all the large parts are stacked together on top of each other to expose their density along the vertical axis. The histogram is based on a discretization of the vertical axis into 190 intervals covering the entire viewing angle. Each interval' s value is incremented every time this particular interval belongs to a large part (it lies between the starting and the ending point). When all the points are inserted,



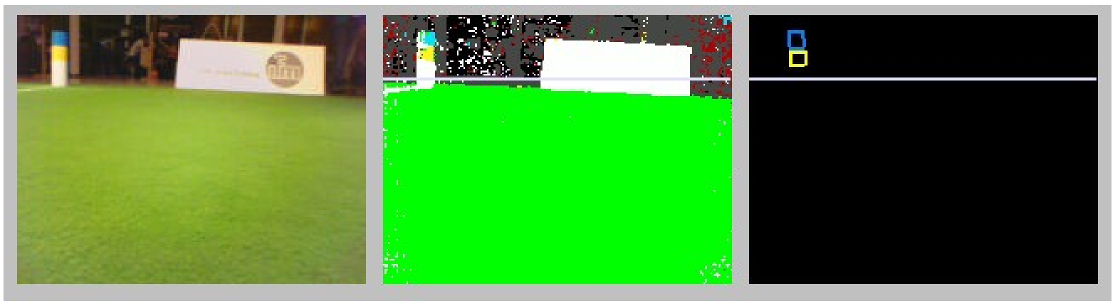
Snapshot 5.11 – Second window: the large parts of the blue-yellow beacon are forming after the vertical scan, shown in red color.

the histogram can give us information about the part of the image where most of the large parts exist.

Regarding the vertical scan, we are looking for the point of the image where the product of the values of both histograms (one for each beacon color) is the maximum. This point is considered to be the center of the *vertical beacon indication*. Then, we only check at which points of the histogram, on either side of the peak, the value first becomes zero, so we know that these particular histogram points can be considered as the limits of the *vertical beacon indication*.

The way in which the vertical histogram is formed, is depicted in Figure 5.9. The information acquired through the definition of both the *horizontal* and the *vertical beacon indication* is depicted in Snapshot 5.12.

From the vertical histogram we need to find out which of the two colors, blue or yellow, is above the other, so as to know the type of the beacon (blue-yellow or yellow-blue) that is currently perceived. Thus, the vertical histogram peak of the blue color is compared with that of the yellow color.



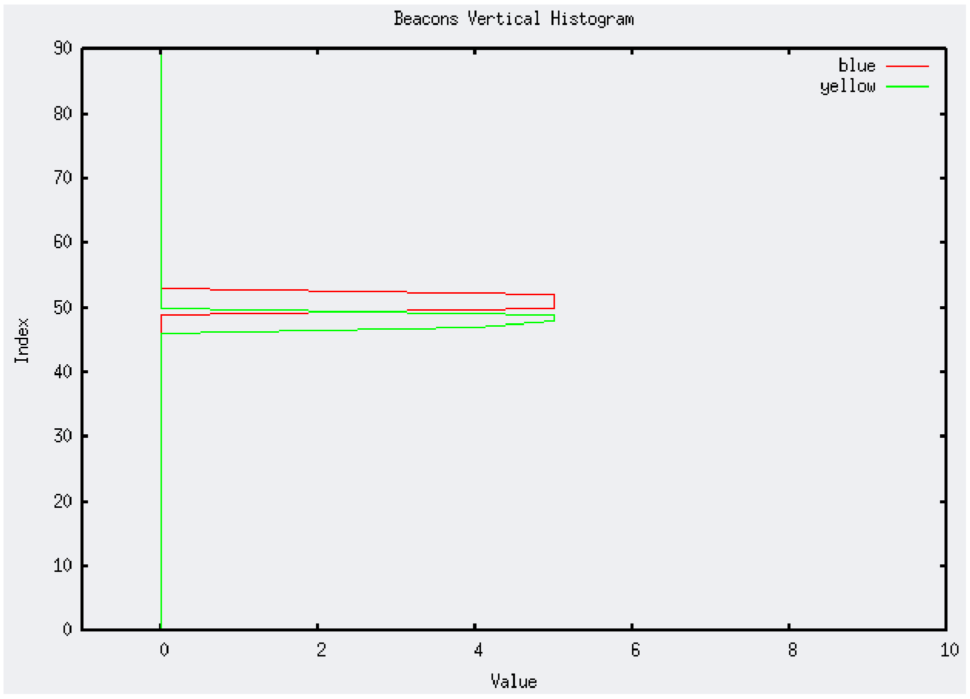
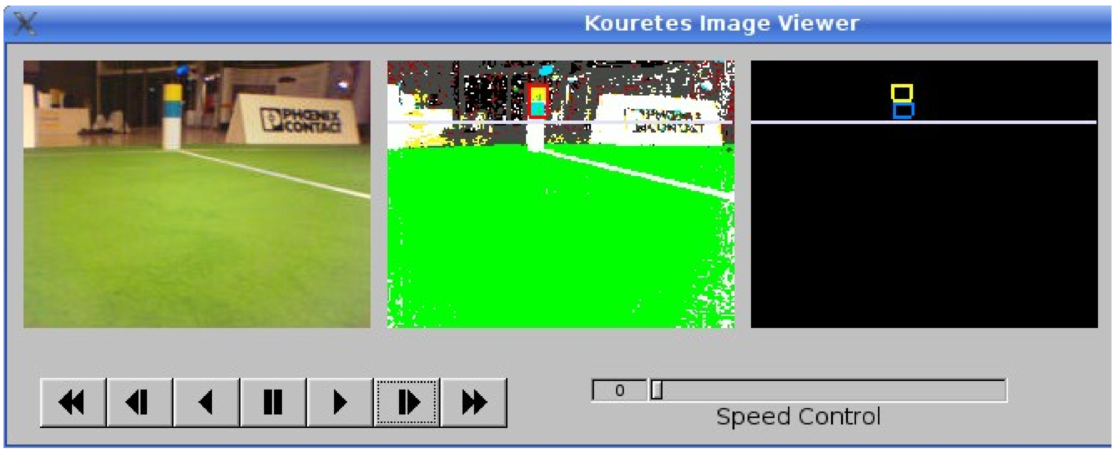


Figure 5.9 - Beacon vertical histogram: the diagram shows how the histogram' s value converts with the large parts' edge points (index) for the blue part of the beacon, during the scan for beacons of the color segmented image (second window of Kouretes image viewer, top image). Note the peak of the histogram where most large parts are assembled, which refers to the blue part of the beacon.



Snapshot 5.12 – Second window: the horizontal combined with the vertical beacon indication of the yellow-blue beacon.

All information acquired by the vertical scan of the image is saved and is further processed, combined with the corresponding information out of the horizontal scan, during the *beacon determination* phase of the process.

<u>Beacon Determination</u>

As we have already mentioned, this phase is reached only if both the *horizontal* and the *vertical beacon indications* have been defined according to the rules that have been set in previous phases. At this moment, we have information about two different indications for the existence of a beacon: one of them comes from the horizontal and the other one from the vertical scan. We are going to see how this information is used and how the final beacon is formed, as well as the different checks that the acquired information should successfully pass in order to refer to a recognized beacon.

First, in order to bound the colored region of the beacon, a *beacon boundary* is formed. This is a *Conditional Boundary* (see also Figures 4.9 and 4.10) whose function has already been described for the SPQRLegged2006. In our case, the horizontal beacon indication's edge points are added (see section 4.4) to form the horizontal dimension of the boundary, whereas the vertical indication's ones are added to form the vertical dimension of the boundary.

After the boundary has been formed and we know how the colored part of the beacon is extended within the image, we also want to pass this information through several filters in order to reject the formed landmark if it doesn't fit some essential specifications.

First, we verify whether the beacon boundary' s dimensions result in reasonable perceived distance. This is achieved by taking into account the distance in which the robot sees the perceived beacon boundary. This distance can be calculated through some geometrical transformations that use both the perceived and the actual dimensions of the beacon' s colored region. Thus, if this distance exceeds some specific threshold (negative or larger than the field size), we cannot use this beacon boundary as information to form the beacon, so we have to reject it. In that case no beacon is perceived.

Another check that is performed, refers to the ratio between the beacon dimensions which should be stable. Since we know the actual dimensions of the beacon' s colored region, we can calculate the ratio between them and compare this ratio to the beacon boundary' s dimension ratio. If the latest abstains a lot more than a specific tolerance from the actual beacon dimensions, then the found beacon boundary is rejected.

If the found beacon boundary has successfully passed all the aforementioned filters, that means that we can consider it as a perceived beacon. So, information about the beacon type (blue-yellow or yellow-blue), its center and its angle with respect to the robot can be easily calculated and sent to the next step of the vision, *landmarks percept* [16].

5.3 Ball Recognition

In this section we are focusing on how the robot recognizes the ball correctly and send the corresponding information to the next level of the robot' s cognition. The ball of the game is orange, according to the game' s rules.

In order to achieve that, we decided to scan the whole image frame, both horizontally and vertically with respect to the horizon, so as to get more accurate information about the color and the position of pixels of interest within the image. Then this information is further processed in order to yield the final result.

First , the *horizon line* and the *vertical line* are constructed in the same way as it has been described for the German Team. These lines are the base for constructing the horizontal and the vertical scan lines.

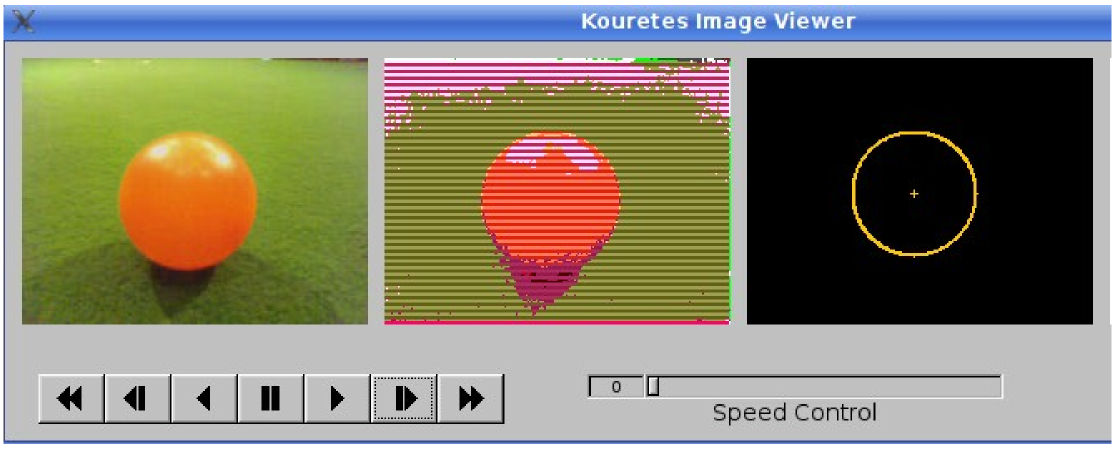
The whole process of the ball recognition consists of five different phases: *calculation of horizontal scan lines, horizontal scan for ball, calculation of vertical scan lines, vertical scan for ball* and *ball determination.*

Calculation of horizontal scan lines

During this phase of the process, the horizontal scan lines are constructed. They are parallel to the horizon and cover the whole image frame. We use 40 horizontal scan lines which are constructed to be equally distanced from each other.

Apart from the scanning area and the number of the scan lines, the construction process is the same as the corresponding one of the SPQRLegged team [17] . Two lines are constructed, *scanline* and *endline* which are parallel to the horizon line and perpendicular to the *vertical line.* Those lines' precise position depends on the sign of the angle between the horizon line and the horizontal side of the image frame. In Figures 4.2 and 4.3 we can see the construction of these lines keeping in mind that in our case the rectangular area of the Figures is the whole image frame instead of the SPQRLegged candidate region.

As before, the rest of the horizontal scan lines are going to be constructed between the *scanline* and the *endLine* and will be parallel to them. In Snapshot 5.13 we can see how the horizontal scan lines are constructed on the image frame, when the angle between the horizontal side of the frame and the horizon line is positive or negative. When the aforementioned angle is equal to zero, the horizontal scan lines are also parallel to the horizontal side of the image.



Snapshot 5.13 – Kouretes Image Viewer: in the second window we can see the horizontal scan lines for the ball which are parallel to the horizon. The angle between the horizon line and the horizontal side of the image frame is, here, zero.

Horizontal scan for ball

During this phase, the horizontal scan lines can scan the image pixel by pixel and check whether one pixel belongs to the orange color class or not. The process that follows is similar to the one used for the goals and the beacons recognition.

The state machine that we use consists of two states, *No Ball Color Found* (NB) and *Ball Color Found* (B), and aims to find the parts of consecutive pixels of the same color. This machine begins from the state *No Ball Color Found,* where the scan line either has not met orange pixels yet or the ones found are less than the *horizontal threshold* in sequence. Once the scan line finds as many consecutive orange pixels as the threshold, the state changes to *Ball Color Found,* where the machine remains as long as the scan continues to meet pixels of the same color or if less than the threshold consecutive pixels of non-ball color. If the scan hits the threshold in consecutive pixels of a different color, the state comes back to *No Ball Color Found* state. We can better see the structure of this state machine in Figure 5.10.

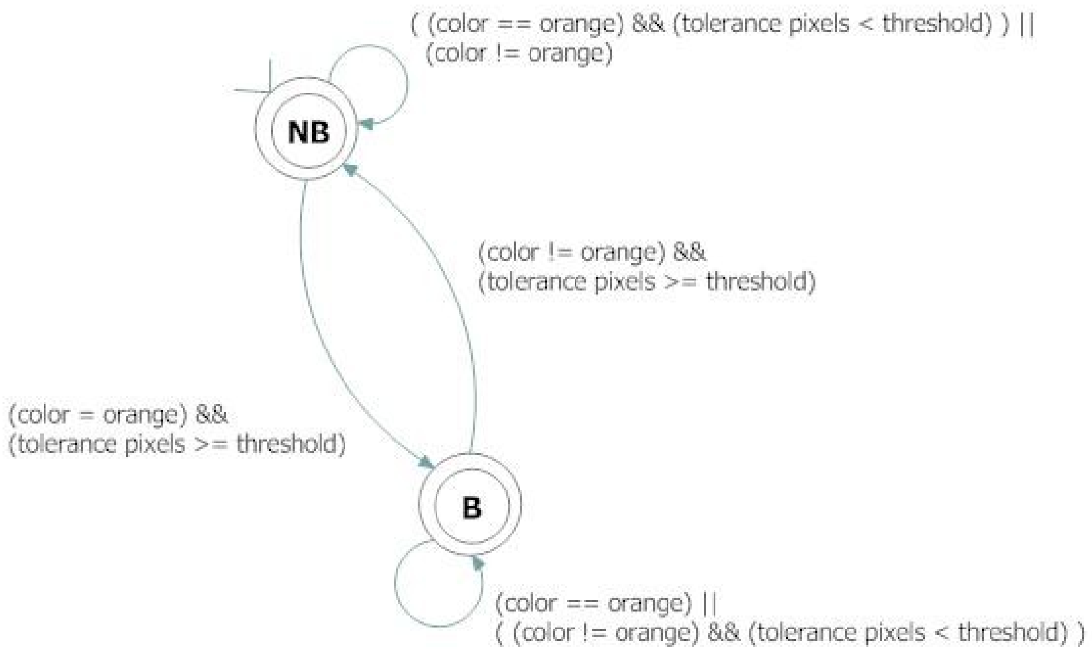
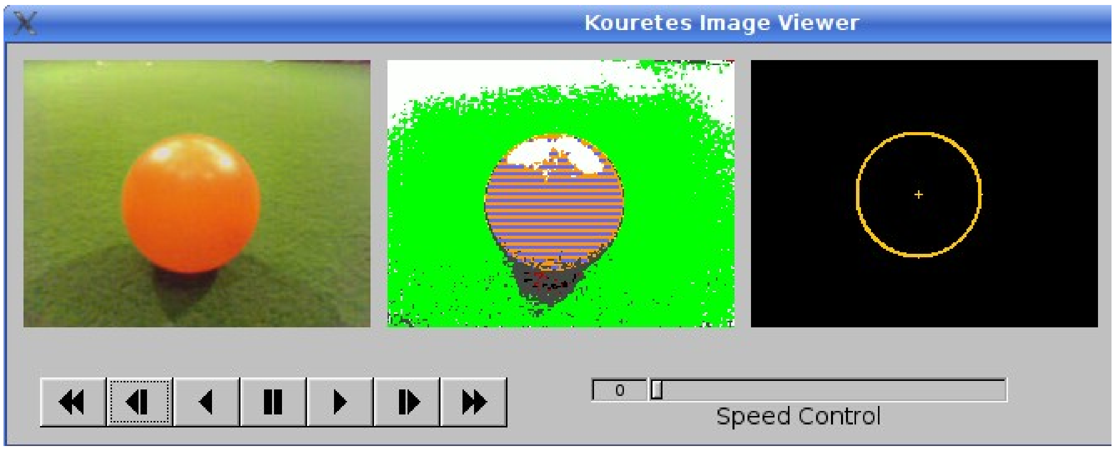

((color == orange) && (tolerance pixels < threshold)) ||
(color != orange)
NB
(color != orange) &&
(tolerance pixels >= threshold)
(color = orange) &&
(tolerance pixels >= threshold)
B
(color == orange) ||
((color != orange) && (tolerance pixels < threshold))

Figure 5.10 – State machine for the horizontal scan for the ball (Kouretes2007).

The state machine works through the end of each scan line until all pixels are processed. Once the scan line has finished the scan of the whole region, large parts of orange pixels are defined with the use of a function that is called either when the machine leaves the *Ball Color Found* state or when the scan line comes to its end and the machine' s state is still *Ball Color Found*. In order to determine a large part, this function takes a range of orange pixels (that is, consecutive orange pixels possibly interrupted by some non-orange color pixels) marking the entrance and the exit to the state *Ball Color Found*. We can see how the large parts of the ball are forming during the horizontal scan in Snapshot 5.14.



Snapshot 5.14 – Second window: the large parts of the ball formed by horizontal scan, shown in blue color.

All large parts found along the horizontal scan lines are added to a data structure that determines clusters. The clustering of large parts occurs in the same way as described for the SPQRLegged team (see Figure 4.5).

Having formed the ball clusters from the large parts, we now create the *horizontal ball indication across,* which shows the extent of the ball across the horizontal scan lines. The *horizontal ball indication across* begins and ends at the most extreme scan lines containing large parts. For example, it could extend from the second till the tenth horizontal scan line.

The large parts, are also added to another data structure, the *horizontal histogram.* This is a structure where all the large parts are stacked together on top of each other to expose their density along the horizontal axis. The histogram is based on a discretization of the horizontal axis into 270 intervals covering the entire viewing angle. Each interval' s value is incremented every time this particular interval belongs to a large part (it lies between the starting and the ending point). When all the points are inserted, the histogram can give us information about the part of the image where most of the large parts exist.

Regarding the horizontal scan, we are looking for the point of the image where the value of the histograms is the maximum. This point is considered to be the center of the *ball horizontal indication along.* From this point, two scans begin, in opposite directions in order to find the limits of the ball indication, horizontally. The first points, in both sides of the ball indication center, where the aforementioned value becomes zero, are the limits of the *horizontal ball indication along*.

The way in which the horizontal histogram is formed, is depicted in Figure 5.11. Both the *horizontal ball indication along* and the *horizontal ball indication across* the horizontal scan lines are depicted in Snapshot 5.15.

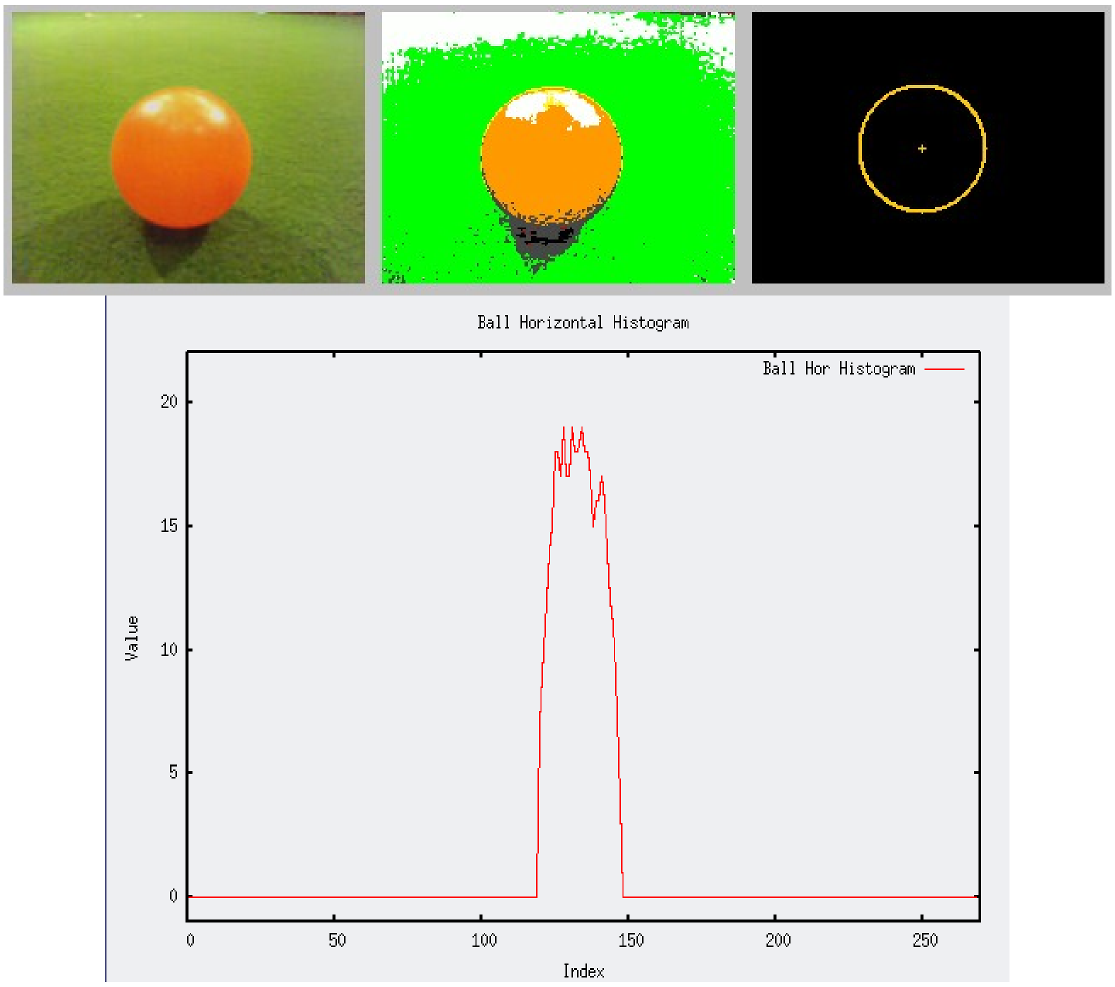
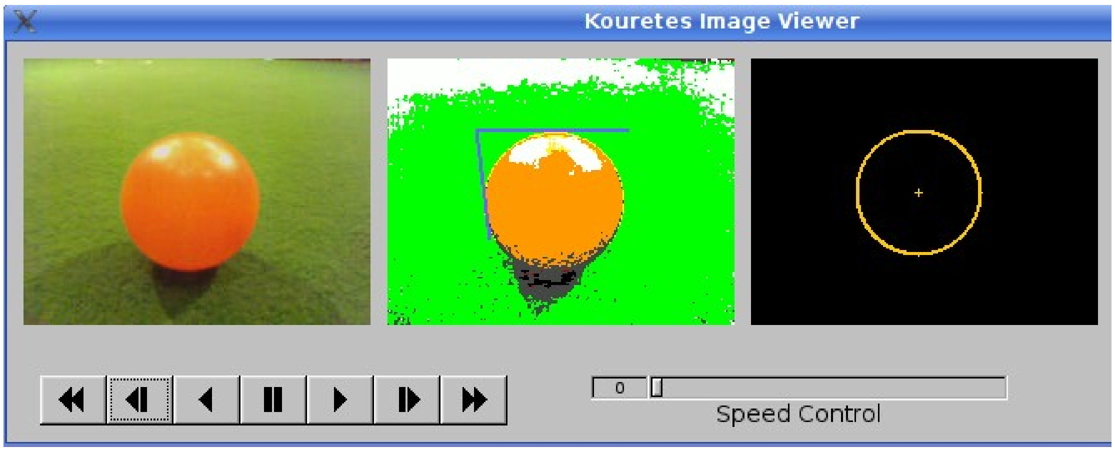


Figure 5.11 – Ball horizontal histogram: the diagram shows how the histogram' s value converts with the large parts' edge points (index) for the orange color, during the scan for ball of the color segmented image (second window of Kouretes image viewer, top image). Note the peak of the histogram which refer to the center of the ball horizontal indication.



Snapshot 5.15 – Second window: the horizontal ball indications (along and across) formed by horizontal scan, shown in blue color.

All information acquired by the horizontal scan of the image is saved and will be further processed during the *ball determination* phase of the process. We are now ready to describe the next step of the process, the *calculation of vertical scan lines.*

Calculation of vertical scan lines

This phase is exactly the same as the corresponding one for the goals. The vertical scan lines are constructed so as to be perpendicular to the horizon and scan the whole image frame. We use 50 vertical scan lines which are constructed to be equally distanced from each other.

The construction procedure begins. Two lines are constructed, *scanline* and *endline* which are parallel to the *vertical line* and perpendicular to the horizon line. Those lines' precise position depends on the sign of the angle between the horizon line and the horizontal side of the image frame. In Figures 5.12 and 5.13 we can see the construction of these lines onto the whole image frame.

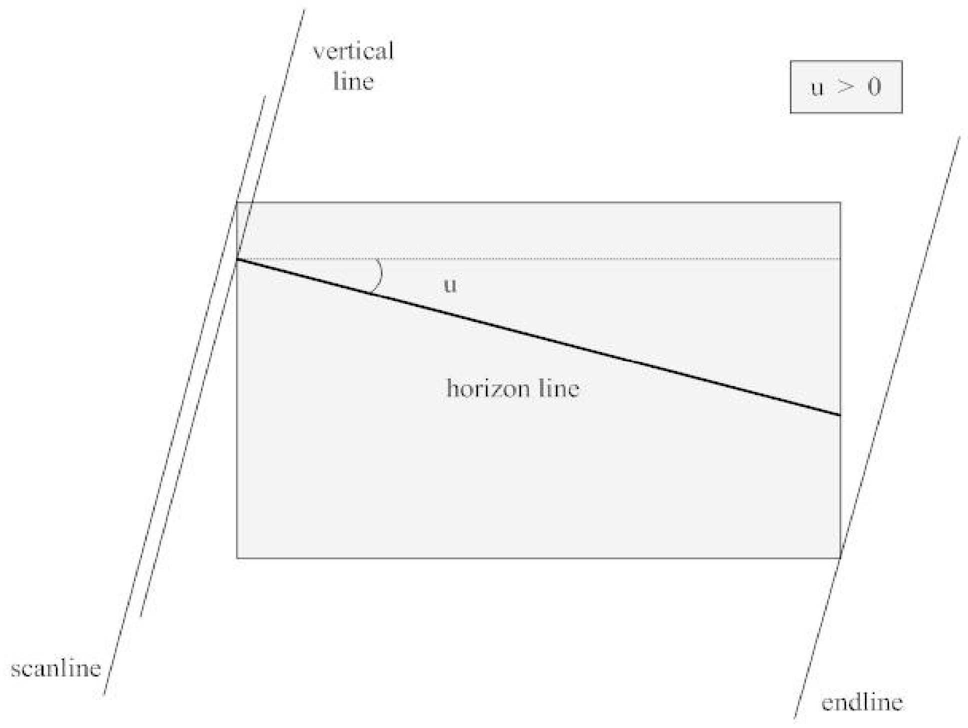


Figure 5.12 – Vertical ball scan lines: construction of vertical line, scanline and endline when the angle between the horizon line and the horizontal side of the image frame is positive.

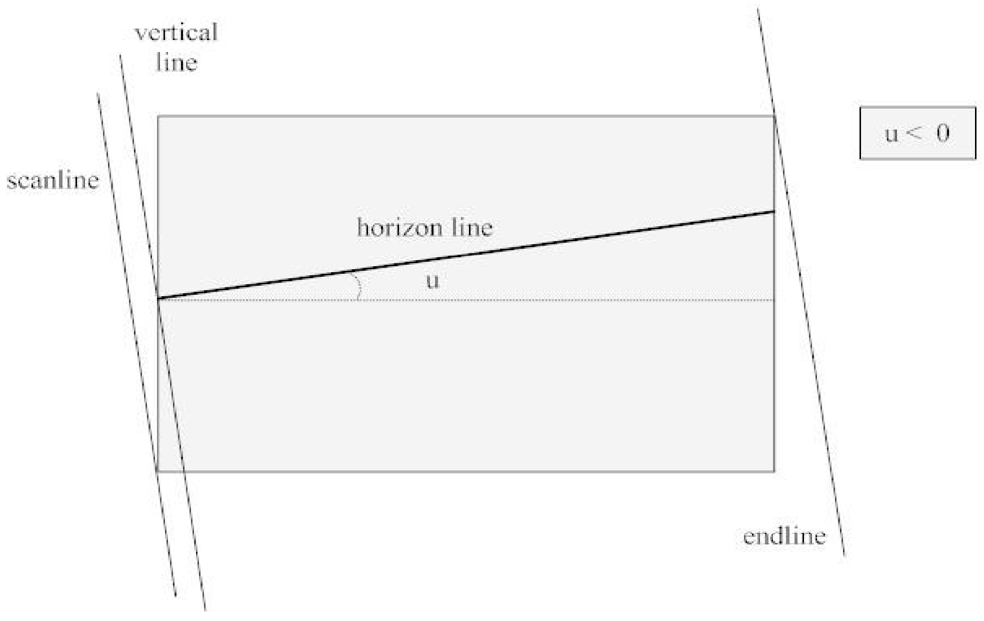
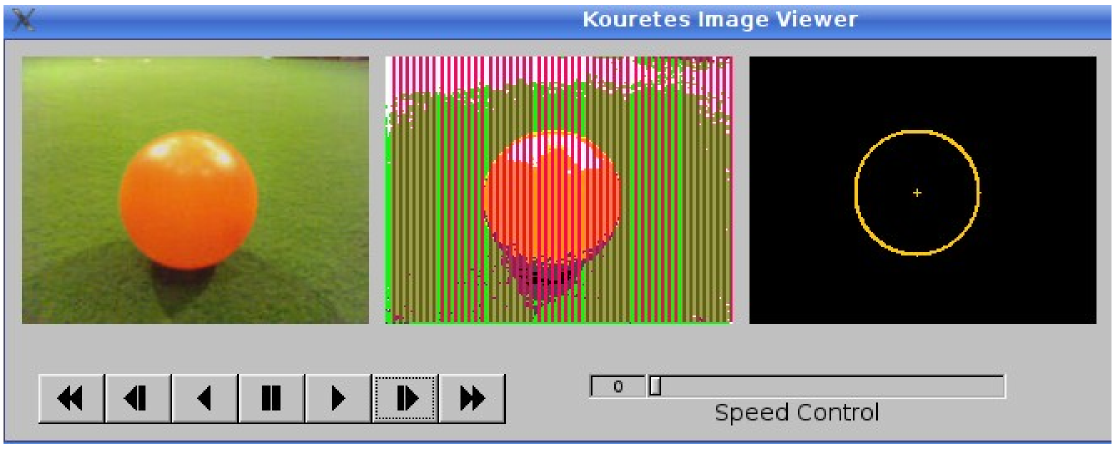


Figure 5.13 – Vertical ball scan lines: construction of vertical line, scanline and endline when the angle between the horizon line and the horizontal side of the image frame is negative.

The rest of the vertical scan lines are going to be constructed between the scanline and the endLine and will be parallel to them. In Snapshots 5.16 and 5.20 we can see how the vertical scan lines are constructed on the image frame, when the angle between the horizontal side of the frame and the horizon line is positive or negative. When the aforementioned angle is equal to zero, the vertical scan lines are also parallel to the vertical side of the image frame.

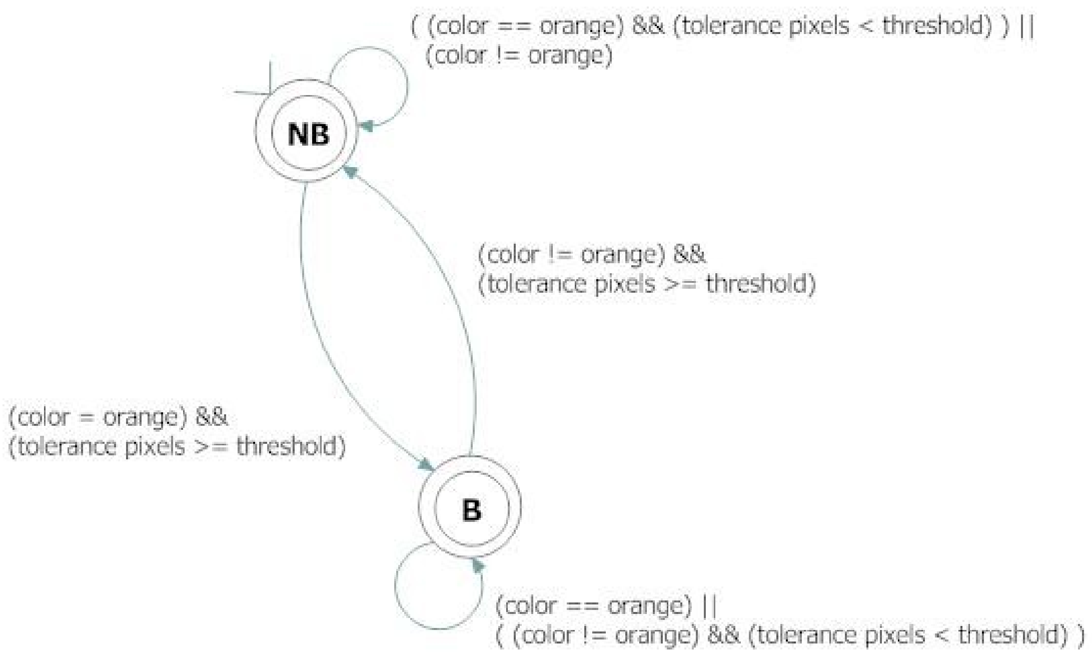


Snapshot 5.16 – In the second window we can see the vertical scan lines for the ball which are perpendicular to the horizon. The angle between the horizon line and the horizontal side of the image frame is equal to zero.

The vertical scan lines are going to scan the whole image according to what is described in the next phase.

<u>Vertical scan for ball</u>

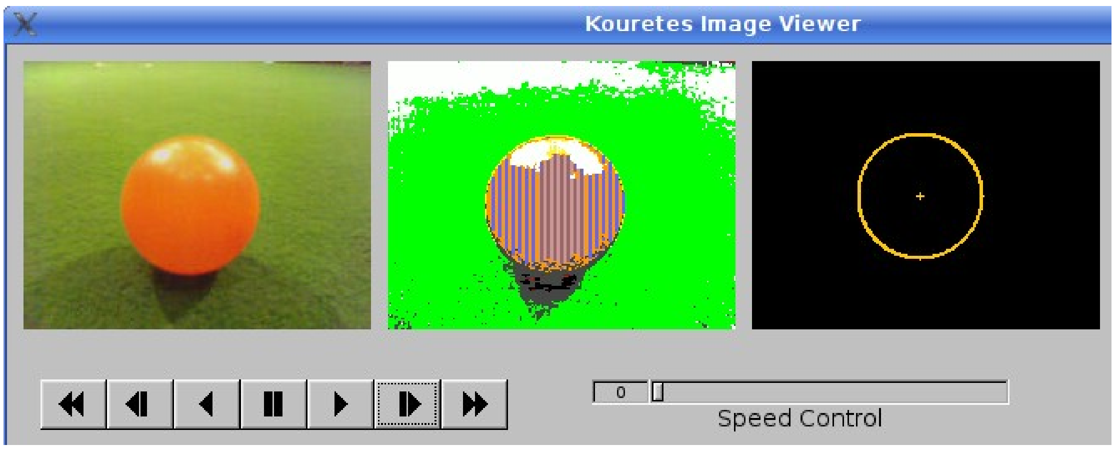
The vertical scan for the ball uses the same state machine as the horizontal one. As the ball can be really small within the image, we keep a low tolerance in orange or non-orange pixels in order to deal with such cases. The state machine for the vertical scan of the image is shown in Figure 5.14.



((color == orange) && (tolerance pixels < threshold)) ||
(color != orange)
NB
(color != orange) &&
(tolerance pixels >= threshold)
(color = orange) &&
(tolerance pixels >= threshold)
B
(color == orange) ||
((color != orange) && (tolerance pixels < threshold))

Figure 5.14 – State machine for the vertical scan for the ball (Kouretes2007).

As the state machine runs for a particular scan line, the large parts are determined using the same function used during the horizontal scan. This function needs to have a range of orange pixels greater than the *vertical large-part threshold* in order to determine a large part. We can see how the large parts of the ball are forming during the vertical scan in Snapshot 5.17.



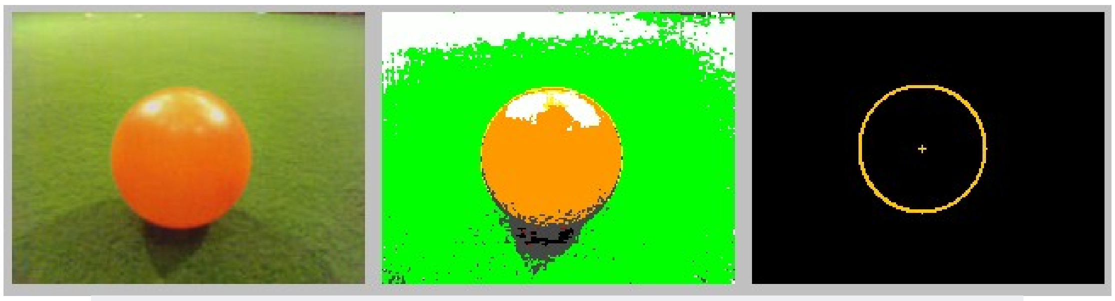
Snapshot 5.17 – Second window: the large parts of the ball are forming after the vertical scan, shown in blue color.

The procedure continues similarly to the horizontal scan. The large parts found along a particular line are added to the structure that determines clusters and the *vertical ball indication across* is created to show how the ball extends within the image across the vertical scan lines.

The large parts, are also added to another data structure, the *horizontal histogram.* This is a structure where all the large parts are stacked together on top of each other to expose their density along the vertical axis. The histogram is based on a discretization of the vertical axis into 90 intervals covering the entire viewing angle. Each interval' s value is incremented every time this particular interval belongs to a large part (it lies between the starting and the ending point). When all the points are inserted, the histogram can give us information about the part of the image where most of the large parts exist.

Regarding the vertical scan, we are looking for the point of the image where the value of the histograms is the maximum. This point is considered to be the center of the *ball vertical indication along*. From this point, two scans begin, in opposite directions in order to find the limits of the ball indication, vertically. The first points, in both sides of the ball indication center, where the aforementioned value becomes zero, are the limits of the *vertical ball indication along*.

The way in which the vertical histogram is formed, is depicted in Figure 5.15. Both the *vertical ball indication along* and the *vertical ball indication across* the vertical scan lines are depicted in Snapshot 5.18.



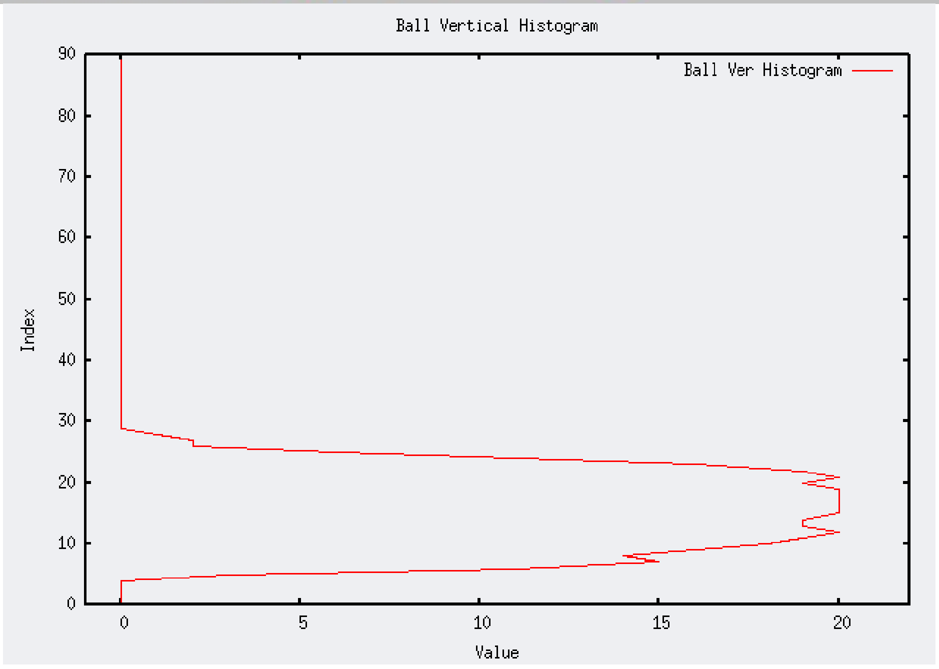
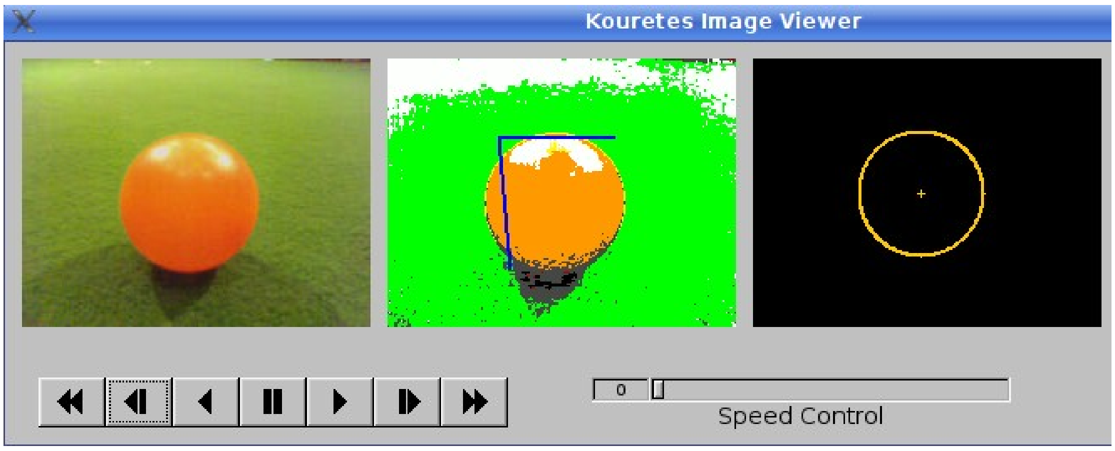


Figure 5.15 – Ball vertical histogram: the diagram shows how the histogram' s value converts with the large parts' edge points (index) for the orange color, during the scan for ball of the color segmented image (second window of Kouretes image viewer, top image). Note the peak of the histogram which refer to the center of the ball vertical indication.



Snapshot 5.18 – Second window: the vertical ball indications (along and across) formed by vertical scan, shown in blue color.

All information acquired by the vertical scan of the image is saved and will be further processed, combined with the corresponding information out of the horizontal scan, during the *ball determination* phase of the process, which we are about to analyze.

Ball determination

At this moment, we have information about four different indications for the existence of the ball: two of them come from the horizontal scan and the other two from the vertical scan. We are going to see how we decide which information is preferable to use and how the final ball is formed by calculating its right center and radius.

There are some cases where a ball indication' s maximum and minimum points, acquired by a specific scan, are identical. Then, both points are replaced by the corresponding ones acquired by the other scan. For example, if the horizontal scan has resulted into a horizontal goal indication along whose minimum and maximum points are identical, but their corresponding (vertical goal indication across) minimum and maximum points, acquired by the vertical scan, are not identical, then both horizontal goal indication along minimum and maximum points are replaced by the corresponding ones from the vertical scan, so they are not identical any more.

With the end of this replacement, we can now use the information that the ball indications give us. In Figures 5.16 and 5.17 we visualize this information for the horizontal and the vertical scan, respectively. We can see that if we combine the starting and ending points of both the horizontal indications (along and across) for the ball, we end up with a square that surrounds the ball. Thus, we can gain information about the center and the radius of the circle, that represents the ball within the image frame, with the use of simple geometrical calculations.

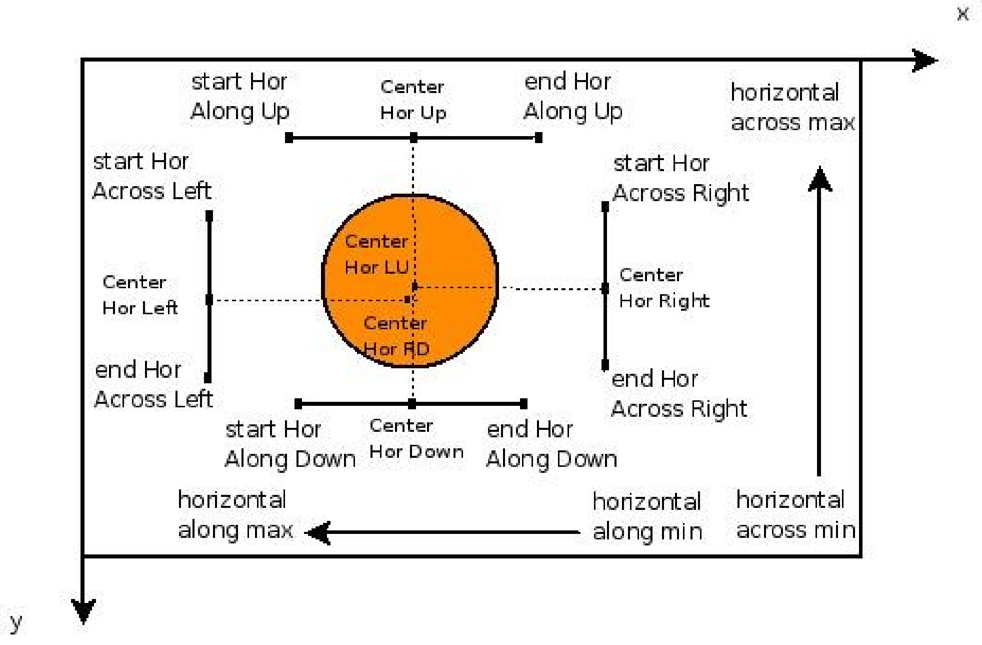


Figure 5.16 – The information acquired out of the horizontal scan for the ball within the rectangular image frame. Estimations about the center.

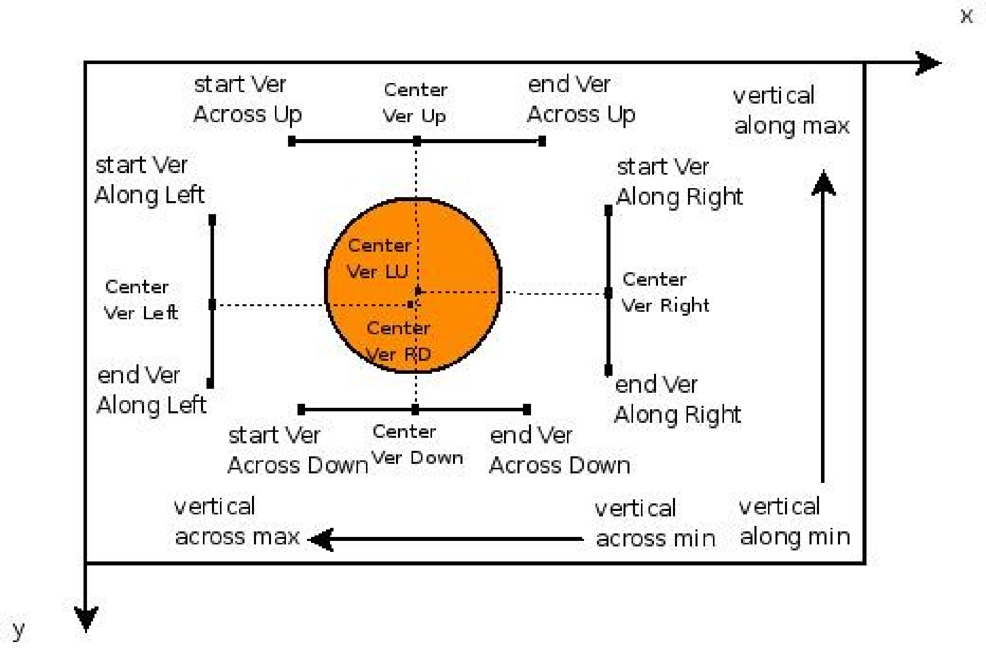


Figure 5.17 – The information acquired out of the vertical scan for the ball within the rectangular image frame. Estimations about the center.

However, one can easily observe that the ball cannot always appear within the image in its complete circular shape; it can also appear as a semi-circle, a quarter-circle or even a smaller part. So, we distinguish between two cases: (i) the ball is completely seen and (ii) a part of the ball is hidden by another object of the field or by the edge of the image frame (see also Figure 5.18).

The way we distinguish the two cases, one from the other, is related to the fact that when the ball is hidden one of the following occurs: the horizontal or the vertical scan *start* with orange pixels or the horizontal or the vertical scan *end* with orange pixels. For example, if we take a look at the bottom picture of Figure 5.18 we can realize that, given that the horizontal scan lines are parallel to the horizontal side of the image frame, the horizontal scan ends with orange pixels' perception. As we can see, in that case the ball is actually hidden from the right.

Another criterion of distinguishing the ball appearance cases is the ratio between the horizontal and the vertical indication of the ball which should not exceed a predefined threshold. If this ratio is greater than the threshold, this may refer to a hidden ball.

The procedure that leads to the ball recognition differs in each of the two aforementioned cases.

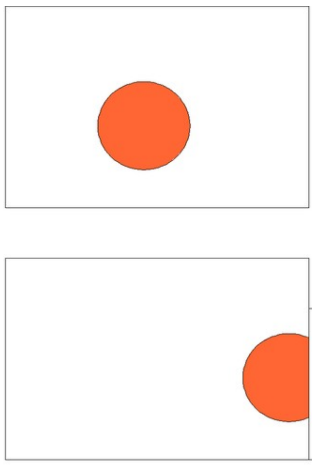


Figure 5.18 – The upper image shows a case where the ball can be seen completely, whereas the bottom one shows a case where the ball is hidden by the right vertical side of the image frame.

(i) Ball Not Hidden: When the ball is completely seen within the image frame, it appears as a circle. With the use of the information depicted in Figures 5.16 and 5.17, we can estimate both the radius and the center.

Each side of the box that surrounds the ball can give us a different estimation for the radius length, by taking the half of its length. So, we have four estimations for the radius out of the horizontal scan and, if we calculate their mean, we can have a total estimation deriving from the horizontal scan for the ball. We follow the same procedure for the vertical indications, as well, resulting into the corresponding total estimation for the ball, coming out of the vertical scan. Then, we take the mean of those two total estimations and we have the final radius of the circle that represents the ball.

In order to find the center of this circle, we follow a similar procedure. The medium point of each side of the box that surrounds the ball (see Figures 5.16 and 5.17) is calculated. So there are four new points that give us two different values for each center coordinate (two values for the x- and two for the y-coordinate). Then, the mean of these values is calculated for each coordinate and gives a total value for the x- and a total value for the y-coordinate. The aforementioned procedure is followed for both the horizontal and the vertical scan information, so the final result for the center point of the circle comes from the mean of the corresponding total coordinates from each scan.

(ii) Ball Hidden: In the case that the ball is not seen completely within the image frame, we use a specific function in order to find the center of the circle whose part is the visible region of the ball within the image frame. This function needs three points of the circle perimeter in order to calculate the center position as the intersection of the middle perpendiculars of the triangle that the three points define. The selection of the points depends on the side where the ball is hidden from. The points coordinates can be easily calculated, since we already have the information for the box that surrounds the visible region of the ball.

The corresponding radius to the calculated center can be found as the distance between the center and another known point of the ball indication, depending on which side of the ball is hidden each time. In Figures 5.19 and 5.20 we can see an example of a hidden ball perception. The three points used to calculate the center are *start Hor Across Right, end Hor Across Right* and *center Hor Left* (we have already seen these points in Figures 5.16 and 5.17) for the horizontal ball indication. We pick the corresponding points from the vertical indication and calculate the corresponding center. The final ball center' s coordinates result from the mean of the two aforementioned centers' coordinates.

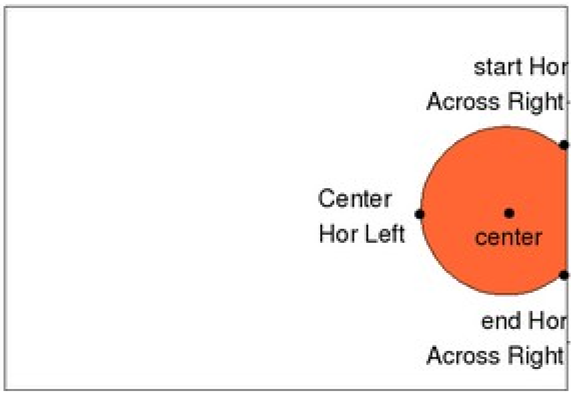


Figure 5.19 – Right hidden ball in the image frame. The center of the ball, regarding the vertical ball indication, is calculated by three points on the circle.

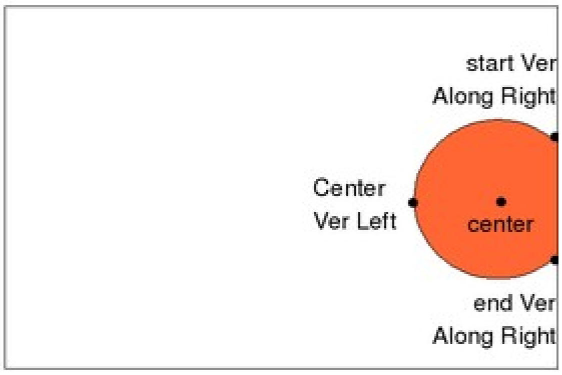


Figure 5.20 – Right hidden ball in the image frame. The center of the ball, regarding the vertical ball indication, is calculated by three points on the circle.

On the other hand, the radius comes from the mean of the two distances: the distance between the horizontal center and the *center Hor Left* and the one between the vertical center and the *center Ver Left.*

As soon as the center and the radius of the ball have been calculated, they are sent to the next level of the cognition, ballPercept [16].

5.4 Implementation

As it has been already mentioned, our approach to the problem has been implemented within the framework of the modular SPQRLegged code. That is why some technical aspects of the implementation were already predetermined.

More specifically, the code for object recognition is object-oriented and was written in the C++ programming language. The code was developed, compiled, and tested under a Linux environment using the g++ 3.3compiler for desktop execution and debugging and the mipsel-g++ cross-compiler for autonomous execution on the robot.

6 Evaluation and results

The last chapter of the present thesis deals with the evaluation of the proposed approach. First, the statistics of the successful object recognition are calculated and compared with the corresponding ones of two other teams: the German Team and the SPQRLegged. Then, the success of the current approach is measured in the context of localization of the perceived objects, using the vision module that is hereby introduced.

6.1 Kouretes 2007 Image Processor statistics

The video stream which derives from the robot camera is processed frame by frame. In order to evaluate the Kouretes 2007 Image Processor there are four categories of results within an image frame that are counted: *true positives, true negatives, false positives* and *false negatives.*

True positives refer to the cases in which an object of interest, for example the ball, is present in the raw image and is also perceived by our image processor. False positives are called the cases in which an object of interest, is not present in the raw image, but is indeed perceived by our image processor. True negatives are the cases in which an object of interest, is absent in the raw image and so it is not perceived by our image processor. Finally, false negatives refer to the cases in which an object of interest is present in the raw image, but it is not perceived by our image processor. After the observation, which has been manually performed, of three log files taken by the robot camera from the four-legged league field during the 2007 RoboCup German Open competition, the cumulative results are presented in Table 6.1. There can be found the statistics of the true positives versus false negatives, as well as the true negatives versus the false positives.

	True Positives	False Negatives	True Negatives	False Positives
Goals	61%	39%	93%	7%
Beacons	34%	66%	99%	1%
Ball	79%	21%	100%	0%

Table 6.1 – The results of the Kouretes 2007 Image Processor in cases where each object of interest (goals, beacons, ball) is either present or absent within the image frame and is either perceived by the vision module or not.

False negatives for all objects can derive from the rapid movement of the robot' s head, which can degrade the objects' original appearance and shape within the image frame. In addition, bad color segmentation or shadows and reflections within the image frame can lead to misunderstanding of the object shape, which can eventually can cause a false negative.

Regarding the relatively low percentage of true positives that are reached for the beacons, it is usually due to the small colored area on which the vision module is based in order to recognize the beacons. On the other hand, the ball perception reaches the highest percentage of all objects and zero percentage of false positives. The latest is highly related with the fact that the only object that is orange within the field is the ball, in contrast with the blue or yellow color which have to do with both the goals and the beacons.

The comparison of these results with the corresponding ones of other teams is the subject of the section that follows.

6.2 Comparison with other Image Processors

Once the results of the present work have been acquired, it is useful to compare them with other works that have been dealing with the specific problem and see if the existing results can be further improved.

It is not possible to compare the goal recognition of the present work with any other available work, since the goal appearance has changed during the last two RoboCup competitions. However, it is possible to compare the ball and the beacon recognition of the Kouretes 2007 Image Processor with the corresponding ones of two other teams, whose source codes is available: the German Team 2004 (GT2004) and the SPQRLegged 2006 (SPQRL2006).

Three ways of comparison are presented: (i) comparing the *statistics* that each team' s image processor has reached, (ii) comparing the *image processing time,* that takes for each processor to perform, and (iii) comparing the way that an object is *perceived*, by means of size and distance from the robot, among the different image processors.

Statistics

The performance of manual observation on the same log files as in section 6.1, using the image processors of the two other teams (the German Team and the SPQRLegged Team) led to the results that are presented in Tables 6.2 and 6.3.

	True Positives	**False Negatives**	**True Negatives**	**False Positives**
Beacons	28%	72%	99%	1%
Ball	67%	33%	99%	1%

Table 6.2 – The results of the German Team 2004 Image Processor in cases where each object of interest (beacons, ball) is either present or absent within the image frame and is either perceived by the vision module or not.

	True Positives	**False Negatives**	**True Negatives**	**False Positives**
Beacons	31%	69%	99%	1%
Ball	51%	49%	100%	0%

Table 6.3 – The results of the SPQRLegged 2006 Image Processor in cases where each object of interest (beacons, ball) is either present or absent within the image frame and is either perceived by the vision module or not.

Note that the GT2004 Image Processor seems to recognize the ball better than the SPQRL2006 one, whereas the SPQRL2006 performs the beacon recognition better than the GT2004. However, the statistics show that the Kouretes 2007 Image Processor has improved both the beacon and the ball recognition, when compared with the GT2004 and the SPQRL2006, by increasing the number of true positives and, correspondingly, reducing the number of false negatives.

Image Processing Time

At this point, it is interesting to compare the time that is needed for the image processor of each team to function. One of the previously used log files (see section 6.1), consisting of 73 frames, is also used here in order to find out the average image processing time out of the image processing time of each frame. Thus, the specific log file has been processed by all three image processors and the results that have been reached are presented in Table 6.4.

	GT 2004	**SPQRL 2006**	**Kouretes 2007**
Average Image Processing Time	42ms	52ms	51ms
Variance	121	111	130

Table 6.4 – The average image processing time and the variance reached by the image processor of three different teams: the 2004 German Team, the 2006 SPQRLegged and the 2007 Kouretes team.

The results are the ones that have been expected. The GT2004 performs a relatively short image processing time, since the grids of scan lines that are used do not cover the whole image, but only the regions that are most likely to contain the objects of interest. Then, the SPQRL2006 needs more time to process the image because, even though the image is again partially processed, extra time is needed for the candidate regions to be determined. Last but not least, the Kouretes 2007 needs an average image processing time of 51ms. Note here that the cost of the increase of true positives in comparison with the GT2004, is the corresponding increase of 9ms of the image processing time. That is mostly due to the fact that the Kouretes 2007 uses a grid of scan lines that is extended within the whole image frame, which demands more image processing time.

Finally, it is also interesting to compare the variance of the 73 time values (one for each frame) between the three different image processors. Table 6.4 shows the corresponding results.

Perception window

In this part of the evaluation, the attention is driven towards the third window of the Kouretes Image Viewer, the *perception window.* The difference between the recognition that is performed by each image processor is obvious in Figures 6.1 and 6.2 for the beacons and the ball, respectively.

The aforementioned figures show examples in which the Kouretes 2007 Image Processor has improved the corresponding object recognition by turning a false negative into a true positive case for both the blue-yellow beacon and the ball.

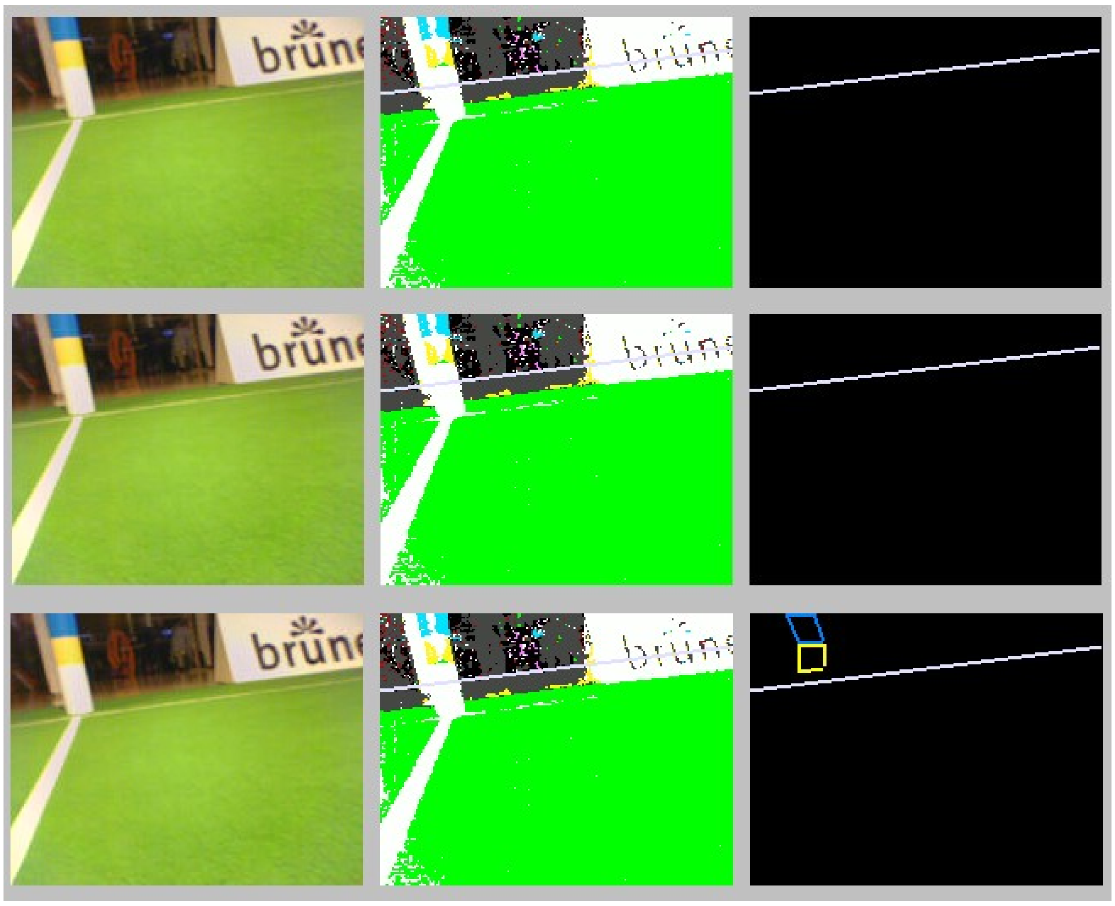


Figure 6.1 – The difference in beacon recognition among the three image processors: the top snapshot is taken from the GT2004, the middle one from the SPQRL2006 and the bottom snapshot is taken from the Kouretes 2007 image processor.

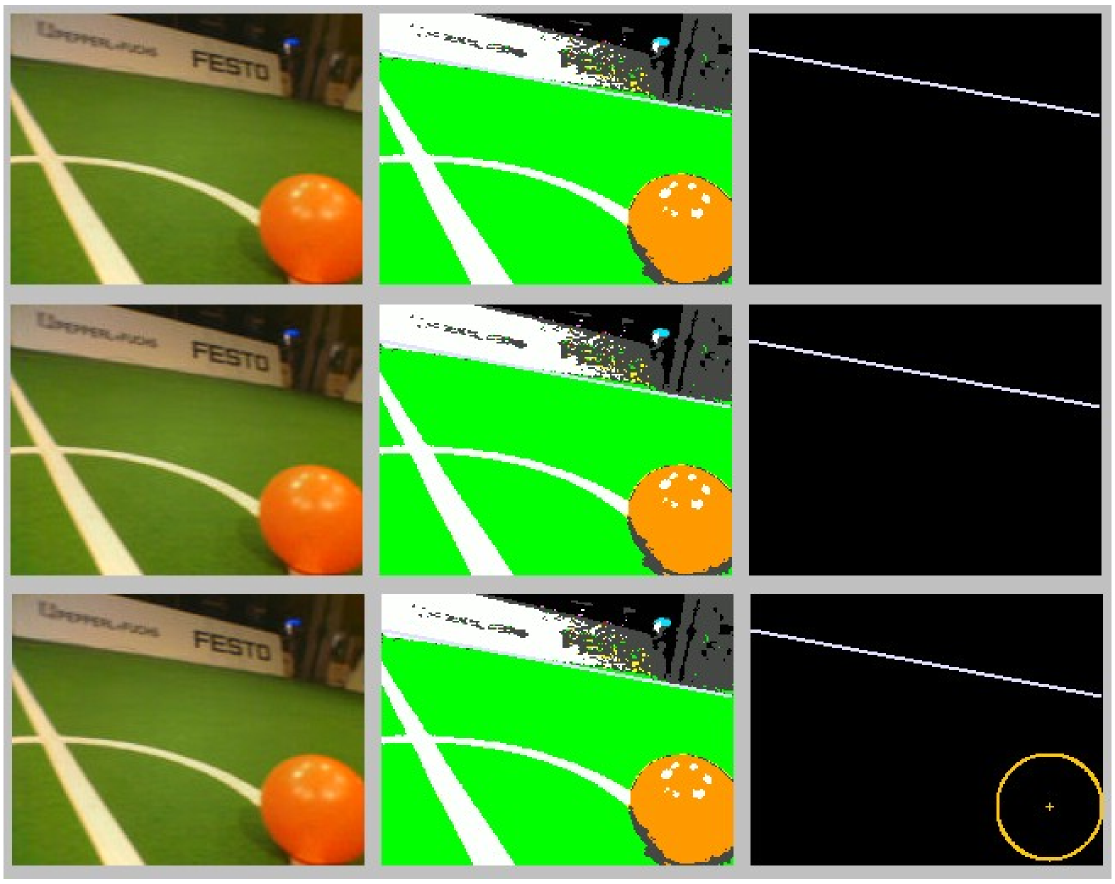


Figure 6.2 – The difference in ball recognition among the three image processors: the top snapshot is taken from the GT2004, the middle one from the SPQRL2006 and the bottom snapshot is taken from the Kouretes 2007 image processor.

Once an object has been recognized and, so, depicted in the perception window, there are also two parameters that are determined: the estimated *distance* between the robot and the particular object as well as the *angle* that the object is extended in respect with the robot. These parameters are to be compared among the three image processors. Figures 6.3 and 6.4 show the beacon and ball recognition, respectively, which is performed in a particular frame by all three image processors. The table that follows each figure (Tables 6.6 and 6.7) shows the values for the distance and the angle which are estimated in each case.

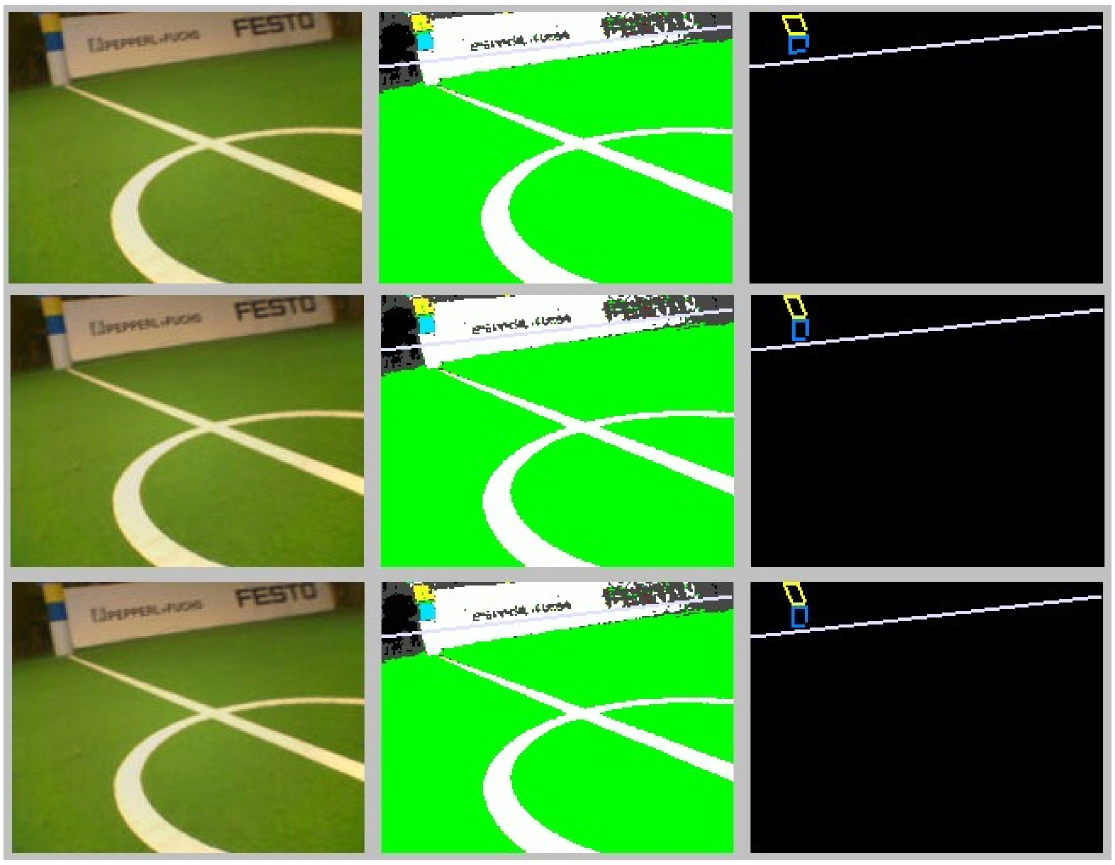


Figure 6.3 – The beacon recognition that is performed for a particular frame by the three image processors: the top snapshot is taken from the GT2004, the middle one from the SPQRL2006 and the bottom snapshot is taken from the Kouretes 2007 image processor.

	GT 2004	SPQRL 2006	Kouretes 2007
Distance (mm)	2509.9	2020.0	2019.1
Angle (rad)	1.8	1.8	1.8

Table 6.6 – The values for the distance and the angle of the yellow-blue beacon when recognized by the three image processors as shown in Figure 6.3.

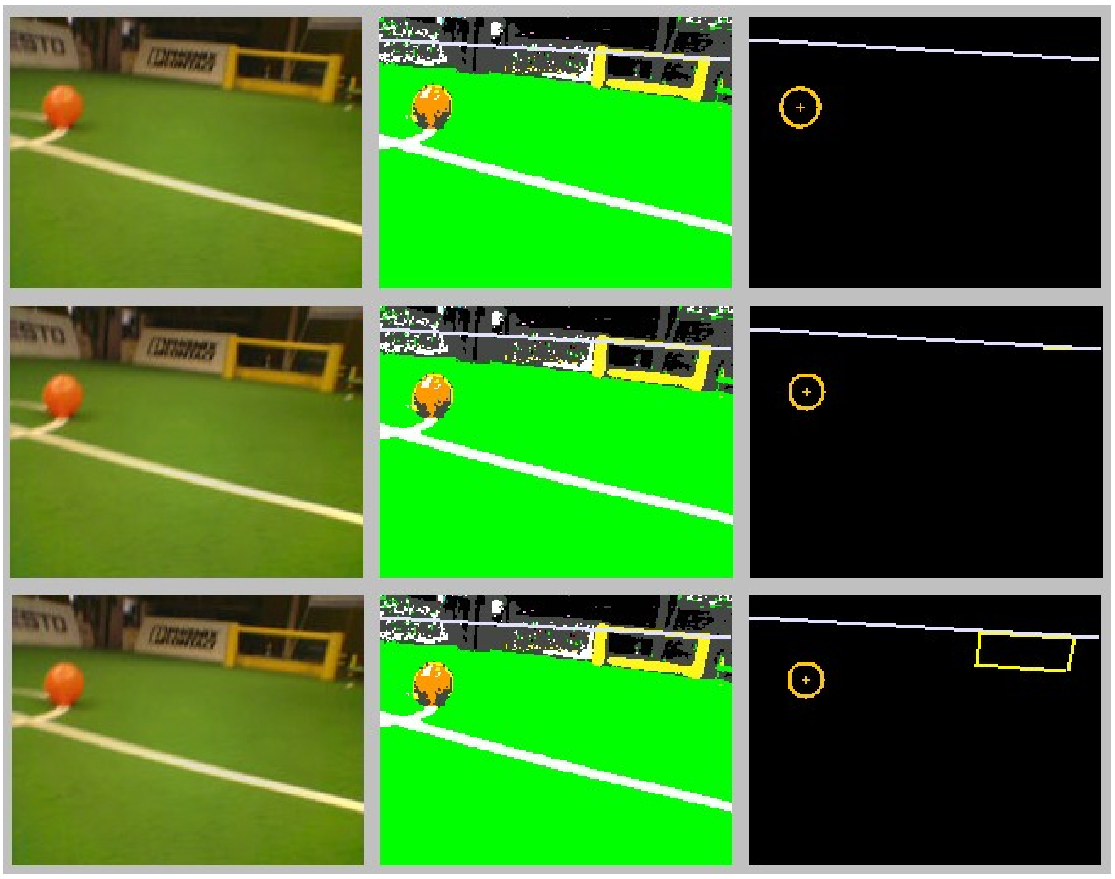


Figure 6.4 – The ball recognition that is performed for a particular frame by the three image processors: the top snapshot is taken from the GT2004, the middle one from the SPQRL2006 and the bottom snapshot is taken from the Kouretes 2007 image processor.

	GT 2004	SPQRL 2006	Kouretes 2007
Distance (mm)	852.4	510.2	507.5
Angle (rad)	- 0.0	- 0.2	- 0.2

Table 6.7 – The values for the distance and the angle of the ball when recognized by the three image processors as shown in Figure 6.4. The negative value of the angle in this case also depends on the robot' s head pose.

7 Conclusion

The last chapter of this thesis intends to sum up the work completed and achievements accomplished during the present project. In addition, some ideas for future research are presented.

7.1 Achievements

This thesis proposes a complete method for object recognition within the RoboCup four-legged league field. In particular, these objects are: the goals, the beacons, and the ball. Our approach is based on scanning the whole image, finding regions of the same color, building a histogram, and deciding whether the colored areas are part of an object of interest or not.

Especially for the 2007 field goals, this approach gives a satisfying solution for a problem that has been first stated this year. That is why the goal recognition' s performance could not be compared to previous related works. However, it has been used successfully by team Kouretes during the 2007 German Open RoboCup competition and during demonstration games in Thessaloniki, in October 2007.

The beacons were also a new kind of object within the 2007 RoboCup four-legged league field. In terms of color, the existing code of SPQRLegged 2006 and the German Team 2004 was modified accordingly, in order to obtain a way of comparing our work for beacon recognition to theirs. This modification was possible because of the fact that the new beacons were similar in appearance to the old ones. The comparison showed that the performance of the Kouretes 2007 beacon recognition was more robust when compared to the other teams' work.

Since our approach gave satisfying results for goal and beacon recognition, there was a thought of generalizing it also for ball recognition. The results showed that, in many cases, there was an improvement in ball recognition, especially when the ball happened to be relatively far from the robot.

As a consequence, the present work provides a complete image processing module to be used by the team Kouretes in the future.

7.2 Future Research

The present thesis project has undoubtedly made a step forward regarding the vision module used by team Kouretes for the RoboCup four-legged league. However, there are still prospects of improvement in some areas of the object recognition.

First of all, it would be really useful to reduce the image processing time. As it has already been mentioned, the scan lines that are used for object recognition are currently covering the whole image frame, a choice that consumes quite a lot of the processing time. So, a specific area for further research could be the finding of the best grid of scan lines, which will reduce the currently demanded image processing time without degrading the corresponding results of object recognition.

In addition, beacon recognition is where most false negatives (see section 6.1) occur because of their relatively small colored area. In the future, an idea for reducing the false negatives would be the participation of the beacons' white part in the recognition process. It is estimated that this could improve beacon recognition because the beacon' s total colored area would increase.

Finally, regarding ball recognition, it has been observed that the false negatives mostly occur when the ball is relatively close to the robot. Then, in some cases the radius is not calculated with accuracy. These cases are not so many as to influence the ball recognition in general, but the latter could be improved if these false negatives could be further reduced.

7.3 Conclusion

At this closing point, we draw a few general conclusions about object recognition within the RoboCup four-legged league field, while pointing out the experience which has been acquired through this project.

Object recognition in RoboCup is considered a very important task. Since the robot depends on knowing the position of the goals and the beacons within the field in order to localize itself, correct perception of these objects automatically becomes a necessary condition. In addition, correct perception of the goals and the ball determines a good playing performance by the robot.

Another interesting aspect of object recognition and robot cognition, in general, relates to the comparison between robot and human vision. One can easily realise that a sense like vision, which is generally considered a must for humans, is a much more difficult process when it comes to robots. Nothing is obvious and a lot of comprehension and work is needed in order to understand how human vision occurs, step by step, and then try to apply these principles onto robot vision.

Regarding what has been acquired through this thesis, the benefits can be counted in knowledge and experience. First of all, knowledge has been acquired in the fields of robotics, artificial intelligence, image processing (color segmentation and classification, several color spaces encodings), reverse-software engineering (comprehension of the SPQRLegged source code) and mathematics (geometric calculations and transformations). The knowledge of new programming tools, such as the Kouretes Image Viewer (see Chapter 5), is also to be mentioned.

Last but not least, since the vision module is the only one needed and used by the robot in order to be able to play soccer within the field, discussion and cooperation with the people in the team, who had undertaken modules highly related to vision, were essential. Thus, the experience of working within a team through this thesis has been invaluable in developing collaboration and communication skills.

Bibliography

[1] "RoboCup : What Is RoboCup", RoboCup Federation Official Web Site (RFOWS)

http://www.robocup.org/overview/21.html

[2] "RoboCup : Objective", RFOWS

http://www.robocup.org/overview/22.html

[3] "RoboCup : Brief of History", RFOWS

http://www.robocup.org/overview/23.html

[4] "Sony Four Legged Robot Football League Rule Book", RoboCup 2000

http://www.tzi.de/4legged/bin/view/Website/History

[5] "Sony Quadruped Robot Football League Rule Book", RoboCup 1999

http://www.tzi.de/4legged/bin/view/Website/History

[6] "Sony Four Legged Robot Football League Rule Book", RoboCup 2001

http://www.tzi.de/4legged/bin/view/Website/History

[7] "Sony Four Legged Robot Football League Rule Book", RoboCup 2002

http://www.tzi.de/4legged/bin/view/Website/History

[8] "Sony Four Legged Robot Football League Rule Book", RoboCup Technical Committee, RoboCup 2003

http://www.tzi.de/4legged/bin/view/Website/History

[9] "Sony Four Legged Robot Football League Rule Book", RoboCup Technical Committee, RoboCup 2004

http://www.tzi.de/4legged/bin/view/Website/History

[10] "Sony Four Legged Robot Football League Rule Book", RoboCup Technical Committee, RoboCup 2005

http://www.tzi.de/4legged/bin/view/Website/History

[11] "Sony Four Legged Robot Football League Rule Book", RoboCup Technical Committee, RoboCup 2006

http://www.tzi.de/4legged/bin/view/Website/History

[12] "Sony Four Legged Robot Football League Rule Book", RoboCup Technical Committee, RoboCup 2007

http://www.tzi.de/4legged/bin/view/Website/History

[13] "Teams 2006" – Standard Platform League Official Web Site

http://www.tzi.de/4legged/bin/view/Website/Teams2006

[14] "Team Kouretes Official Web Site"

http://www.intelligence.tuc.gr/kouretes/

[15] "German Team Official Web Site"

http://www.germanteam.org

[16] "German Team 2004 Team Report"
http://www.germanteam.org/GT2004.pdf

[17] SPQR-Legged RoboCup 2006 code
http://www.dis.uniroma1.it/~spqr/download/SPQRLegged2006.tgz

[18] S.P.Q.R. Legged Team Description Paper, RoboCup 2006
www.dis.uniroma1.it/~spqr/pub/SPQRL-TDP06-pre.pdf

[19] Sony AIBO Official Web Site (down as of March 2007)
openr.aibo.com

[20] Alex North,"Object recognition from sub-sampled image processing", Bsc Thesis, Department of Computer Science, University of New South Wales, Australia 2005
http://www.cse.unsw.edu.au/~robocup/2005site/reports05/north05-subsampled.pdf

[21] "rUNSWift 2005 Team Report"
http://www.cse.unsw.edu.au/~robocup/2005site/index.phtml

[22] "Cerberus Team Official Web Site"
http://robot.cmpe.boun.edu.tr/~cerberus/wiki/index.php?n=Main.HomePage

[23] "Cerberus 2006 Team Report"
http://robot.cmpe.boun.edu.tr/~cerberus/wiki/uploads/Downloads/Cerberus2006-TR.pdf

[24] RoboCup Simulation League, Wikipedia
http://en.wikipedia.org/wiki/RoboCup_Simulation_League

[25] RoboCup Small-size robot league picture
http://www.cs.cmu.edu/~robosoccer/image-gallery/small/ssl_game.jpg

[26] RoboCup Small-size robot league picture
http://www.dis.uniroma1.it/~iocchi/ART/img/sq1.jpg

[27] RoboCup Humanoid league picture
http://www.newscientist.com/blog/technology/uploaded_images/
humanoid_penalty-790891.jpg

[28] RoboCup four-legged league picture
http://www.smh.com.au/ffximage/2006/06/07/470_robocup08a,0.jpg