MASTER IN COMPUTER SCIENCE

Robots in health care
an AIBO controller for elderly entertainment

Capeqi, Theodhora

Award date:
2006

Awarding institution:
University of Namur

Robots in health care

An AIBO controller for elderly entertainment

Theodhora Capeqi

Thesis provided in sight of obtaining the Master Degree in Computer Science
Academic year 2005 - 2006

Abstract

As aging population in our modern society grows progressively, and as life cost grows too, we are faced to a generation of elderly population willing to live in their homes, in order to avoid stays in clinics or retirement houses. But often, even those who can afford living in medical environments suffer from solitude. This phenomenon appears because there is a shortage of nursing stuff, but also because old or disabled persons are more and more abandoned by their families. For those reasons, the use of robots in health care is having a huge success, since it brings a solution to the medical staff shortage as well as to the economic issue.

In respect to this context, this thesis presents the development of a controller, for the Sony AIBO (Artificially Intelligent roBOt) ERS-7 dog-like robot, projecting to make it find and fetch a newspaper in a domestic environment, with the aim of being used as an entertainment pet by the elderly.

As this work covers many topics of the robotics domain - from perception to self-localization, dealing with behaviours and motions - an overview of each of the concerned topics will be presented first.

These topics will also appear in the presentation of our contribution, with Perception being the most important part, since it constitutes the main source of knowledge for the robot and therefore influences its behaviours. In our work, Perception is based on image data processing, and behaviours are the result of the execution of some pre-defined plans designed through Petri Nets formalism.
Finally, a critical view over the relevant and reliable character of the robots' use in our daily life will be exposed.

Keywords: Image segmentation, robotic localization, behaviour control

Résumé

Dans notre société moderne, le taux de la population âgée s'accroît progressivement, de même que le coût de la vie. Ainsi, nous nous trouvons face à une génération de personnes âgées, qui désirent continuer à vivre chez eux, et éviter les cliniques et les maisons de repos. Mais souvent, même ceux qui peuvent se permettre de vivre dans un centre médical, souffrent souvent de solitude. Ce phénomène est causé d'une part par la pénurie de personnel médical, et d'autre part par l'abandon familial des personnes âgées ou handicapées.

Dans cet ordre d'idées, ce mémoire concerne le développement d'un contrôleur, pour le modèle ERS-7 du "robot chien" AIBO (Artificially Intelligent roBOt) fabriqué par Sony, qui lui permettrait de retrouver et de prendre un journal dans un environnement domestique, afin que le robot puisse être utilisé comme un animal de compagnie par les personnes âgées.

Ce travail recouvre plusieurs sujets liés au domaine de la robotique - allant de la perception à l'auto localisation, en utilisant des comportements et des mouvements - c'est pourquoi une vue d'ensemble sera présentée pour les sujets concernés.

Ces mêmes sujets se retrouvent aussi dans les sections où nous présentons notre travail, dans lequel la Perception est la partie la plus importante, vu qu'elle constitue dans notre cas, la principale source de connaissance du robot et qu'elle influence les comportements de ce dernier. Dans notre travail, la perception est basée sur les techniques de segmentation d'images, et les comportements sont gérés par l'execution de plans conçus sous forme de réseaux de Petri.
Enfin, nous exposerons une vue critique sur la pertinence et la fiabilité de l'utilisation des robots dans la vie quotidienne.

Mots-clefs: Segmentation d'images, localization robotique, contrôle de comportement

Thanks

First of all, we would like to thank the members of the RoboCare project at the CNR Institute and the SPQR team at the University "la Sapienza" in Rome, for welcoming and helping us.

We would especially like to thank the professors Daniele Nardi, Luca Iocchi, and the researcher Riccardo Leone who accepted us in their project and team, and who supervised us during our whole stay. We also thank them for offering us the opportunity to go to Rome and work on such an impassioning domain.

Special thanks also to the SPQR team as well, for their availability, their practical help, their supervision and their warm welcome.

Finally, we thank our thesis guide, Professor Pierre-Yves Schobbens who helped us realizing this thesis.

Contents

Abstract ii

Résumé iii

Thanks iv

Introduction 1

1 Computer Vision 4
- 1.1 Image acquisition . . . 5
  - 1.1.1 Vision systems . . . 5
  - 1.1.2 Color Spaces . . . 5
- 1.2 Image pre-processing . . . 8
- 1.3 Image segmentation . . . 8
  - 1.3.1 Threshold-based segmentation . . . 9
  - 1.3.2 Edge-based segmentation . . . 11
  - 1.3.3 Region-based segmentation . . . 13
- 1.4 Object recognition . . . 15

2 Image data interpreting 17
- 2.1 Introduction . . . 17
- 2.2 Autonomous robot localization . . . 17
  - 2.2.1 Bayes filters . . . 19
  - 2.2.2 Kalman filters . . . 20
  - 2.2.3 Markov Localization . . . 23
  - 2.2.4 Monte Carlo localization . . . 24
- 2.3 Object Location . . . 26
- 2.4 Learning . . . 28
  - 2.4.1 Machine learning . . . 28
  - 2.4.2 Environment mapping . . . 29

3 Decision Making 31

4 An overview of AIBO ERS-7 **34**
4.1 Hardware and Peripherals . . . 34
4.2 OPEN-R Software Development Kit . . . 36

5 Image Processing **37**
5.1 Color segmentation . . . 38
5.1.1 Color segmentation through transformation of the color distribution . . . 39
5.1.2 Summary . . . 41
5.2 Feature detection . . . 41
5.2.1 Horizon calculation . . . 41
5.2.2 Scan-lines construction . . . 42
5.2.3 Landmarks detection . . . 44
5.2.4 Newspaper detection . . . 50
5.3 Related Tools . . . 51
5.4 Related Problems . . . 52

6 Object Modeling **55**

7 Behavior Control **57**
7.1 A short overview on Petri Nets . . . 58
7.2 An overview on Petri Nets Plans . . . 59
7.3 Our plans . . . 62
7.3.1 Bi-colored lines plans . . . 63
7.3.2 Newspaper plans . . . 72
7.4 Related Tools . . . 75

8 Debate on robots **77**
8.1 Technical limits . . . 77
8.2 Beyond technique . . . 78
8.3 Conclusion on debate . . . 81

Conclusion **83**

Bibliography **90**

Annexes **90**

A Current controller **91**

B S.U.S.A.N: non-linear filter **93**

C Working context **96**

List of Figures

1 Overview of an autonomous robot workflow . . . 3

1.1 Color space [57] . . . 6
1.2 HSV color space [56] . . . 6
1.3 HSI color space [55] . . . 7
1.4 (a) segmentation of the image used to create the look-up table; (b) segmentation of the same image using the same look-up table under different lighting conditions [34] . . . 9
1.5 (a) the original greyscale image; (b) the histogram corresponding to the input image (X-axis represents intensity values, Y-axis represents the number of pixels); (c) image resulting by a segmentation with a global intensity threshold of 120 [45] . . . 10
1.6 (a) the original greyscale image; (b) the corresponding histogram; (c) image resulting by a segmentation with a global intensity threshold of 120 [45] . . . 10
1.7 (a) Original noisy image; (b) Image segmented with Roberts operator; (c) Same image segmented with Sobel operator [45] . . . 12
1.8 (a) The Original image; (b) Edge direction Segmentation of the original image with Prewitt; (c) Edge direction Segmentation of the original image affected by Gaussian noise [45] . . . 12
1.9 (a) Original image; (b) edge magnitude of the image segmented with Prewitt; (c) edge direction of the image segmented with Prewitt [45] . . . 13
1.10 (a) An example of region splitting; (b) Tree image splitting description [35] . . . 14
1.11 (a) Spherical tank image serving as template; (b) Arial picture of gas tanks factory; (c) resulting image of pattern matching using correlation [23] . . . 16

2.1 Example of the robot's odometry for a path in a given map. The accumulation of small odometry errors has large effects on the latter position estimations [66] . . . 18
2.2 Graphic representation of the probability distribution of the robot's estimated pose resulting from a combination of translation and rotation covariance of a Gaussian [33] . . . 21
2.3 An example of localization using multimodal beliefs and Kalman filters [33] . . . 22
2.4 An example of localization using multimodal beliefs and Markov filters [33]. . . . 23
2.5 An example of localization using MCL [33]. . . . 25
2.6 In the pinhole model, each point in the real world is connected to a point in the image plane by a line passing through the central point c which refers to the camera [33] . . . 26

3.1 (a) Deliberative paradigm; (b) Reactive paradigm; (c) Hybrid deliberative/reactive paradigm . . . 32

4.1 Front view of AIBO's ERS-7 hardware and peripherals [58], cited in [50] . . . 35

4.2 Rear view of AIBO's ERS-7 hardware and peripherals [58], cited in [50] . . . 35

5.1 Our implementation of the robot's working steps . . . 37
5.2 (a) an image captured by the color camera of AIBO; (b) typical color distribution of the H component for this image; (c) the color distribution of H, obtained after application of the transformation function [20] . 40
5.3 The horizon-line is obtained by intersection of the projection plane P and the horizontal plane H [65] . . 42
5.4 Construction of grid-lines according to the horizon [65] . . . 42
5.5 The field of work is composed by three different bi-coloured lies and a newspaper at the end of the blue-white line . . . 44
5.6 (a) points in the Cartesian space, (b) lines in the Hough domain corresponding to the points in the Cartesian space [33] . . . 47
5.7 (ρ, θ) are the parameters of a line expressed in polar coordinates [33] . . . 47
5.8 (a) points in the Cartesian space, (b) sinusoids in the Hough domain corresponding to the points in the Cartesian domain [33] . . . 48
5.9 Remote debugging tool: the robotControl window . . . 51
5.10 Perception simulation: the Ipmonitor set of windows . . . 52
5.11 Blue cast defect [34] . . . 53
5.12 Motion Blur occurrence [34] . . . 53
5.13 Stripes of different luminosity occurrence [39] . . . 54

6.1 Full line = the actual straight walk of AIBO, dotted shape = expected straight walk of AIBO . . . 55

7.1 (a) a place; (b) a place with a token; (c) a transition; (d) an arc labeled with the weight 5 ;(e) a unitary arc 58
7.2 Representation of an input and an output place . . . 58
7.3 (a) Initial marking of a water molecule creation plan, (b) Final marking of the same plan [17] . . . 59
7.4 Representation of an ordinary action . . . 59
7.5 Representation of a sensing action . . . 60
7.6 A sequence of two plans [69] . . . 60
7.7 (a) A conditional structure; (b) A loop structure [69] . . . 61
7.8 (a) A fork structure; (b) A loin structure [69] . . . 61
7.9 An interrupt structure [69] . . . 62
7.10 Plan executed to follow the lines when the goal is to find the newspaper . . . 63
7.11 Plan where the robot searches for a bi-colored line by moving its head during 6 seconds . . . 64
7.12 Plan executed when the robot recognizes a yellow-white line . . . 65
7.13 A simple model of a robot . . . 66
7.14 Variation of the robot's trajectory when one of the translations changes . . . 67
7.15 (a) Translation; (b) Rotation; (c) Rotation and Translation . . . 69
7.16 Kinematic model for Line following . . . 69
7.17 if $b < 0$, $A * \exp_{bt}$ behaves like a decreasing exponential stabilizing around the x-axis, if $b > 0$. $A * \exp_{bt}$ behaves as an increasing exponential moving away from the x-axis . . . 71
7.18 (a) behaviour of the system when $b < 0$; (b) behaviour of the system when $b > 0$ . . . 72
7.19 Plan for going to the newspaper . . . 73
7.20 Plan where the robot searches the newspaper by moving its head during 6 seconds . . . 74
7.21 Jarp windows . . . 75
7.22 (a) A plan giving motion commands; (b) corresponding Pet file . . . 76

8.1 AIBO equipped with a newspaper container 84
8.2 (a) the right image shows the result of segmentation for the left image captured while two source of lights were available; (b) when only one source of light is available for the same color-mapping table the walls are not recognized! 86

A.1 Placing the lines 91
A.2 Placing the newspaper 91
A.3 Placing the robot 92

B.1 Application of SUSAN on a dark and white image 93
B.2 the red area is the USAN space associated to each nucleus 94

List of Tables

1.1 Summary of image segmentation methods and their important features. 15

2.1 Summary of robotic localization methods 25

Introduction

Background and context

Nowadays, elderly and disabled persons, suffer more and more from solitude. There are two reasons to this phenomenon. At one side, these people are more and more abandoned by their families, but as they can't afford living in retirement houses or medical clinics, they live alone in their homes. At the other side, there is a shortage in nursing staff, therefore even those who can afford medical help, don't always get the service.

Then robots, which have always been used as "cheap slaves", seem to fit for a solution to these issues. For thirty years, researchers in Artificial Intelligence (AI) have been working hard to elaborate intelligent machines, capable of being part of a solution to critical problems in our modern society. The new fashion is the manufacturing of more "friendly" robots, serving not only as "slaves" but also as "companions". Indeed, many research centers, be they in Universities or private institutions, work on the manufacturing or implementation of human or animal-like robots, in order to be used as companions or to assist humans (see articles in [28], [49]).

One example of such a robot is AIBO, the dog-like robot manufactured by the Digital Creature Laboratory for Sony in Japan. AIBO was initially created to be a sophisticated toy, enriched with capabilities such as playing with a ball, understanding voice commands, recognizing its owner etc. Hence, this robot has also been serving scientific purposes since its creation in 1999 [68], ranging from promoting research in AI through challenging competitions like Robocup [47], to serving as an entertainment pet for the elderly [16] as described in Annex C.

Objectives

The aim of our internship in Rome, was to implement a new task for AIBO, consisting in making it walk around in a domestic environment from one room to another, with the goal of finding and fetching a newspaper.

Besides its funny appearance, this project was serve two purposes:

- AIBO will be used as an entertainment pet by elderly persons, be that in their own house or in a retirement house or other medical environment. In this way we bring a solution to the "lonlyness" phenomenon to which aged persons are faced.

- it was planned to participate with this work at the open challenge of the Robocup@Home competition (see Annex C),

However due to the short amount of time allocated for realizing the project (four months), and the delay accumulated due to the acquisition of the basic knowledge (one month) on the robotics domain (which was initially unknown to us), the objectives of our project were reduced to making AIBO navigate in one single room, and only recognize the newspaper and approach it.

Our practical work mainly focused on two parts: Vision, since the camera data was the principle source of knowledge that we used, and Decision Making or Behaviour Control, since the robot had to react.

This thesis has two main goals. First, we provide an overview on machine vision, robot localization and mapping, machine learning and decision making, and their corresponding concepts and methods. Then we present useful and representative applications of some mathematical and probabilistic theories (including some that we encountered during our studies at the University of Namur).

Outline

Figure 1[1] illustrates the structure of this thesis according to the main work flow of an autonomous intelligent robot, where the ellipses represent sets of data serving as input or output at the different steps, while the rectangles represent the main actions performed in the current step.

We start with the sensing step where data coming form robot sensors are processed in order to retrieve information either about the robot state or its environment. In our case we only focus on image sensors, providing robots with vision capability. That is why Chapter 1 is entirely dedicated to Computer Vision, which deals with image acquisition and information extraction. The low-level information extracted by the Sensing process, is used by the robot, which combines it with its high-level knowledge and extracts information on its state or position, or on the position of surrounding objects. We classify this process as the Interpreting process, since low-level data are interpreted to obtain high-level information. Chapter 2 presents this step. Chapter 3 deals with the main existing alternatives for the implementation of the Decision making step[2], where actions are performed producing motion requests for the robot actuators.

In the subsequent chapters we start the part of the thesis which more specifically concerns our work, beginning with the AIBO's hardware and software presentation in Chapter 4. In Chapter 5, we present our image processing and interpreting implementation, where some sections are dedicated to the presentation of the related tools and encountered problems. Chapter 6 is specific to our work and is very important, since it corresponds to the implementation of the object's model and keeps track of the characteristics of all the perceived objects. In Chapter 7 we expose our implementation of the decision making process and the tool that was used. Then, in Chapter 8 we enter a debate over the ethical, economic and

[1] The schema in figure 1, is inspired by the implementation architecture of the German Team. Indeed we adopted their implementation architecture (see [65]), as well as SPQR and many other teams taking part in the RoboCup Soccer competitions (see Annex C).

[2] Since the Acting step is a low-level part dealing with robot actuators and their motions, and since we only used existing low-level actions (those implemented by SPQR), no Chapter is dedicated to this step.

scientific questions raised by the robotics field.

Finally, we conclude, with a presentation of the project's results, its possible improvements, and future development.

Annex A contains a sort of user guide, so that the robot can perform the implemented tasks properly. Annex B presents a specific image pre-processing method that we studied during our internship but that we didn't use. Finally Annex C gives an overview of the research groups and working fields, which were directly or indirectly involved in our practical experience.

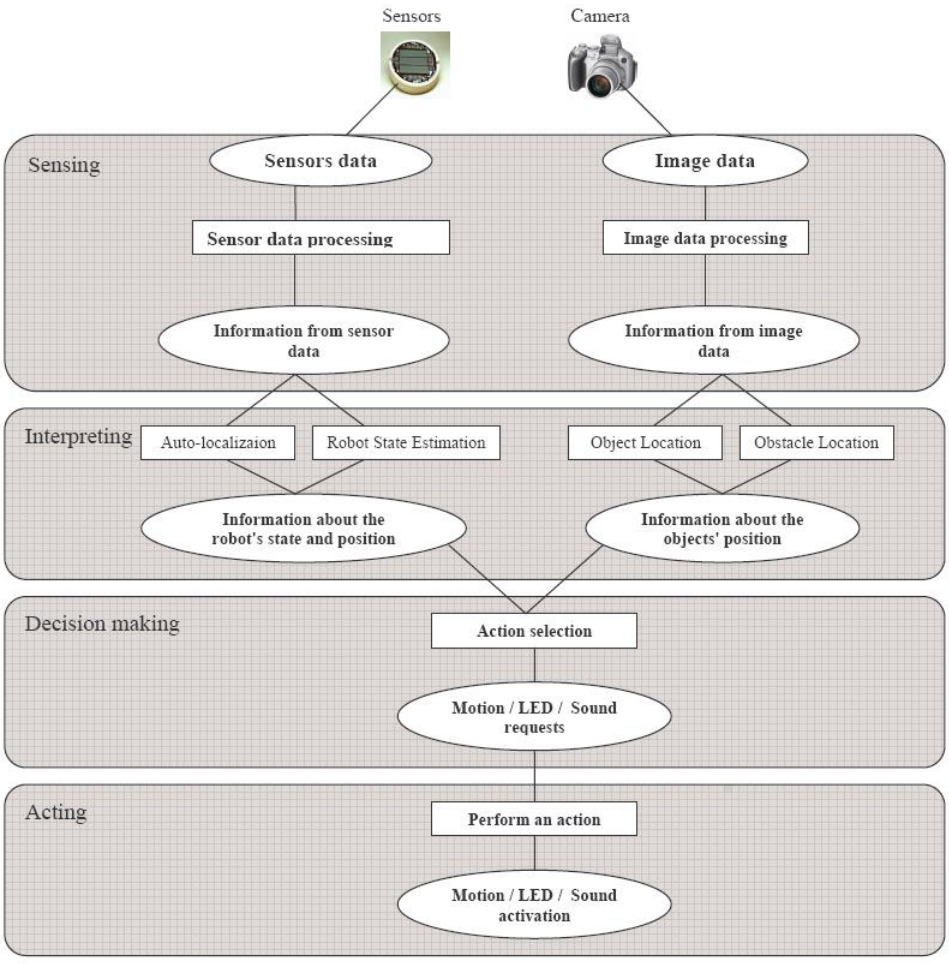


Fig. 1: Overview of an autonomous robot workflow

Chapter 1

Computer Vision

Computer Vision is a discipline that tries to imitate biological vision by making computers "understand" an image: i.e. identifying and locating some features contained in it. Computer Vision serves two main purposes; either providing image information to a human operator (e.g. medical imagery for cancerous cell detection), or to support high-level processes for autonomous machines (e.g. navigation of a robot in its environment).

When Computer Vision is applied on vision-based autonomous robots or to vision based inspection systems, then we talk about *Machine Vision*. The main task for Machine Vision is image processing, commonly divided into low-level and high-level processes.
Low level vision consists of four processing steps:

- *Image acquisition*: to capture images by a vision sensor;
- *Image pre-processing*: to improve image quality for a better feature extraction in the future[1]. This step allows to recover from defects in data caused by stripes, intensity differences, darkness or transparency appearing in the image;
- *Image segmentation*: to separate objects of interest from the background on an image;
- *Object recognition*: to extract properties of objects such as their size or their position in the image.

High level image processing steps try to imitate the human cognition system by recognizing or comparing features in images, and our ability to make decisions, by taking into account information contained in the image. These steps are based on the robot's prior knowledge about the environment and on its goals and plans which describe how to achieve these goals. The set of goals and prior knowledge constitute the high-level knowledge of the robot.
In the following sections of this chapter, we will give an overview on the four low-level image processing steps.

[1] This process is optional since it depends on the image quality.

1.1 Image acquisition

1.1.1 Vision systems

Image acquisition depends on the vision system. We generally distinguish three types of vision systems [32]:

- *Monoscopic systems.* They are composed by a single camera, providing only partial information on the scene (because of the limited range of vision of the camera) depending on its orientation, and representing only two dimensions.
- *Omni-directional systems.* They provide 360° images of, and can be obtained by combining a normal camera with a mirror or a so-called fish-eye lens.
- *Stereoscopic systems.* They are composed by two or more normal cameras providing two different views of the same scene, and therefore providing depth information.

Image acquisition also depends on the characteristics of the camera. Indeed, based on the camera we use, we can obtain gray-scale or color images, and bad or good quality images.
When working in color-coded environments (i.e. color is an important factor for differentiating objects from each other), it is important to perform a color calibration of the camera. *Color calibration* is the process of mapping raw camera pixel values to color labels, such as green, yellow or blue. Of course, we need an expert that knows how to calibrate the colors, and we need to choose between different color representations, which are described in the following sub-section.

1.1.2 Color Spaces

Color is the most important information to be extracted from an image. Furthermore, one of the biggest challenges when working in color-coded environments is to develop an architecture for robotic vision that is independent of the present lighting conditions (see section 1.2 on image processing). In fact, as each object in the environment is distinguished according to its color; e.g. in the RoboCup competitions, the ball is orange and one of the goals is yellow. But if the lighting conditions change, the yellow goal can be confused with the orange ball. This implies that the part of the system dedicated to image processing, should not be sensitive to lighting changes, so that color classification will be independent of the external environment.

As lighting in a domestic environment is quite variable, autonomous color classification was also an important requirement for our work. But before giving a description of how this problem is currently solved, we need to first introduce the notion of "color spaces".

The *Intersociety Color Council* has created a *Universal Color Language*, which classifies colors into color spaces. A *color space* is a three-dimensional coordinate system where colors are represented in terms of measurable values. Dozens of color spaces exist but the most known color models are RGB, YUV and HSV.

RGB (Red Green Blue), is a set of primary colors used by video cameras, televisions and PC monitors. When combined, these colors can create any other color on the spectrum. Such a space can be represented by a three-dimensional system of Cartesian coordinates (figure 1.1). RGB is not efficient for real world

pictures, since it does not handle luminosity variation, as opposed to YUV.

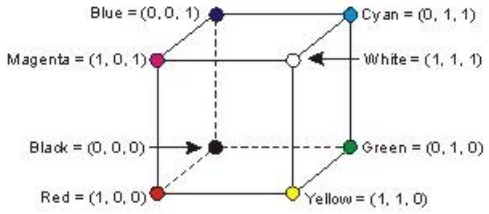


Fig. 1.1: Color space [57]

YUV[2] is obtained as a linear transformation of RGB (see transformation formulas in [13]), where Y gives information on the luminosity of the pixels, while U and V give information on their color. This color space allows the representation of color information from the luminance component (which we perceive as brightness), and is used worldwide for television and motion picture encoding standards (e.g. PAL) or for JPEG/MPEG compression. However, there is interdependence between the three components, i.e. when the luminosity on a pixel changes, its color changes too. So, this model is equally not adapted for systems where changing lighting conditions are frequent.

There also exist more intuitive color spaces, modeled on how human beings perceive colors; HSV and HSI models which are both non-linear transformations of the RGB color cube. In *HSV* (figure 1.2), H stands for hue and represents the color, S stands for saturation and represents the "vibrancy" of the color, wile V stands for value and represents the brightness of the color. This color space is also known as HSB (Hue, Saturation, and Brightness).

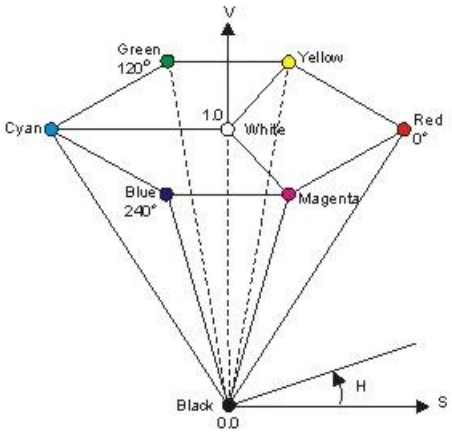


Fig. 1.2: HSV color space [56]

[2]The YCbCr and YPbPr color spaces are derived from it and are sometimes inaccurately called YUV.

On the other hand, in *HSI* (figure 1.3), hue gives the color type, saturation gives the degree of color contrast, and intensity gives the color brightness. It is similar to HSV but better reflects the intuitive notion of "saturation" and "lightness" as two independent parameters. As you can see with the different figures, HSI fits better for image processing which operates on the luminosity component, while HSV fits better for manipulations on hue and saturation.

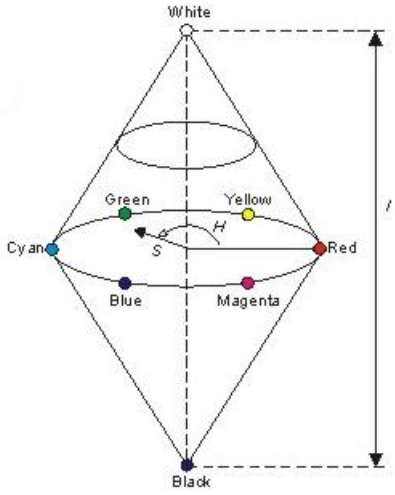


Fig. 1.3: HSI color space [55]

1.2 Image pre-processing

When image quality is poor, performing pre-processing useful. Generally, image pre-processing consists in image sharpening (i.e. de-blurring fuzzy images), image normalization (i.e. normalizing image brightness, contrasts or color intensity), noise reduction, feature suppression or enhancement, and edge detection.
There are different ways of classifying image pre-processing methods in categories (see [63], [24]). Here we only present two main categories which differ by the result they provide:

- *Image degradation suppression* methods, such as brightness transformation or Discrete Fourier transform [24], can enhance images suffering from non homogeneous object illumination.
- Then we have the linear (or smoothing) and non-linear *reduction filters*. The former, are very easy to implement due to their mathematical simplicity and are efficient in reducing noise, but they also blur sharp edges. Meanwhile the latter reduces noise and still preserves the structures of the image, i.e. texture and edges have still good quality.
  Among the non-linear filters in literature we can find: Wavelet-transform [14], Local-information based transform [6], Gradient Inverse Weighted [10], Weighted Median Filter [8], and SUSAN filter (see Annex B).

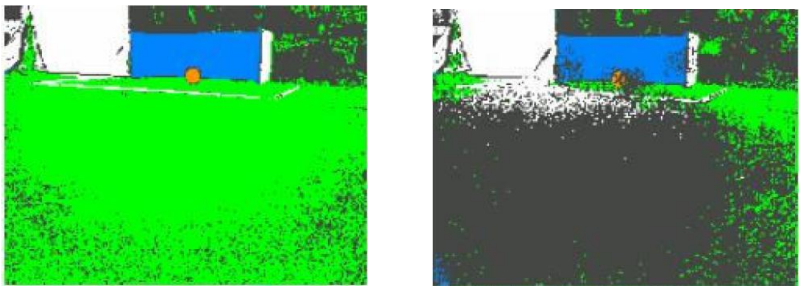
In the literature edge detection, is generally presented among the pre-processing methods. In this thesis, in the pre-processing step, we only consider image enhancement methods, while edge detection techniques will be separately exposed in the segmentation processing step described in the following section.

1.3 Image segmentation

Image Segmentation is the processing step that consists of separating features from background in the image. To do so we must be able to differentiate their color classes. Thus, to perform correct segmentation we must associate the correct color class to each pixel of the image.

Assigning to each pixel in the image, a color class among a predefined set of color classes, is possible through *color calibration* of the camera, which consists on storing the pixel-color associations in specific structures, called *look-up tables* (also known as color tables). Color calibration can either be performed manually, or automatically.

Manual calibration consists of manually creating the look-up table off-line, e.g. before the robot starts its task. There are two drawbacks to this method; it is time-consuming (since it is manual) and not robust to lighting changes. In fact, suppose that lighting changes while the robot is executing its task, in this case the look-up table does not fit anymore as showed in figure 1.4.



(a) (b)

Fig. 1.4: (a) segmentation of the image used to create the look-up table; (b) segmentation of the same image using the same look-up table under different lighting conditions [34]

Other calibration approaches have been proposed to overcome these pitfalls, but none of them can offer both automatic and adaptive calibration. Two groups appear in the literature [21]:

- *Off-line automatic approaches.* Here a representation of the colours of the environment is generated off-line (e.g. before the robot is switched on and starts its task). This generation can either be performed by an external program or by the robot itself. However, the same colour representation is used during the whole task, since re-running the automatic calibration during the task (to adapt to lighting changes), is very difficult to achieve.
- *On-line adaptive approaches.* These methods are robust to illumination changes since the colour representation is dynamically adapted during run time, as in [21]. However, the computational resources or time requirements for these methods are significant.

Once this colour association is performed and image is pre-processed if necessary, we can enter the segmentation process. Generally the image segmentation procedures are regrouped in classes according to the segmentation "attribute" they are based on (see [54]). Namely, the threshold-based methods perform segmentation according to the pixels, edge-based methods according to edges in the image, region-based methods according to regions. Those methods and the corresponding examples are presented in the following three subsections.

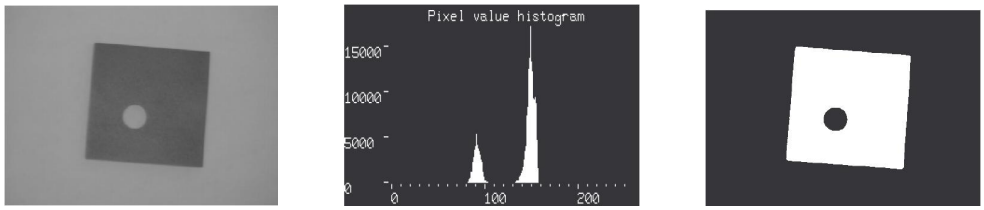
1.3.1 Threshold-based segmentation

The *threshold-based or thresholding* approach is conceptually the easiest segmentation approach, and is based on pixel information [31].
Their objective is to separate the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background.

First, the histogram of the intensity levels in the image is created (where the X-axis represents the intensity values and the Y-axis the number of pixels corresponding to these values). Then the histogram is analyzed in order to choose the intensity thresholds. this analysis results on separating the histogram in several classes, those having values under a threshold and those having values above it. To all pixels

belonging to one of this classes is associated the same colour class.

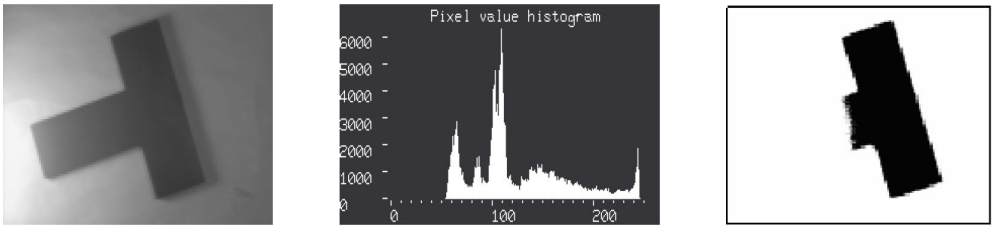
Thresholding is efficient only when the difference between the intensity levels of objects and background is significant, and they have problems to cope well with image noise as well as with blurred edges. Nevertheless if the image is previously corrected and noise is reduced by pre-processing methods, this approach gives good results.
Thresholding methods can be divided in two main classes; static and dynamic thresholding. When thresholds are fixed off-line (i.e. not during run time), we talk about *static threshold*. An example is given in figure 1.5.



(a) (b) (c)

Fig. 1.5: (a) the original greyscale image; (b) the histogram corresponding to the input image (X-axis represents intensity values, Y-axis represents the number of pixels); (c) image resulting by a segmentation with a global intensity threshold of 120 [45]

The threshold example in figure 1.5 is called the global threshold, which is distinguished from the local or adaptive threshold, which takes into account the illumination changes in the image. Figure 1.6 shows the limits of the global threshold in presence of a not uniformly illuminated image.



(a) (b) (c)

Fig. 1.6: (a) the original greyscale image; (b) the corresponding histogram; (c) image resulting by a segmentation with a global intensity threshold of 120 [45]

The use of static threshold can be extended to colour images [39], where a pair of thresholds is fixed for each colour. But the static threshold implies that, while fixing the thresholds, there is a prior knowledge about the content of the image. Indeed, for greyscale images we knew that the object is dark in a bright background, and for colour images we know which colours may be present in the image, in order

to fix the corresponding thresholds. Sometimes this prior knowledge is not available. Another drawback of static thresholding, consists in its sensitivity to light changes. Indeed figure 1.6 gives a good example. Imagine that the scene in the image in figure 1.6(a) had uniform illumination when we set its threshold to 120. If a source of light reflects on the same scene during run time, as we can't change the threshold, the object of interest (here the T letter) risks to not be recognized. These drawbacks are overcome by the dynamic threshold.

Dynamic threshold is thus performed on-line (i.e. during run time). The histograms related to the chromatic components (e.g. hue, saturation and colour for HSV), are generated first. Then they are scanned in order to compute the best threshold, according to the following algorithm:

- calculate the two absolute maxima of the histogram, which distance between them is inferior to a pre-defined value,
- calculate the minimum value between the two maxima, and set this value as the new threshold,
- apply the first three steps to the two portions of the histogram separated by the previously fixed threshold, until it becomes impossible to repeat the process.

Many algorithms based on dynamic threshold exist and give excellent results, but are computationally too expensive to be applied in practice [39]. Furthermore as no prior knowledge about the image is used and it is automatic, we have no guarantee that the objects extracted after the threshold process, actually correspond to objects of the environment. Many thresholding techniques can be found in literature, and a recent survey and evaluation of them is presented at [36].

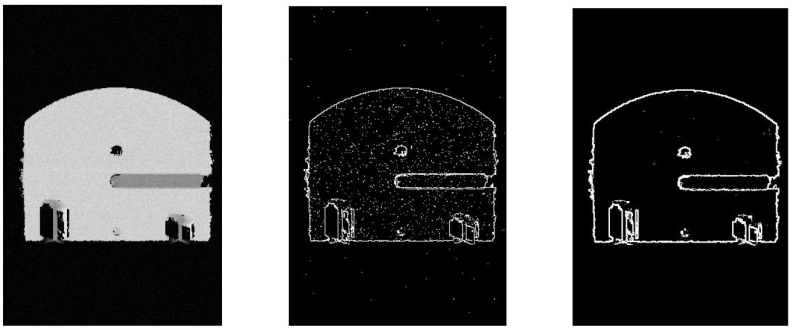
1.3.2 Edge-based segmentation

Edge based methods try to find edges appearing in the image, i.e. pixels that are on the boundary between two regions differing from each other in their chromatic component values (e.g. intensity or colour). To do so, we must apply edge identification operators also known as edge-detectors.
There are four criteria that can be taken into account when comparing edge-detectors [9]:

- Good detection, i.e. little false positives and false negatives for edges.
- Good localization, i.e. the position of the edge in the segmented image, must be as close as possible to its position in the original image.
- Single edge identification, i.e. only one segmented edge must correspond to an original edge.
- Computationally efficient, i.e. the edge detection algorithm must be as rapid as possible

Among the most common edge detection operators in literature we find: Roberts Cross edge detector, Sobel edge detector, Canny edge detector, Laplacian of Gaussian (LOG) operator, and the Prewitt operator.

The *Roberts Cross* [46] edge detector is the cheapest one in computational terms, but is very sensitive to noise (figure 1.7) and weekly recognizes edges unless they are very sharp in the original image. Furthermore, the object location in the segmented image is shifted. While the *Sobel operator* [22] is

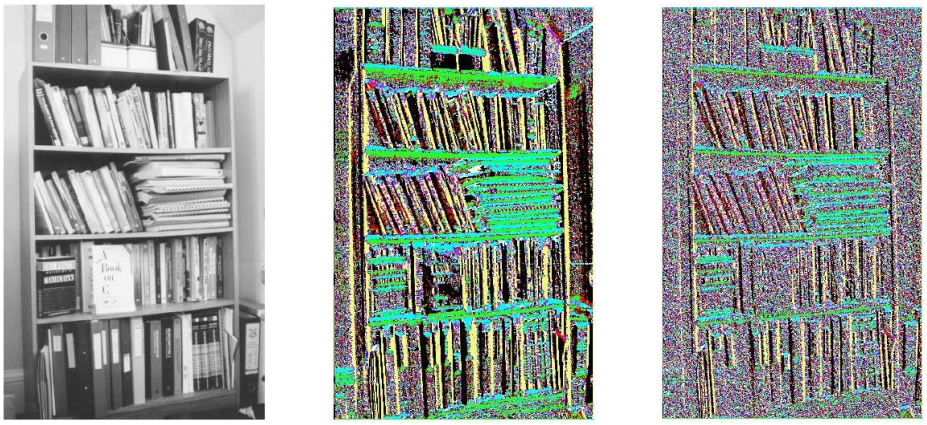


(a) (b) (c)

Fig. 1.7: (a) Original noisy image; (b) Image segmented with Roberts operator; (c) Same image segmented with Sobel operator [45]

computationally slower but less sensitive to noise and produces sharpest edge recognition.

A variant of Sobel is the *Prewitt operator* [25] which works almost the same way as Sobel, but is sensitive to Gaussian noise (figure 1.8) and is non isotropic (i.e. its diagonal edges are more accentuated than the vertical or horizontal ones (figure 1.9)), at the contrary of LOG [12] which applies equally to all directions.



(a) (b) (c)

Fig. 1.8: (a) The Original image; (b) Edge direction Segmentation of the original image with Prewitt; (c) Edge direction Segmentation of the original image affected by Gaussian noise [45]

LOG is a very expensive in computing time and it is very noise sensitive, producing thick edges and loosing thus thin structures of the image. However at the contrary of the three previous operators, LOG

tries to join the close edges in order to form a segment, and this explains its elevated computational time. The *Canny operator* [9] was provided by Canny as an optimal edge detector according to the first three criteria for a good edge detector, and provides thus good results in their respect. But as LOG, it also tries to join edges, and is computationally very expensive.

(a) (b) (c)

Fig. 1.9: (a) Original image; (b) edge magnitude of the image segmented with Prewitt; (c) edge direction of the image segmented with Prewitt [45]

1.3.3 Region-based segmentation

The most common region-based methods are the *region growing* ones, and are complementary to the edge-based methods, because they take into account spatial information of the image (i.e. spatial relationship between objects in the image) [3].
Fu and Mui proposed such a region growing method [30], which consists of choosing one or more initial pixels, called seeds, analyze the neighbouring pixels, and associate them to regions according to a predefined homogeneity criterion[4]. Region growing methods are classified either as region merging, or as region splitting.

In *region merging*, adjacent regions are compared, and merged if they are close enough in some property. Such algorithms begin by segmenting the image in small regions. Then all adjacent regions satisfying the merge criterion within a given "tolerance" threshold are merged. The process is stopped when there is no pair of regions left which fulfil the merging criterion. An algorithm based on region merging was proposed by Beaulieu and Goldberg [5].
With this kind of procedure, shifting between the real border and the segmented one is possible, since incorrect pixels can be added to a small region. The result also depends on the search strategy employed among the neighbours, on the seed chosen and the "tolerance" threshold chosen. Such values can either

[3]Segmentation images resulting from edge-based methods and region growing methods are usually not the same. Region growing techniques generally provide better results, for noisy images where edges are extremely difficult to detect.

[4]The homogeneity criterion is based on the choice of some threshold values, which is problematic since thresholds depend on image data.

be set manually by the user (but then he has to test different values until he finds the optimizing ones), or automatically (e.g. determining it from peaks in a histogram).
The *Seed Region Growing* algorithm where seeds are manually set is presented in [44], while the *Improved Seeded Region Growing* algorithm, which is non-parametric (i.e. no seeds or thresholds must be tuned by the user), is presented in [37].

In *region splitting*, large non-uniform regions are broken up into smaller areas which may be uniform. These algorithms begin from the whole image, and divide it up until each sub region is uniform and satisfies the segmentation conditions. The usual criterion for stopping the splitting process is when the properties of a newly split pair do not differ from those of the original region by more than a threshold.

Such a method is illustrated in figure 1.10(a), where the image I is primarily divided into four different regions (here represented by quadrants) according to pixels' properties. But I_4 is split a second time since not all the pixels within it are similar. And finally we can describe the splitting by a tree structure as presented in figure 1.10(b).

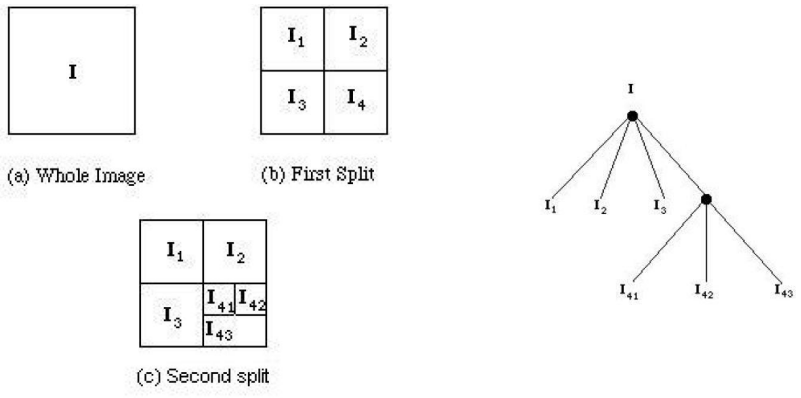


Fig. 1.10: (a) An example of region splitting; (b) Tree image splitting description [35]

In such methods it is difficult deciding where to make the partition, and in most cases splitting is used as a first stage of a hybrid split and merge method, which is practiced quite often and consists on an alternating mix of merging and splitting steps.

Another common group of region based methods is the *clustering*, which consists of classifying objects of the image according to some specific properties of these objects. Basically the pixels are mapped to feature vectors in a feature space, and then statistical methods are used to detect clusters within this space.
Here, each cluster represents a set of closely associated features in the feature space. The simplest and most popular clustering algorithm is the k-means technique introduced by J. McQueen and E.Forgy.

The main drawbacks of clustering techniques rely on the ambiguity of the clusters extension in the feature space and the high computational cost of the statistical methods used for it.
While the performance of region growing methods, depends on the execution order of the operations to be performed [38], as described in table 1.1, which recollects the different segmentation methods overviewed in this image segmentation section.

Category	Method	Noise	Computational cost
Threshold-based	Static threshold	noise sensitive	efficient
	Dynamic threshold	noise sensitive	expensive
Edge-based	Roberts	noise sensitive	cheep
	Sobel	robust	expensive
	Canny	noise sensitive	expensive
	LOG	robust	expensive
Region-based	Region growing	robust	variable
	Region splitting	robust	variable
	Clustering	robust	expensive

Table 1.1: Summary of image segmentation methods and their important features.

1.4 Object recognition

Object recognition in the literature often refers to one of the following three problems [43].

The *Specific Object Recognition* (e.g. face or landmarks recognition), which consists of recognizing an object in the image as being one of the specific objects for which shape and appearance is known by the robot. For example, at the end of the previous image-processing steps, the robot disposes of a few properties of the object such as colour and edges. This information can be sufficient to identify which is the object in the image, if object characteristics are known à priori.
Indeed, by knowing precisely the colour, the real size (in centimeters), the position and the shape of the object in the environment, it is possible to compute its size from its image size (in pixels) and its distance and position with respect to the robot's body (see section 2.3).
Furthermore, if the shape of a feature can be described by a mathematical expression, we can use methods such as the Hough Transform (see section 5.2.3) to generate its parameters (i.e. angle and distance). The Hough transform is able of pointing out lines, curves, or circles in the image[5].

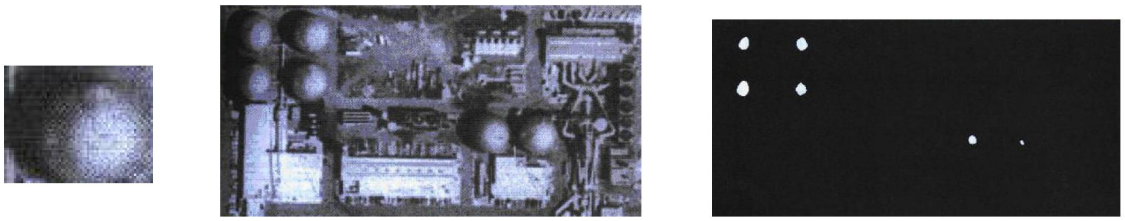
The other problem referred in literature is the *Generic Object Classification* (e.g. optical character recognition), where the task consists of labeling the object of the image as belonging to one of several known object categories or classes of objects (e.g. "car" or "chair").

Then we have the *Object Detection* problem, where the system possesses references to objects or categories of objects, which will be mapped with objects in the processed image (typically showing natural

[5]In literature Hough is sometimes classed among the model-based segmentation class of methods [15].

indoor or outdoor scenes).

One technique for object detection is the pattern matching using correlation, where the goal is to find every instance of a specific object in the scene by applying a special template, i.e. an image of the object of interest. Suppose for example that the spherical gas tank image in figure 1.11(a) is an example template.



(a) (b) (c)

Fig. 1.11: (a) Spherical tank image serving as template; (b) Arial picture of gas tanks factory; (c) resulting image of pattern matching using correlation [23]

In order to locate all possible gas tanks in the image presented in fig.1.11(b), the gas tank template is applied to this image. The resulting image in figure 1.11(c), shows in white all groupings of pixels that correlate with the template, while groups of pixels that do not correlate with the template are black. This problem, as well as the generic object classification, is always coupled with the visual feature learning process [43][6].

For visual recognition, two major classes of techniques can be identified [43]; *the model-based* methods which use the object's geometric models to extract features, and the *appearance-based* methods, which extract features directly from example images without explicitly modeling the object's shape properties. Many references to various types of employed features are given in [43].

[6]Visual feature learning is not developed in this thesis, since in our practical work it was never planned to implement it, thus we did not study this field.

Chapter 2

Image data interpreting

2.1 Introduction

The perception techniques reviewed so far, directly operate sensor data with few or no knowledge about the environment, therefor they are qualified as low-level techniques.
When high-level knowledge (e.g. information about the "structure" of the environment) is used in addition to this latter, the robot can either localize itself or localize the objects in the surrounding environment.

In section 2.2 we will introduce the different flavours of robot localization problems, and will give an overview of the most common methods used to solve them. Section 2.3 deals with the object location problem, and finally section 2.4 gives an overview of the machine learning issues, and the different existing approaches to solve them.

2.2 Autonomous robot localization

Robot localization has been referred to as *"the most fundamental problem to providing a mobile robot with autonomous capabilities"*. It provides the robot's pose or state (location and orientation) at a moment in time, according to sensor measurements and a priori knowledge of the environment.
Sensors generally provide perceptual data, range data[1] and odometry data (i.e. information about the motion of the robot's actuators). Recently RFID technology has appeared to be a more efficient tool for localization, using RFID-tag readers as sensors [29]. Indeed the robot is equipped with a RFID-tag reader, that can read any RFID-tag placed at a distance of one meter from the robot. And as each RFID-tag has a unique identification number, the robot can uniquely identify locations in the environment in a more efficient and accurate way.

In literature we can usually find three categories of localization problems: the position tracking, the global localization and the kidnapped robot problem [52].
Position tracking is the most frequent problem treated in the literature due to its simplicity. Indeed the initial robot state or position is known, and to localize itself, the robot must compensate its incremental

[1] Range data is provided by a range finder which emits a beam of light that covers a horizontal 180 degree range, for which it measures the distance to the nearest objects.

odometry errors. To compute the actual position, these algorithms often make restrictive assumptions on the error size and the shape of the robot's uncertainty (i.E. they assume that error and uncertainty rates are low).

Particularly challenging is the *Global localization* problem, when the robot's initial position is unknown and must be determined from scratch. In such cases the robot is able to handle multiple, distinct hypotheses on its position.

For autonomous mobile robots operating in dynamic environments where human agents are present, the *Kidnapped robot* problem is often present. This is the kind of problem that the robot faces when a person picks him up, changing thus its position and corrupting by the way the robot's sensor measurements. In this case the task of pose estimation is even harder then the previous ones, since the robot is convinced that he is located at a certain position which is not the current one.

The major issues faced by autonomous robots during the localization process are:

- *odometry inaccuracy* which is due to a difference between the movements that the robot really makes and the those he believes he makes (figure 2.1);
- *sensor inaccuracy and errors* which are due to "mis-understating" of the sensor data, e.g. the robot believes he sees a door instead of a wall;
- and *environmental uncertainty* which is due to the unpredictability characteristic of the most interesting environments (i.e. dynamic ones).

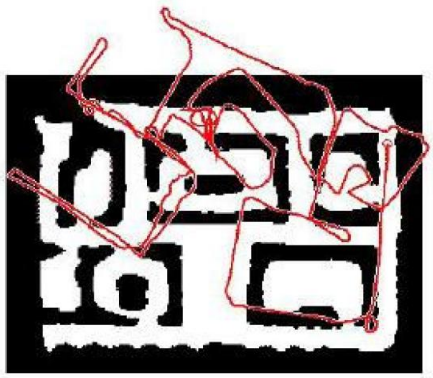


Fig. 2.1: Example of the robot's odometry for a path in a given map. The accumulation of small odometry errors has large effects on the latter position estimations [66]

Most of the existing algorithms, such as Kalman filters introduced in sub-section 2.2.2, only address the position tracking problem. Hence, a generalization of Kalman filters, such as the Markov localization described in section 2.2.3, result in more powerful global localization algorithms. Finally, there is an approach that allows solving the three localization problems, called Monte Carlo localization, and it is presented in section 2.2.4. All of these methods are variants of the Bayes filters, that is why Bayes filters are introduced first in section 2.2.1.

2.2.1 Bayes filters

Bayes filters, allow to probabilistically estimate the pose of the robot from a sequence of sensor observations.
Probabilistic localization means that the robot seeks to estimate its position[2], by density estimation over the space of positions conditioned on available data, generally by using the Bayesian estimation.

Assume that s_t is the robot's pose at time t and $d_{0...t}$ is the data[3] gathered through time, the posterior estimation b_t also called belief at time t, can be written as:

$$b_t(s_t) = \Pr(s_t | d_{0\ ...\ t}) \tag{2.1}$$

As the computing complexity increases exponentially over time with the number of sensor measurements, Bayes based approaches resort to the Markov assumption, i.e. observations at time t only depend on the most recent observations. In other words, if one knows the current state, then future data is independent of past data.
Thus, a system respects the Markov assumption if it respects (2.2), i.e. if the state at time t only depends on the previous $k+1$ observations before observing t.

$$\Pr(s_t | d_{0\ ...\ t}) = \Pr(s_t | d_{t-k\ ...\ t}) \qquad where\ k > 1 \tag{2.2}$$

By applying to (2.1), a sequence of manipulations detailed in [52], we obtain the following recursive equation:

$$b_t(s_t) = \eta_t \Pr(o_t | s_t) \int \Pr(s_t | s_{t-1}) b_{t-1}(s_{t-1}) ds_{t-1} \tag{2.3}$$

Where η is the normalization constant, to insure that this equation is a real probability distribution and its maximum value is 1; and $b_{t-1}(s_{t-1})$ is the belief of being at state s_{t-1} at time $t-1$. The resulting transformation in (2.3) is the recursive update equation of Bayes filters.

In this equation appear two types of probabilities:

- $\Pr(o_t | s_t)$ describes the likelihood of making observation o given that the robot is at state (or location) s, and it is referred to as the *observation model*,

[2] The position or pose generally refers to the robot's coordinates (x, y) and his heading direction.
[3] There are two types of data; observation data gathered by camera images or laser range scans, and action data provided by motor controls or odometer readings.

- $\Pr(s_t|s_{t-1})$ which specifies where the robot might be at time t, given that he previously was at state s_{t-1}. This conditional probability is the *motion model* or *evolution model*, which is a probabilistic generalization of the robot's kinematics (i.e. the change of pose).

The two models are used in order to perform Bayes filter update in two steps:

- the *prediction step* where at each update, the robot predicts its pose according to the update rule of the motion model in (2.4)

$$b^-(s_t) \longleftarrow \int \Pr(s_t|s_{t-1}) b_{t-1} ds_{t-1} \tag{2.4}$$

- *the correction step* where the motion-based predicted belief, is corrected by using observation (2.5)

$$b(s_t) \longleftarrow \eta_t \Pr(o_t|s_t) b^-(s_t) \tag{2.5}$$

Bayes filters are an abstract concept, since they provide only a probabilistic framework for recursive state estimation, but their implementations require specific specification for the perceptual model, the motion model, and the representation of the belief $b(s_t)$ [4]. And the properties of such implementations differ on the representation of the probability densities that is used.

2.2.2 Kalman filters

Simple Kalman Filters

Kalman filters are a variant of Bayes filters, which approximate beliefs by unimodal Gaussian distributions (i.e. a unique top in the density distribution). Gaussian distributions are represented by their mean, which gives the expected location of the person, and their variance, which refers to the uncertainty in the estimation.

Kalman filters are computationally very efficient, but have restricted representational power since only unimodal distributions are considered. In addition, as system and observations noise are considered as very low (close to 0), they work better when uncertainty on the robot's location is not too high.

However Kalman filters can easily be adapted for global localization problems and can consider multimodal Gaussian distributions, as described below.

Kalman filters adapted for global localization

The simple Kalman filter solves the position tracking problem, but can easily be adapted for global localization problems, by "cheating" and using for example markers to estimate the initial pose. Let's see how it works through an example.

Suppose we have a set of markers in the environment, such as the robot has *a prior knowledge* of the structure of the environment and the position of those markers therein.

When the robot is switched on, it has no idea about its initial location. Hence, by recognizing markers in the images provided by its camera and using its *prior knowledge* on the environment, it

[4] $b(s_0)$ is initialized with prior knowledge about the robot's initial pose, or is uniformly distributed if no prior knowledge exists.

makes assumptions about its location. Through time, as the previous location is known, the current location is predicted thanks to odometry readings, and odometry errors are taken into account. This is the *prediction phase* of the algorithm. While the *updating phase* consists on correcting the predicted position by considering image processed data.

Practically, if the robot calculates that on its reference system the marker is at position (x, y), then it will assume that it is located on a circle with center (x, y) and distance d, which is the estimated distance between the marker and the robot (see figure 2.2).

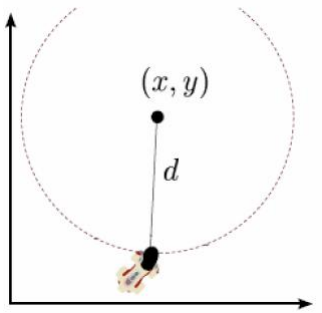


Fig. 2.2: Graphic representation of the probability distribution of the robot's estimated pose resulting from a combination of translation and rotation covariance of a Gaussian [33]

Extended Kalman filters

Multi-hypothesis tracking (MHT) overcome Kalman filters limitations by extending to multi-modal beliefs and still conserving computational efficiency.

MHT represent the belief by mixtures of Gaussians where multiple and distinct hypotheses can be tracked, each one represented by a separate Gaussian. Each Hypothesis is given a *weight* (also known as *importance factor*), determined by how well they predict the sensor measurements. Consequently, the extension of Kalman filters to multi-modal beliefs makes MHT more widely applicable than the simple Kalman filter.

The example in figure 2.3 shows how a robot equipped with a door detecting sensor can localize in a corridor by using multimodal beliefs.
At step (a) the initial position is known to the robot.
After travelling a certain distance without considering observation data, the robot makes a pose prediction based on the odometry data and considers odometry error by solving equation (2.4), resulting on the belief distribution represented in step (b).

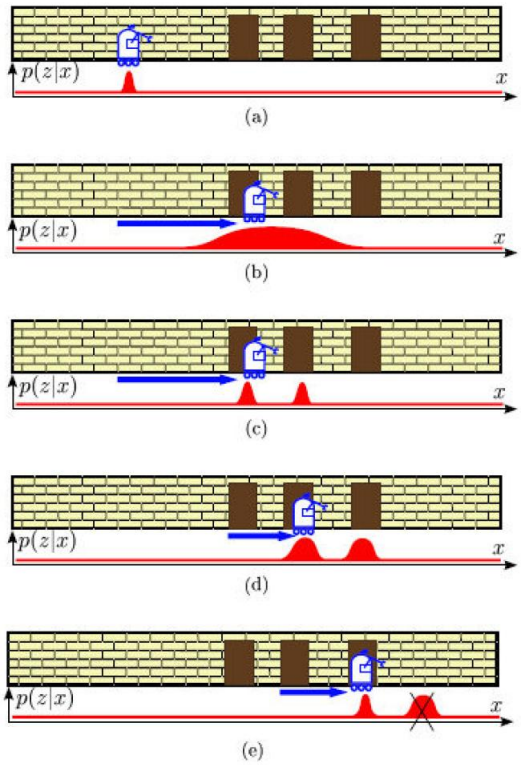


Fig. 2.3: An example of localization using multimodal beliefs and Kalman filters [33]

Once the robot observes a door, he updates the predicted estimation by equation (2.5), and gives same probability to the possibility of being at the first door or at the second door, as described by step (c).
As the robot moves, the distribution "moves" proportionally. When he detects a new door, considering the proximity of doors, odometry noise and the previous hypotheses, he considers two hypotheses: one estimating that he is located at the second door and the other estimating that he is located at door three. After further movements, thanks to its odometry data the robot predicts two possible locations as illustrated at part (d) of the picture. But after detecting a door, the robot updates its location history, and eliminates the incorrect hypothesis in step (e).

Through this example we can clearly understand the difference between unimodal and multi-modal based techniques. If we apply the simple Kalman localization, when the robot detects a door, he can only estimate its location around this door, and he is unable to consider if he is located at door one, door two or three. So the system deals with only one hypothesis i.e. one door. While with MHT, we can see that during prediction and updating phases the robot maintains and checks more than only one hypothesis.

2.2.3 Markov Localization

We will introduce the Markov localization, by using the same example as above described in figure 2.4.

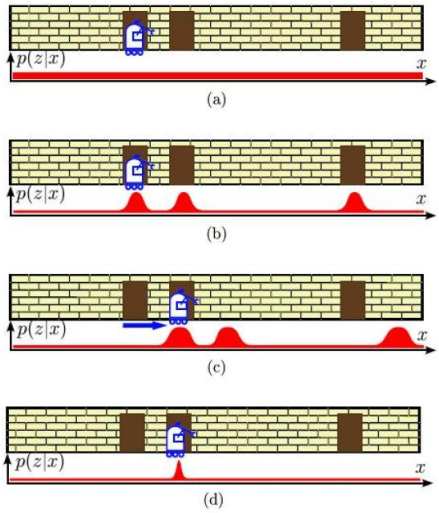


Fig. 2.4: An example of localization using multimodal beliefs and Markov filters [33].

The initial position is ignored and it is assumed a uniform probability distribution of the poses in the environment at step (a).
As soon as a door is detected in step (b), all "door locations" are given higher probabilities than the other possible locations, by computation of equation (2.5).

After further movements, the belief in step (c) is updated uniquely based on motion data according to equation (2.4).
Finally, when a door is detected in step (d), belief is corrected by taking into account sensor observation according (2.5).
In spite of its simplicity, this method is computationally very expensive since it deals with multiple hypotheses, and it only works in discredited environments, i.e. represented by topological or grid maps (see section 2.4.2 over environment mapping). Monte Carlo localization (MCL) overcomes all these pitfalls.

2.2.4 Monte Carlo localization

MCL is the most famous particle filter based localization technique. Particle filters represent beliefs by distribution of "samples", each one having a weight which is its probability within the set of samples. The biggest is the number of samples within a region the highest is the probability that the real state is located therein.
Particle filters realize Bayes operates through a three-step procedure known as *Sampling Importance Resampling (SIR)*:

- *Sampling:* does the update process by sequentially generating samples according to the previous set of samples
- *Importance weighting:* a weight of probability is assigned to each sample
- *Resampling:* consists on replacing "light" weighted samples by "heavy" samples.

Based on the example in figure 2.5, we can explain MCL as follows. In the first sampling phase (a), it is initially supposed that the robot can be located anywhere in the environment. Thus we have a uniform distribution of probability.
When the robot detects a door at phase (b), it enters the importance weighting phase, where all "door location" samples are given a higher weight (and are "heavier") than the others.
During the resampling phase in (c), only these "heavy" samples are selected in ixpense of the others which will not survive.

MCL methods have many advantages, such as their low cost in computational terms, since they focus computational resources in areas of the state space with the highest probability, by sampling in proportion to the posterior likelihood. It is even possible to adapt the size of samples in order to fit to the system computational resources, which is generally very limited for autonomous robots. This allows to decrease the particle filter computational requirements which is exponential in the number of states.

MCL methods are also very easy to implement, however they suffer from two main drawbacks:

- A well localized robot might lose track of its position if MCL fails to generate a sample in the right location. So one has to insure that the sample set is big enough.
- Algorithm results degrade a little when the sensors are too accurate. So it's better to use this technique for very noisy sensors, or by assuming that the error rate is higher then the real one, which is a bit counter-intuitive.

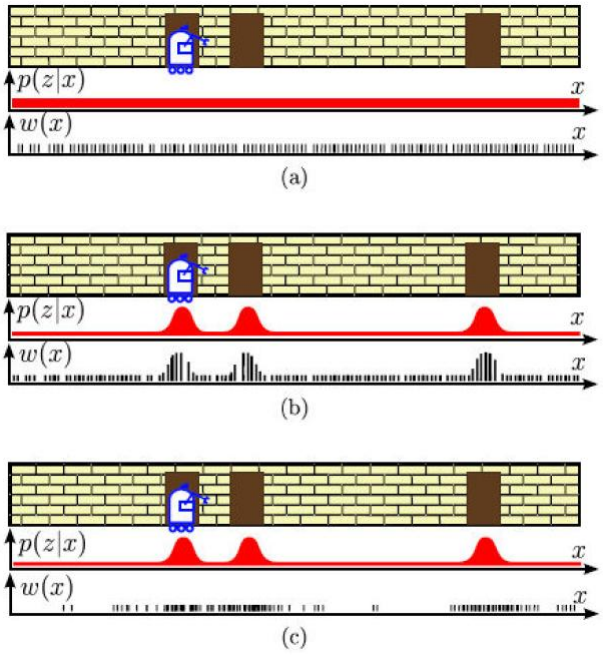


Fig. 2.5: An example of localization using MCL [33].

To conclude this section on self-localization, table 2.1, gives a comparative view of the Bayes filters implementations for position estimation, reviewed so far.

Category	Problem solved	Belief representation	Computational cost
Simple Kalman	Position tracking	unimodal gaussian	polynomial
Extended Kalman	Position tracking	multimodal gaussian	polynomial
Markov	Position tracking	multimodal/unimodal gaussian	optimal
	Global localization	multimodal/unimodal gaussian	optimal
MCL	Position tracking	particle filters	exponential
	Global localization	particle filters	exponential
	Kidnapped problem	particle filters	exponential

Table 2.1: Summary of robotic localization methods

2.3 Object Location

In the case where robots use a camera to acquire information[5] on the environment, once an object has been recognized in the image, an important further step consists of estimating where this object is lying in the real world. This means that we need to use a correct spatial camera model which relates the image coordinates with the world coordinates.

One way of doing such a correspondence is by using the classical *pin-hole model* [18], which assumes that the camera lens corresponds to a single point[6] called *projection point*.
All rays of light enter through this central point **C**, and are printed upside down on the image plane α (figure 2.6), such that to each world point p corresponds a point p' in the image plane.

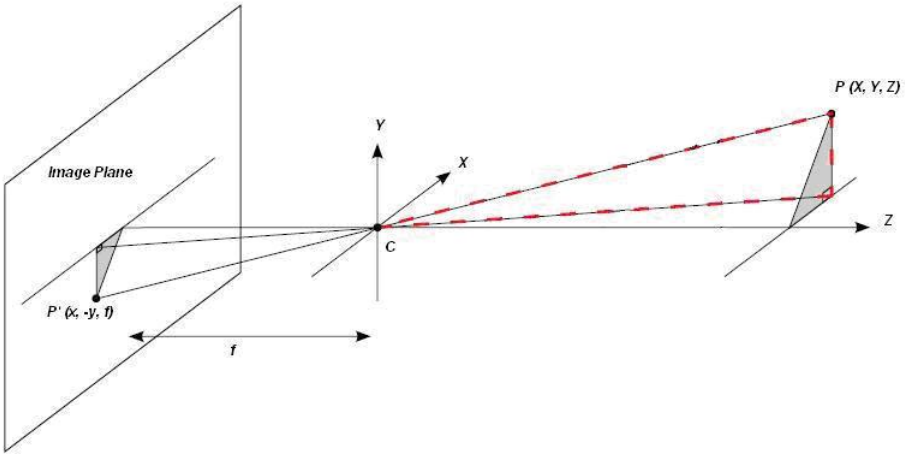


Fig. 2.6: In the pinhole model, each point in the real world is connected to a point in the image plane by a line passing through the central point c which refers to the camera [33]

In figure 2.6, plane β represents the sensor plane, where the coordinates of the camera system of coordinates (x, y, z), are translated into coordinates of the sensor system of coordinates (x, y) where the z coordinate is not considered anymore. And where the distance f between the lens and the image plane is called *focal length*[7].

Assuming that the origin (or optical centre) of the camera's system of coordinates is in the centre of the image, as the triangles Cyf and CYZ are very similar, we can compute:

[5] The case where robots use range finders to locate objects (e.g. walls) in the environment, is not described here, because we didn't make use of it in our practical work, thus didn't study this case.
[6] This assumption (lens = point) represents reality very well when the lens dimensions are very small.
[7] The focal distance is measured in pixels, so that it is easy to convert image distances (in pixels) into the world distances (in centimeters).

$$y = -\frac{fY}{Z} \qquad x = -\frac{fX}{Z} \qquad \textit{for palne } \beta \textit{ coordinates}$$

$$y = \frac{fY}{Z} \qquad x = \frac{fX}{Z} \qquad \textit{for palne } \alpha \textit{ coordinates}$$

The *Direct Linear Transformation (DLT)*, is an improved model, taht takes into account the fact that the image center often does not correspond to the optical center [38]. This model is expressed by the following matrices.

- World coordinates:

$$p = \begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix}$$

- World coordinates:

$$q = \begin{bmatrix} fx_i & fy_i & f \end{bmatrix}$$

- Intrinsic parameters:

$$K = \begin{bmatrix} \sigma_x & \sigma_\theta & u_0 \\ 0 & \sigma_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

- Extrinsic parameters:

$$M = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{21} & r_{22} & r_{23} & t_{21} \\ r_{31} & r_{32} & r_{33} & t_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In (2.6), M contains the extrinsic parameters which represent the rotations (r_{ij}) and the translations (t_{ij}) of the camera system with respect to the real world system, in order to perform a correct switching from the camera coordinates to the world coordinates.
While K is a matrix containing intrinsic parameters which represent the corrections that must be taken into when switching from pinhole model to the real world model.

Finally, the equation we must solve in order to generate world points form image points is given in (2.7) :

$$P_i = K \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot M \cdot P_w \tag{2.7}$$

2.4 Learning

One of the most challenging domains of research in robotics is robot-rescuing. Indeed as robots operate in totally unknown environments (post-disaster), they neither now their initial position, nor have an idea of the environment map. Then, learning methods should be used, which enable the robot with 'state-action' concepts, such as: 'go to the door, then left, then after the second fire extinguisher right', or 'perceptual' concepts, such as: 'corridor', 'hallway', 'window' or 'human'.
This is the most challenging approach in robotics, which aims to provide machines with efficient capabilities of learning through interaction with their environment.

Machine learning incorporates all methods allowing building a model of the real world, either based on a partial or incorrect prior models, or from scratch. This means that when a mobile autonomous robot is switched on in an environment that he doesn't "know", he will have to deal with two kinds of problem:

- Localize itself in the environment according to a partially structured or erroneous map that he must update [66]. In robotics this problem is referred to as the *mapping*.
- Find the objects of interest (e.g. a person or a cup), according to generic features of these objects [43], known as the *visual feature learning* problem[8].

2.4.1 Machine learning

We can find four common machine learning approaches in literature. Methods based on *supervised learning*, generate a function that maps the correct known output (e.g. the correct class of objects) to a given input (e.g. an object detected in a captured image).

While in *unsupervised learning*, the machine[9] simply receives inputs, but outputs are to be constructed and no feedback from its environment is received. The goal of the system in this case, is to produce from the inputs a set of patterns (thus the outputs) that will be used for decision making by predicting future inputs.

In methods based on *reinforcement learning*, the machine interacts with its environment by taking actions and receives rewards (positive feedback) or punishments (negative feedback) to this actions, provided from the environment. Reinforcement learning attempts to find a way of maximizing future rewards over the machine's lifetime.

Finally, there is *inductive logic programming (ILP)*, where from a database of facts and expected results, which are divided into positive and negative examples, the ILP based system tries to derive a logic program that proves all the positive and none of the negative examples. ILP consist on learning from examples and background knowledge, and often uses the Prolog specifications.

[8] Visual feature learning is not developed in this thesis, since it was not needed. Indeed, in our practical work, there are only a few landmarks and a newspaper to recognize, so that a complete description of them can be provided to the robot.

[9] Here the term machine means a robot or a computer, since machine learning is not only used in robotics, but also in different imagery fields such as medical imagery, landscape imagery etc.

2.4.2 Environment mapping

In section 2.2, we talked about localization techniques where a map of the environment is given as input to the robot. But if we want the robot to efficiently carry out missions in human interaction environments, where the surrounding is unknown (e.g. rescue robots in devastated places), the robot should also be able to acquire and maintain models of the environment by itself. Thus, the localization problem becomes more challenging, since the robot must localize itself by using a map that it will have to build incrementally as it moves through the environment.

Such a problem is referred to as the *Simultaneous Localization And Mapping (SLAM)* or the *Concurrent Mapping and Localizing (CML)* problem. A large amount of research has been concentrated on the SLAM problem and SLAM methods are nowadays very powerful, however they are very little employed outside the research field [66].

As the name indicates, in this case the robot deals with two issues (i.e. mapping and localization), so that the above described methods on localization (or more efficient variants or mixtures) are used at higher computational costs.
Indeed, in the SLAM problem, the robot will have to locate itself according to a partially structured or erroneous map, so that new assumptions must be considered (E.G. the correctness rate of the map, estimation of the ignored map dimension ...), and the number of hypothesis to deal with is much bigger. It results that SLAM is significantly more difficult than each task (mapping or localization) in isolation[10]. As we already have an idea of half of the job (localization), we will now say some words about mapping techniques.

The robotic mapping field is widely divided into two approaches; *metric* and *topological approaches*. The former plays with the geometric properties of the environment, while the latter considers the connectivity between different places.

One of the most successful applications of the metric approach is the grid-based mapping method proposed by Elfes and Moravec (referenced in [66]). This method provides metric maps which represent geometry characteristics of the environment (e.g. county areas, vegetation, soil types or lakes). They tessellate the environment into small and even grids, where each grid cell may indicate the presence of an obstacle or of the robot in this cell. Since each grid resolution must be fine enough to capture any important detail, which requires enormous time and space complexity, the grid-based representation is efficient in environments of moderate size and complexity. Environment modelling through grid construction is based on sensor data interpretation and odometry, through artificial neural network where Bayes's rule is used to estimate probabilities of grid occupancy.

Topological approaches[11] as their name indicates, represent robot environments by graphs where only relations between points are described, without any geometry notion like scale or distance (e.g. a metro line map). Nodes in the graph correspond to distinct situations, places or markers, and arcs represent the existing direct path between them. The position of the robot relative to the model is determined

[10] We could easily "lighten" the complexity of SLAM methods, by using external positioning devices, or by using sophisticated engineered landmarks placed in the environment, but that is scientifically less challenging.

[11] Topological map algorithms are not used for SLAM since they don't create geometrically accurate maps of the environment however they are part of the most common mapping techniques and are worth to be mentioned.

based on current sensor readings but also on landmarks or markers recognition.

Compared to grid-based methods, topological models have several advantages. They are faster planning, problem solvers and provide more intuitive interfaces for human instructions (e.g. "go at the living room").
Furthermore, as their dependence on odometry readings is not important, they are not victims of bad odometry estimations due to slipping or drifting phenomena, which can be fatal in grid-based methods as they accumulate through time.
On the other hand, topological methods have more difficulties in differentiating places that look alike, and construction of maps is even more difficult for large-scale environments because of sensor noise and the dependence of sensor input to the robot view-point.
One common point between the two models resides in their large requirements in memory.

SLAM techniques based on non visual sensors (e.g. sonar sensors that return the proximity of surrounding obstacles, or range-finders) and odometry contributions, work well on simple and structured environments, which is not the case of natural environments such as houses. That's why research also investigates on the utility of principally vision-based mapping systems. Then we enter the notion of *Visual Simultaneous Localization And Mapping (VSLAM)*. A good overview on robotic mapping approaches is given in [66].

Chapter 3

Decision Making

Since Artificial Intelligence aims to imitate human beings, making a robot behave is part of the challenge. Let's have a look to a common definition of a behaviour and see how it is adapted to robots. In the encyclopaedia, a behaviour is explained as follows:

"The term generally refers to the actions or reactions of an object or organism, usually in relation to the environment or surrounding world of stimuli" ... "the sum of all motions, gestures and signals, both visible and invisible, created by the organism."

Such a definition of the behaviour can also be applied to the robots, which according to their environment and to their goals, can walk, take or release objects, communicate etc.

We can distinguish three types of such "biological" behaviours; reflexive, reactive and conscious behaviours in robotic agents [7].

Reflexive behaviours are said to be conscious-less since they are immediately generated on the basis of sensor input, like for example the knee-jerk reflex which involves a sudden kicking movement of your lower leg after the tendon just below your kneecap has been tapped.

Reactive behaviours are learned, just as we learn biking. However they don't require conscious thinking but still can't be considered as reflexes.

The third category contents the *conscious behaviours* which are performed after a reflection process, such as reciting a poem.

Programming robotic behaviours essentially consists on defining them as a sequence of actions, and then deciding under which conditions a given sequence of actions must or can be used. Behaviour programming architectures presented in figure 3.1, differ on the position and the "conscious degree" they give to the decision making step [1].

The Deliberative paradigm (figure 3.1(a)) [1] was the first to occur in the late sixties, and is characterized by the fact that the robot maintains a global representation of the world called model, and a planner which decides which actions to take based on this world model.
All sensors' data are here analyzed by an appropriate module (Sense), in order to extract information on the environment, and update the current model.

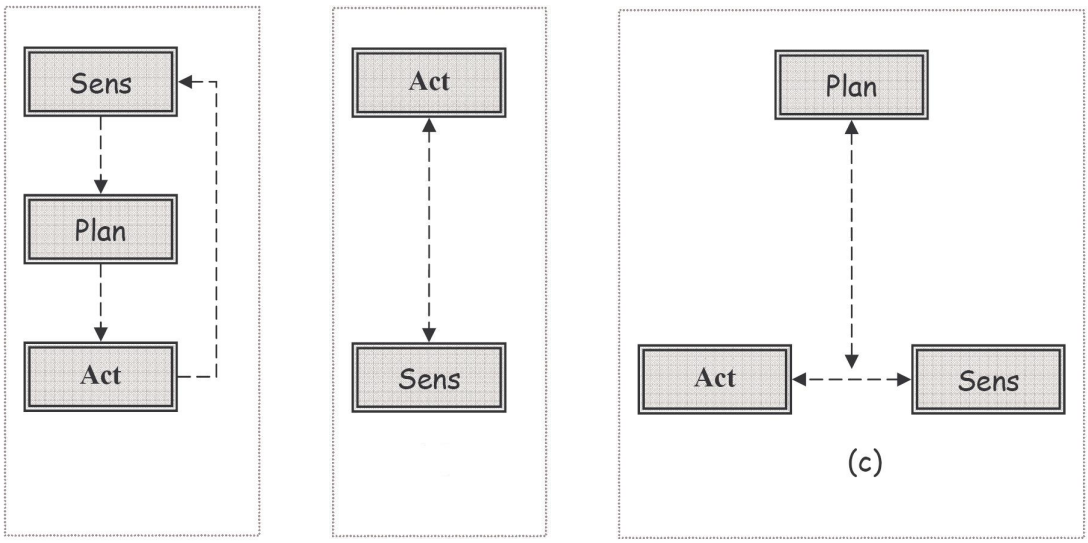


Fig. 3.1: (a) Deliberative paradigm; (b) Reactive paradigm; (c) Hybrid deliberative/reactive paradigm

Then the deliberative module (Plan) will generate the most suitable behaviour to perform based on the world model. At the end, a specific low-level module (Act), will translate the chosen behaviour into a sequence of mechanical actions that must be executed by the motors. This reasoning is similar to human thinking; e.g. "I see the coffee machine, I plan how to get to it, I walk".
However, on-line planning (i.e. generate new plans during run time) might often ask too much computational resources than the robot benefits from, or might be too time consuming and non-efficient for real-time applications.
Thus, generally the robot is equipped with a set of off-line pre-calculated plans. In this case the job of the deliberative module will consist on choosing the "best" plan among a set of plans previously given. However, planning is not the only expensive task in this paradigm, maintaining and updating a model of the world also requires important computational resources, and the result is very sensitive to environment changes.

To overcome the poor efficiency of the deliberative architecture, the *reactive approach* (fig.3.1(b)) [1] was proposed, where the robot does not maintain or update a global representation of the environment, and no planning is performed.
Hence, a computationally poor robot can do the job, since he doesn't neither memorize its previous actions nor predict actions for a longer future (i.e. further than the next step), it provides very fast responses and is thus more reactive.
Indeed, the robot is equipped with a set of behaviours (pairs of stimulus-action(s)), and a "coordinator" which chooses if and/or which behaviour should be triggered by specific stimuli (corresponding to sensors' information). Thus the Sense module corresponds to stimuli, and to the Act model is associated one or

more actions.
This means that in opposition to the deliberative architecture, where the output of one module is the input of the next module, the modules of a reactive architecture are totally independent. But the lack of a global model is also an inconvenient, since the robot only relies on sensors which can be noisy and then the data they provide is not always reliable.

Since reactive architectures are only suitable for environments that are unpredictable or cannot be easily modelled, they can be improved by taking into account some knowledge of the world. It's the principle of *hybrid deliberative/reactive paradigm* (fig.3.1(c)) [1]; composed by a planning level and a reactive level.
The planning module decomposes a task in a set of subtask to be performed based on the global knowledge of the world, and then each subtask is performed according to the reactive model. Hybrid deliberative/reactive are the most successful for the moment, although their implementations are specific to the applications.

In front of these choices, the question that appears is: which is the best architecture to implement? It depends on the working environment; for some researchers having an internal representation of the surrounding world is useless, while others consider it as important.

Generally reactive paradigms are interesting in environments that are static (i.e. the environment doesn't change meanwhile the robot deliberation) and deterministic (i.e. the robot is sure that it can perform its action without encountering any problem).
Briefly, environments that the designer knows and can predict all possible behaviours within it. But this is rarely the case since robots are more often used in a dynamic and non deterministic world, and in that case a deliberative model is also needed.
The choice is also influenced by the short or long-term property of the behaviour. Some behaviours are low-level and consist on reacting instantly on changes in the environment and are therefore very short-term and reactive. While more high-level behaviours, pursuing a global and high-level goal, try to prevent frequent state changes and make more deliberative and long-term decisions. Of course a robot might need to perform both kinds of behaviours, and then the hybrid architecture and planning becomes interesting [65].

These architectures can be implemented through different implementation approaches; by using state machines approach as did the German Team and Chaos team for Robocup 2005 [65], [7], hybrid automata or hierarchical structure of decision trees [50], or symbolic planning as did SPQR team in 2005 [17].

Chapter 4

An overview of AIBO ERS-7

The first generation of Sony's AIBO (ERS-110) was launched in 1999, and the AIBO model we worked with is the third generation ERS-7 model. The latest model on the market today is AIBO "Champagne"[1]. In the following section we will have a look at AIBO's ERS-7 model[2] hardware and peripherals [58].

4.1 Hardware and Peripherals

AIBO has a 64 bits RISC processor operating at 576 MHz. He uses two types of memories; an internal memory of 64MB storage capacity, and a removable memory stick of 16MB through which the robot is programmed. He is also equipped with several sensors and effectors that he uses to interact with the environment. In this model we can identify three of the human senses; Touch, Sight and Hearing.

Touch. AIBO has two types of touch sensors, electrostatic, and pressure. The electrostatic sensors are positioned on the dogs head and back, while the pressure sensors are positioned under the paws and chin. He also has acceleration sensors, a temperature sensor and a vibration sensor.

Sight. AIBO has a low cost CMOS image sensor used for capturing images of the environment at a rate of 30 frames per second.
The camera can work in three resolutions 208x160, 104x80 and 52x40, and he has a 56.9 degrees horizontal angle of view, and 45.2 vertically degrees angle.
Beside the camera, AIBO is also equipped with long and short range sensors, which can operate at distances from 10 cm to about 90 cm, which can provide information on the space. They are placed in the nose and on the chest.

Hearing. AIBO can record sounds with its two microphones located at each side of the head, and the sound can be recorded in stereo.

We can also classify AIBO's effectors in two groups; movement and communication effectors. *Movement effectors.* AIBO has 20 Degrees of Freedom (DOF). He has three joints in every leg; elevate, rotate

[1] AIBO "Champagne" might also be the last AIBO's generation, since Sony stopped producing AIBOs in March 2006.
[2] Detailed information on the model can be found at [60].

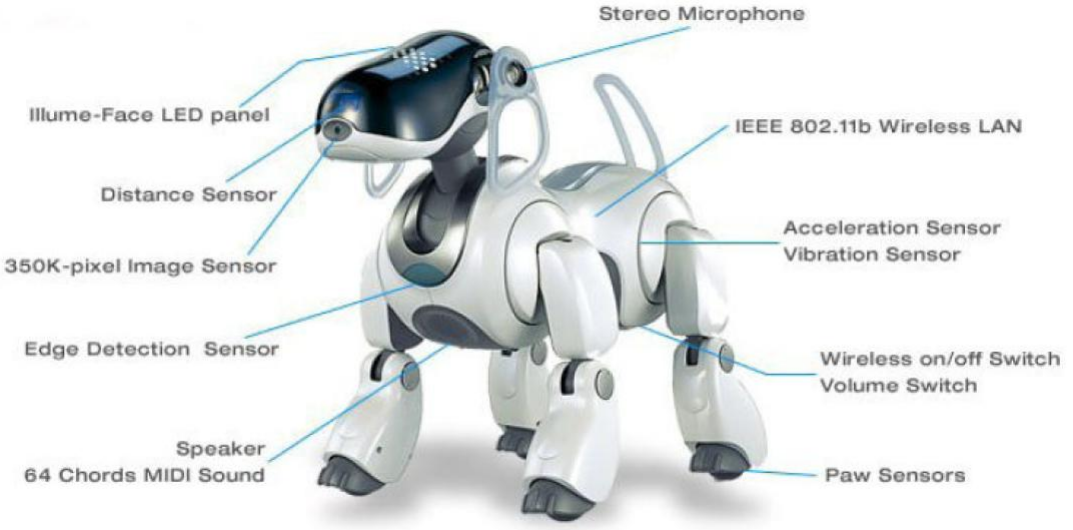


Fig. 4.1: Front view of AIBO's ERS-7 hardware and peripherals [58], cited in [50]

Fig. 4.2: Rear view of AIBO's ERS-7 hardware and peripherals [58], cited in [50]

and knee (3x4 DOF), three joints in the head's neck; tilt, pan and nod, two joints in the tail; tilt and pan, one joint for the mouth and two movements for the ears; flick up and flick down (2x2 DOF).

Communication. There are three communication means for AIBO; visual, audible and wireless. The visual communication is done by LEDs, situated on the dog's head and back. The audible communication is done by a mini speaker located on the chest. The wireless communication is done by an IEEE 802.11b wireless Ethernet interface.

4.2 OPEN-R Software Development Kit

OPEN-R[3] is a modularized and object-oriented software which serves as an abstraction layer to the Aperios operating system. It has a layered architecture that consists on a system layer and an application layer.

The application layer is the layer where a program is written by the user. It uses the system layer interface to access the robots hardware. The system layer contains the necessary services to access the robot's hardware.

The system layer contains modules which are called objects. These objects are different from standard C++ objects, but there are similarities.
There are three important objects provided by the system layer which provide different services to the application layer:

- *OVirtualRobotComm* is the object that handles the sensors, effectors (joints and LEDs) and camera.
- The *OVirtualRobotAudioComm* object handles the audio communication (speakers and microphones).
- The object that handles TCP/IP communication is ANT.

The modules communicate through messages. These messages are passed through inter-object communication, which uses predefined communication channels. The messages can contain any C++ type data and an identifier specifying which method should be executed when the message arrives. Every OPEN-R program contains several concurrently running modules, but each module can only process one message at a time. This is because every module in OPEN-R is single-threaded, thus every module must have its own message queue.

[3] All information on OPEN-R can be obtained on its website [59]

Chapter 5

Image Processing

From this chapter on, we will describe our implementation which is organized according the same architecture as SPQR. This architecture is depicted in figure 5.1. The squares in this image, represent the different modules of the architecture and the arrows represent their flows.

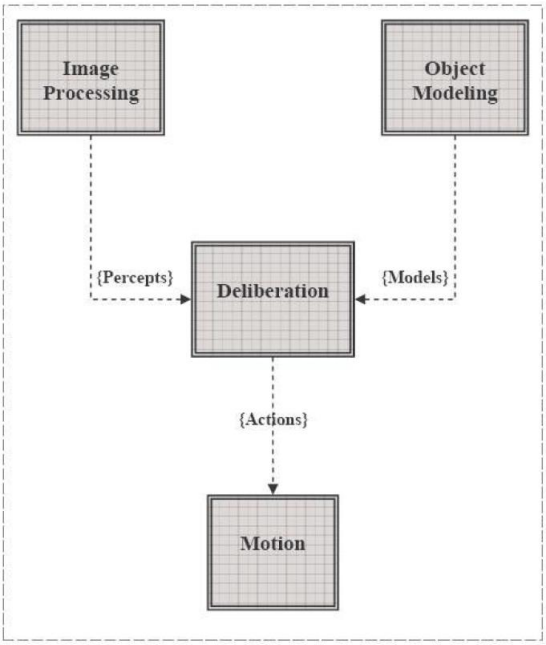


Fig. 5.1: Our implementation of the robot's working steps

In the following chapters we will gradually present these modules, starting with the Image Processing[1] described in this section.

We remind that our goal for this work consisted in programming AIBO for acting like a companion pet in an indoor environment. More specifically, AIBO should be able to fetch a newspaper. As the robot is just a set of chips, wires and motors working together, educating a robot is considerably more complicated then educating a real healthy dog. Indeed as a real dog can see, walk, and think, our task consists on making a set of electronic devices see, walk and think.

Because of the little time for achieving this large and complicated task, most of the techniques we used in this work were inspired by the work of SPQR and GermanTeam [65], [61].

In our work, the camera is the only sensor of the ERS7 robot that is used to gather information on the external world, thus a suitable image processing is necessary to achieve percepts and behaviors as accurate as possible. That's why during our practical work, we mainly focused on the perception process, represented by the ImageProcessing module in the implementation architecture. This module is given as input an image and the position of the robot's camera relative to its body (in order to know where the robot is looking when the image was captured). The output consists on a set of objects of interest (objects known to exist in the environment) that were recognized in the image. Then the position of each object in the robot's system of coordinates is computed.

We distinguish two steps in Image processing. As AIBO evolves in a colored environment, as first step, the image is scanned once, in order to associate a color class to the pixels in the image. This is the job of the color segmentation (or color classification) and is presented in the first section of this chapter. After the segmentation process, the image is scanned a second time, to detect objects in it. This step is explained in the second section of this chapter. In sections three and four we will respectively present the tools we used, and the different issues for the Image processing process.

5.1 Color segmentation

Working with AIBO imposes several constraints concerning the choice of the segmentation algorithm to implement. Indeed, AIBO suffers from limited computational power (576 MHz CPU), its camera provides low quality color images, and he operates in a colored environment where real-time reaction is important.

Therefore, many segmentation techniques presented in literature cannot be applied. Furthermore, our goal is to use AIBO in an environment with varying lighting and where the user should not have to perform any kind of technical task to make the robot "see" properly. This means that our color segmentation must be "adaptive" to changing lighting conditions, and the camera calibration (or creation

[1] We didn't work on the motion level; we only used some of the motions already implemented by SPQR. For more information on Motion Control, you can refer to the Motion Control of the German RoboCup soccer team (GT) [65], since SPQR uses a big number of their motions. In this thesis the different motion requests that we used will be explained in the Deliberation part.

of color/lookup tables) must be automatic and not manual.

Thus, we need to use a color segmentation approach with the advantages of an off-line automatic calibration and of on-line color segmentation (see section 1.3 on Image segmentation). However, in the methods proposed by the RoboCup community, there is not yet an ideal mix of the two approaches.

For example in [62], machine learning is used to build color tables automatically. However, the same table is used during the whole robot's mission, being thus sensitive to lighting changes. At the other hand, in [26] is presented an implementation of on-line color segmentation on AIBO, with automatic calibration. Hence manual calibration is still preferred.

We decided to use the color-segmentation algorithm proposed by Luca Iocchi (member of the SPQR team) in [20] and also described in [34], since it is an on-line color segmentation where calibration is almost completely automatic, it is efficient (the method has been experienced during some of the games in RoboCup 2005), cheap in computational terms and adapted to our environment.

Indeed, the method that he has implemented, requires one initial manual calibration and the setting of a few parameters before the robot begins its task. Afterward, the color tables are automatically updated, in order to adapt to condition changes. However, one future work could focus on the realization of a complete "self calibration" method, eventually by using machine learning techniques. In the following subsection we will briefly explain how the method works.

5.1.1 Color segmentation through transformation of the color distribution

The color space (i.e. the color representation) of the images captured by the color camera of AIBO, is YUV.

However, as we told in section 1.1.2 on the color spaces,there is an interdependence between the color and the luminosity components of YUV. While we want the color to be recognized independently from its luminosity. For this purpose, HSV (Hue, Saturation, Value) color space is most suitable.

Consequently, a conversion from the YUV space to HSV must be performed. In order to have an algorithm which is efficient for real-time situations, this transformation is not computed each time an image is processed, but a table associating to each YUV value its HSV correspondent is provided off-line [34]. Then the color space is partitioned in distinct color classes that are known in advance, and will be associated to the pixels of the image (by the segmentation process)[2].

There are two manners of defining the set of color classes, either it is done *statically* (i.e. classes are defined before the robot starts its work), or it is done *dynamically* (i.e. the color class partition is performed during the robot task execution).

Once again, in order to have a system which is not sensitive to changing lighting conditions, in the implemented method, dynamic segmentation is performed (see section 1.3 on image segmentation). Then

[2] As this method exploits knowledge on the color classes that are known in the RoboCup soccer environment [20], it also fits for our environment since the color classes we employ are a sub-set of those used for the soccer game. That's why we said that this method is adapted to our environment.

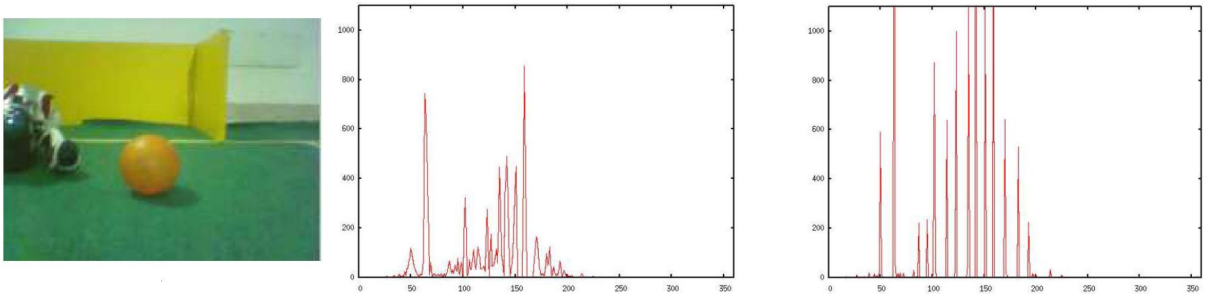
a histogram of the image is considered only for the H (hue) component of the color space HSV, since H alone can discriminate among the colors red, yellow, green, blue and pink [20].

However, not all pixels are considered for the histogram construction. Indeed, as you can see in figure 1.3 (in section 1.1.2) which gives a representation of the HSV color space, pixels with low saturation (S component) represent white colors, and those with low luminosity (V component) represent black colors, thus their Hue value is hard to distinguish. That's why, pixels used for the histogram creation, must have saturation and luminosity values within a respective threshold [34].

As usual for the segmentation process, once the histogram is created, we search for peaks in it, and to each of this "peak points", is associated a color class thanks to the lookup table. Then all points situated between two minimal points, will be attributed as color class the one that corresponding to the maximum point between them [34].

Finding peaks, can be done either by using static thresholds, either by using dynamic ones (see section 1.3.1 on Threshold-based segmentation). But as dynamic thresholds are computationally expensive and we need a cheap method, static thresholds are used. But we also need a method which is robust to light variations, which means that we would like the histogram to not change too much in presence of light changes, especially close to the thresholds! To do so, a transformation function is applied to the color distribution (or histogram) of an image, so that the fixed thresholds can still be used in the new color distribution [20].

Figure 5.2 shows an example of such a transformation, where the values of the H component are presented in the X-axis of the diagram, in degrees. In figure 5.2(b) the hue value around 50 corresponds to orange colors, that around 60 to yellow, and that higher then 100 to green. After the transformation, presented in figure 5.2(c) the same interpretation is still correct. This transformation provides a new histogram where there is large distance between the peaks, which means that in case of small illumination changes (where the H component can vary by an order of 10 degrees), the fixed thresholds still give good results.



(a) (b) (c)

Fig. 5.2: (a) an image captured by the color camera of AIBO; (b) typical color distribution of the H component for this image; (c) the color distribution of H, obtained after application of the transformation function [20]

5.1.2 Summary

As we saw, some parameters have to be set manually and only once, depending on the initial lighting conditions before the robot starts working. These parameters are :

- the four (two minimas and two maximas) threshold values of saturation and illumination that are necessary for the selection of candidate pixels for the color distribution creation,
- the different thresholds on H to separate the different colors,
- and also lower limits for S and V.

However, some work has to be done, in order to have these parameters automatically set by the system. The method is also computationally cheap for two reasons;

- the color distribution is not computed for all pixels (i.e. we don't check the color of each pixel),
- the automatic color distribution transformation is performed only once every 25 frames (for setting the thresholds), and it will be statically applied to segment the subsequent images.

At the end of the segmentation step, we have an image, represented by a set of pixels, each one having a corresponding color class.

5.2 Feature detection

The image resolution we used is 208×160 pixels, but not all pixels are analyzed for object extraction. In order to decrease the computational charge and find an object rapidly but still doing the best to avoid false positives, we chose to scan only the pixels on the vertical and horizontal lines, that we will call scan-lines. The construction of scan-lines is based on the horizon of the image, which is explained in the following sub-section.

5.2.1 Horizon calculation

The definition and implementation of horizon-line that we used, is exactly the same as the German Team. What we call horizon-line[3] is the intersection of the projection plane P (where a scene of the real world is projected by the robot's lens) with a horizontal plane H parallel to the ground at the height of the robot's camera C [65].

To calculate the horizon-line, we will consider its endpoints (h_l and h_r), defined as its intersection points with the projected image, in the system of coordinates of the camera represented by the three red axes in figure 5.3.

For each image the position of the camera relative to the robot's body may be different, according to the movements of the robot's head and neck. Those positions are stored in a matrix named Rotation Matrix. As we only know the vector of coordinates of the endpoints in the system of coordinates of the

[3] the position of the horizon-line in the image only depends on the orientation of the camera.

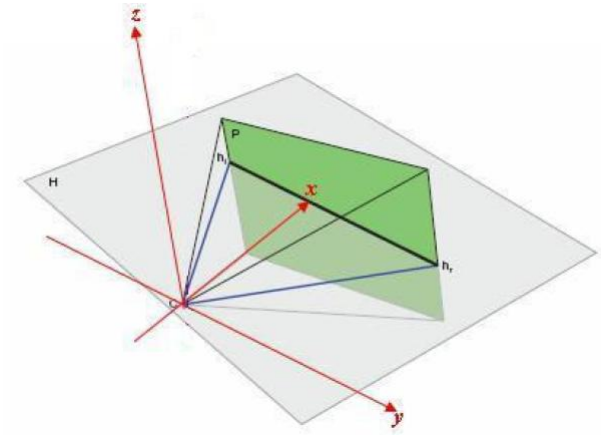


Fig. 5.3: The horizon-line is obtained by intersection of the projection plane P and the horizontal plane H [65]

robot, the rotation matrix will allow converting this vector to the system of coordinates of the camera [65].

5.2.2 Scan-lines construction

Scan-lines will be constructed as parallel and perpendicular to the horizon[4] (figure 5.4).

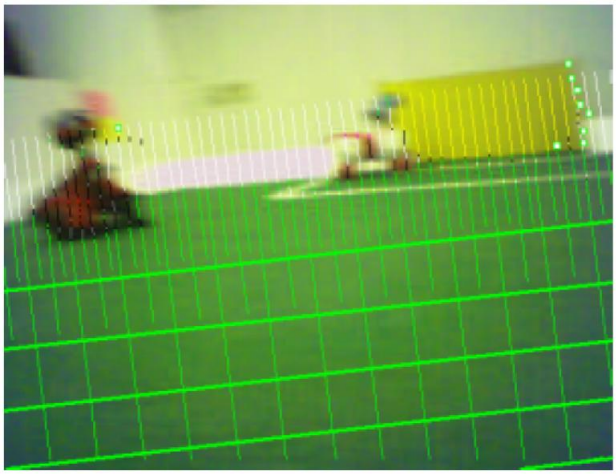


Fig. 5.4: Construction of grid-lines according to the horizon [65]

Each scan-line is scanned pixel by pixel, searching for colors or series of colors that might indicate an

[4]This image is just an example, in our implementation the distance between all horizontal scan-lines, and between all vertical ones is fixed (5 pixels).

object of interest. Since all the objects of interest are lying on the floor, only the bottom image horizontal scan-lines are scanned, while the vertical scan-lines are all analyzed. During the scanning the following parameters are taken into account:

- a pink, blue or yellow pixel which immediately follows or precedes a green pixel, might indicate respectively the beginning of a colored-white line or the end of a white-colored line,
- a white pixel immediately following a green one, could be either the beginning of a white-colored line or the beginning of a newspaper,
- a white pixel immediately preceding a green one, could be either the end of a colored-white line or the end of a newspaper.

As a consequence we always memorize the characteristics of the last point of interest seen (color and coordinates) as well as the total number of colored pixels of interest. Remembering the total number of interesting points allows us to eliminate false positives. Indeed the total number of line points or newspaper points met in one scan-line must be more then 10 and more then 4 in a vertical scan-line. If it's not the case, this can mean two things:

- the association of the white color to this pixel is an error,
- or there is a bit of a line or newspaper in the image.

However in both cases we consider that no object of interest has been detected. The algorithms charged of detecting lines and newspaper points are implemented in the RoboCareImageProcessor class. Let's see what happens in this class:

- first an image correction is performed[5], i.e. once the image was segmented and a class of color was associated to each pixel, the image-correction algorithm tries to eliminate errors by associating to a little group of badly colored pixels, the color of their neighbors,
- then all perceptions that were performed for the previous image are reset i.e. all point or object of interest detected in the previous image is deleted[6],
- vertical and horizontal scan-lines are generated,
- horizontal scan-lines are analyzed first, then vertical scan-lines, searching and storing in appropriate structures all points of interest,
- finally the line-finding algorithm will check if any line or the newspaper is present in the image. In the case lines where detected, we project the lines in the robot's space and store them in a list of projected lines.
  As we need to save the previous lines seen (see the section on Object Modeling), all these lines' position will be updated according to the odometry variation between two frames. In case no line was detected, the newspaper-finding algorithm is launched. As for the lines, a detected newspaper is projected in the robot's space.

[5] After the correction process, some errors might still be present.

[6] Information obtained by the image (presence of an object of interest, its position, the position of the robot and the time when he saw it) is still present in the Model of the object.

5.2.3 Landmarks detection

The image processing that we perform, aims to provide information for finding position of the robot in the environment, in order to follow the right path to go to fetch the newspaper. In section 2.2, we described the different existing localization approaches. In our work, we used a landmark based localization instead of a map based one. The reason to this choice, is that the robot is supposed to operate in any environment. But we cannot define a map for each user's living place! Thus we believe that it is more logical to use landmarks that provide to the robot information on its position.

The landmarks that we chose to use are bi-colored lines made of paper and sticked on a green carpet (represented in figure 5.5).

We chose to use a carpet because the robot slips quite easy on the floor. The choice of the carpet color is not critical[7]. What is most important, is to have a carpet color, that can be easily distinguished from the lines. Equally, it is important for the lines to have different colors, that cannot lead to confusion, i.e. avoid colors such as yellow and orange, light-blue and dark-blue ... That is why we chose the colors yellow, blue and pink.

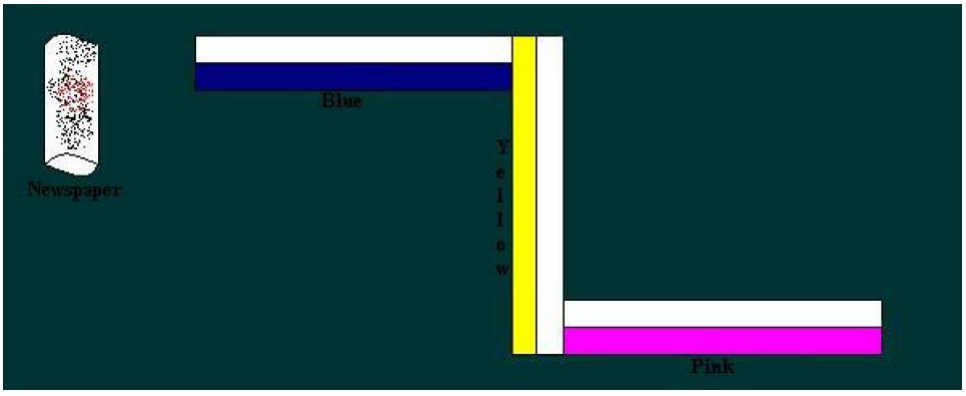


Fig. 5.5: The field of work is composed by three different bi-coloured lies and a newspaper at the end of the blue-white line

Of course as those lines serve as landmarks, they must have a unique different color. The reason why we have each line made of a white and a colored line, is for extracting information on the direction. Indeed, each color represents a room in the environment, and their two colors help the robot to find its direction. If the goal of the robot is to find the newspaper, then he must follow the sequence of color-white lines, and once he has found the newspaper and he wants to go back, he must follow the white-color lines.

This means that the robot knows which path he must follow in order to get to the news paper, and which one he must follow in order to come back. Therefore, anytime the robot sees a line, he knows exactly where he is, and where he must go. However, he is shore of its location, only if he is shore of the line he has seen. Indeed, he may have recognized a white-pink line instead of a pink-white line. Thus,

[7]We used a green carpet because we worked in the laboratory of the SPQR team, which uses green carpets as field for the RoboCup soccer competitions

we also must manage uncertainty (see section 7.3.1 on line following), but we don't deal with it by using probabilities, as the localization techniques described in 2.2. However, even if can deal with uncertainty, recovering from false positives is very hard, consequently, a good landmark recognition is very important.

Our current implementation of line recognition is based on edge detection. An edge is a sequence of pixels of a scan line, where there is a significant variation of the intensity or at least one of the components of the color space.

Various edge detection techniques are proposed in literature such as Roberts, Sobel or Canny edge detecting operators (see section 1.3.2 on edge-based segmentation), but they are computationally expensive and given the limited computational capacities of AIBO, an easier technique less demanding was applied.
Indeed, our edge points for lines are detected only by color comparison. For example, if a pink pixel is the right neighbor of a green pixel in a horizontal scan-line that might indicate the beginning of a pink-white line. Then, we count the number of pink lines that we encounter, until we find one of the right neighbors with a different color.

We remind that to avoid false positives, a minimum and a maximum number of potential line points are fixed, in case some errors subsist after the image classification (or segmentation) and correction. As the output of this edge detection part, defines only where edges are in the image, we need to forward the set of edge points to a feature finding method which will determine both what the features are and how many of them exist in the image.

Many edge based object recognition algorithms exist and some are presented in [67], but we chose to use the Hough Transform, since it is not affected by image noise and is tolerant to gaps in the feature description. This quality is important in our case for two reasons:

- the material of the colored lines is such that light might reflect on it,
- the characteristics of the color camera some lighter stripes appear in the image (see figure 5.13). However we need to recognize a whole colored line, even if some white or brighter pieces appear in it.

In the two following subsections, we will introduce the line-finding technique based on the Hough transform, and we will explain how we implemented it.

Introduction to the Hough Transform

The Hough transform is used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the conventional Hough transform [19] is most commonly used for the detection of regular curves (which describe feature boundaries) such as lines, circles, ellipses, etc.

There is also a *generalized Hough transform (GHT)* [3], employed in applications where a simple analytic description of features is not possible. But as GHT needs large amounts of memory and long computation time to recognize an object, we decided to use the conventional Hough transform.

The Hough transform can identify the parameters of an equation which best fits a set of given edge points. Suppose we have a vector of n variables $X \in R_n$, and a vector of m parameters $P \in R_m$, as arguments to a generic function f. We want to describe a set of points satisfying:

$$f(X, P) = 0$$

Given a set of different edge points $S = X_i : i = 1, ..., k$ from an image, where k is the number of points, by solving the following system:

$$\begin{aligned} f(X_1, P) &= 0 \\ f(X_2, P) &= 0 \\ &... \\ f(X_k, P) &= 0 \end{aligned}$$

We can find a vector of parameters $\overline{P}$, belonging to the function of a curve that passes through all (usually the case if $k \leq m$) or most (if k>m the system is over constrained) of the edge points. Then can write:

$$h^{X_i} = \begin{cases} 1 & \text{if } f(X_i, \overline{P}) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Which is a function that tells if a point X_i , belongs or not to the line having the parameters of the vector $\overline{P}$. Then we can write the Hough Transform as follows:

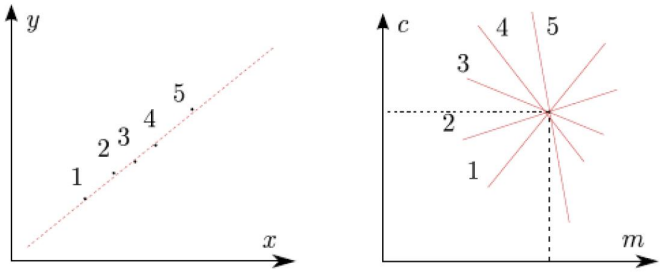
$$HT_S(p) \ = \ \sum_{i=1}^{i=k} h^{x_i}(p)$$

As we can see, Hough function is a "voting function", which counts how many edge points "fit" for the function represented by the parameters of the vector p.

If we assume that we know the function we're searching for, then we just have to calculate the maximum of the Hough function by testing all edge points to all vectors of the parameters' space, in order to find the "best" function. A straight line in a bi-dimensional Cartesian space can be expressed as:

$$y = mx + c$$

With this formula, to each point in the Cartesian space (x, y) can correspond a set of lines in the Hough space (c, m) as shown in figure 5.6, such that m may vary from minus infinity to infinity (e.g. in case of lines parallel to the y-axis). And as we need space to store this set of lines, and the machine can't represent ∞, it is better to use a discrete space of parameters. To overcome these pitfalls, polar coordinates are used (figure 5.7) [33].



(a) (b)

Fig. 5.6: (a) points in the Cartesian space, (b) lines in the Hough domain corresponding to the points in the Cartesian space [33]

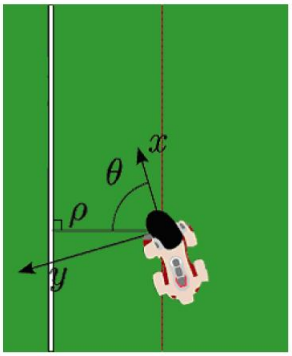
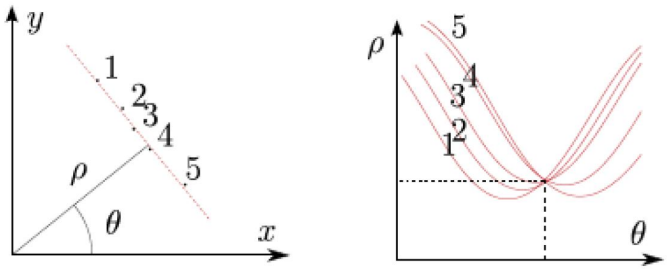


Fig. 5.7: (ρ, θ) are the parameters of a line expressed in polar coordinates [33]

Assume we are given a line described by ρ and θ; where ρ represents the distance from the origin (i.e. camera of the robot) to the line, and θ represents the angle between the X-axis and the line. We can obtain its corresponding Cartesian equation by (5.1), where (x, y) are the Cartesian points of a candidate point.

$$\rho = x\cos\theta + y\sin\theta \tag{5.1}$$

With this formula, to each (x, y) point in the image is associated a curve in the Hough domain (ρ, θ) as shown in figure 5.8. With polar coordinates, the set of values that is associated to each parameter is limited within intervals, such that $\theta \in [0, 2\pi]$ and $\rho \in [0, (M_2 + N_2)^{\frac{1}{2}}]$, where $M \times N$ are the dimensions of the image [39].



(a) (b)

Fig. 5.8: (a) points in the Cartesian space, (b) sinusoids in the Hough domain corresponding to the points in the Cartesian domain [33]

A pseudo-algorithm of the Hough Transform looks as follows:

- initialize the parameters[8] space:

  $S_\rho = \rho_{min}, ..., \rho_{max};$

  $S_\theta = \theta_{min}, ..., \theta_{max};$

- for each edge point (x, y) do:

  $for(\theta = \theta_{min}; \theta \leq \theta_{max}; \theta++)$ {

  $\rho = x\cos\theta + y\sin\theta;$

  $Voting[\theta][\rho] = Voting[\theta][\rho] + 1;$

  }

[8] If the values in the Voting array are below threshold, we conclude that no line is formed by the edge points.

- only the local maxima having a value in the Voting array higher then a fixed threshold are considered [33].

Our implementation of the Hough Transform

Before launching our Hough algorithm, we must first initialize the following parameters:

- the superior limits for θ is $(\frac{\pi}{60})$ and the superior limit for ρ is 263. The tables containing the discrete parameters[9] values will have a size of 60 cells for θ and a size of $(\rho * 2 + 1)$ for ρ,
- initialize the maximal threshold: $houghMaxThreshold = 0.2$. As the value of this parameter grows, the number of lines detected by Hough decreases,
- initialize a threshold for parallel lines filtering: $houghFloodingRatio = 0.1$. Any vote v' around a maximum such that $v' > houghFloodingRatio$ is not considered.

Once we're ready we call the Hough related functions, in a kind of main function located in a class that produces an array of lines from a given array of edge points. In our implementation we precede the follwing steps:

- check that we have at least 10 line points. Under this number of points, Hough is expected to provide a high rate of erroneous parameters for lines,
- fill in the Hough Voting Table by solving (5.1) for each point of the image and each parameter;

$addHoughPoint(intx, inty)$ {

$for(inti = 0; i < SIN\_TABLE\_SIZE; i++)$ {

$intrho = x * costable[i] + y * sintable[i];$

$intirho = rho/(1 << 16); //shift\ left\ by\ 16$

$//add\ a\ vote\ to\ lines\ with\ \theta = I\ and\ \rho = irho$

$houghTable[i][rho2index(irho)]++;$

$//add\ a\ vote\ to\ lines\ with\ \theta = I\ and\ \rho = irho - 1$

$houghTable[i][rho2index(irho + 1)]++;$

[9] In order to not calculate sin and cos on-line, which is time consuming, in our implementation an array is provided off-line, containing for each value of ?, its corresponding sin and cos value.

$//add\ a\ vote\ to\ lines\ with\ \theta = I\ and\ \rho = irho + 1$

$houghTable[i][rho2index(irho - 1)] + +;$

}

}

- create a list of all the maxima found,
- scan the list of maxima, and convert the polar parameters of each line into image parameters,
- save all lines in a "line-container" structure where each line is characterized by ρ and θ, the time when it was perceived and a color[10].

5.2.4 Newspaper detection

As the newspaper that we used for our work is almost white, we decided to employ a feature extraction technique based on blobs construction.
A blob consists on a set of pixels having same color. Once a white pixel wp is detected in the image, all its adjacent neighbors (2x2) are checked. If they have a white color too, they are added to the blob. Once we encounter a character or a field or coloured-line pixel in the newspaper, the blob creation is stoppped. As there are a lot of characters and images in a newspaper, a big number of different blobs is formed, and we have to put them together. Thus, a demanding algorithm had to be designed to paste the different blobs together.

Such calculations are not only time consuming, as we don't know in advance how many blobs we shall have, we have to declare a structure with a maximum size often bigger than its actual size. That's why we decided to detect the newspaper, exactly as lines by using the Hough transform.
This choice is relevant since the shape of the newspaper in the environment is rectangular, just like lines. Thus, the newspaper is represented by ρ and θ parameters, the time when it was seen and its white color. However, not only the newspaper is white, the lines are also half white, and errors might occur when during image processing, as only the white half of a line has been processed so far, the robot might believe that it is a newspaper and begin to search for its parameters.

To overcome such false positives, we assume that the robot must first scan all the image searching for lines, and if no line was found, but enough points were found to detect a newspaper, then launch the newspaper detection. As a consequence, the image might be scanned twice; first to look for lines, and if no line is identified, another scan looks for newspaper blobs.

Then we thought of la ess demanding solution where, while scanning the image, if we encounter white points and don't still know if they are line or newspaper points, we all save them in a structure of points. At the end of the scanning, the Hough algorithm is launched, either for lines or for the newspaper. This

[10] Here we have the parameters of the line in image coordinates only, and we will transpose the line parameters into the robot's system of coordinates later, in the Object Modeling step.

technique is faster and works pretty well, though some false positives occur when a the robot takes an image where only the white part of a line appears. In that case the robot will believe he saw a newspaper. This problem can be overcome, by pretending that the robot only looks for the newspaper if he reached the end of the blue line. The robot will stop looking for a newspaper during the rest of its work, once he went close to the newspaper, he recognized it, and turned around.

As we were in a hurry once the image processing part was finished, and had to implement the robot behavior and motion, we did not implement the necessary instructions to stop the robot looking for the newspaper after he has already found it. However, recognizing a newspaper instead of a line occurs rarely and the current implementation still gives good results.

5.3 Related Tools

We used two different tools during the Image Processing work flow. RobotControl (figure 5.9) is the remote debugging tool produced by the German Team [65] specifically for the Robocup competitions. GT uses it as a debugging tool, but we only used it to create logs, i.e. a kind of video recorded on the PC, by capturing the images that the robot sent to the pc by wi-fi. This tool is not robust since sometimes no log is taken when you click on the record button, sometimes it doe not connect to the robot, or it also happens that the window suddenly generates an error and closes. In all these cases, we had to re-launch the application until it worked properly.

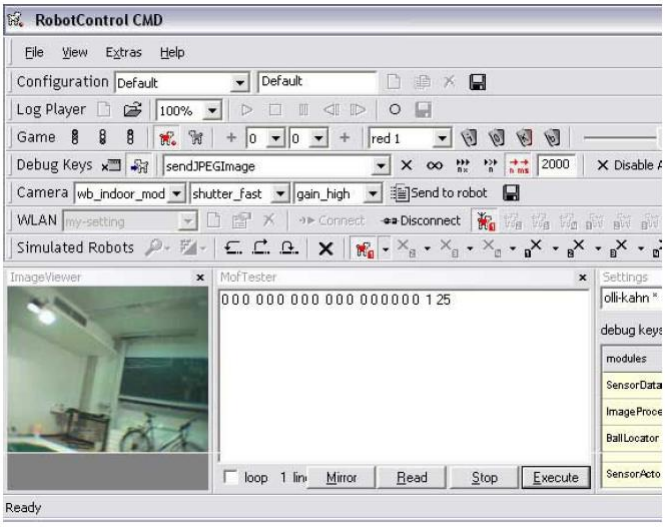


Fig. 5.9: Remote debugging tool: the robotControl window

Ipmonitor (figure 5.10) is a tool implemented by the SPQR Team. It is a kind of Perception simulator, which allowed us to debug the implementation of the Perception module, without using the robot, but using the logs previously recorded with RobotControl.

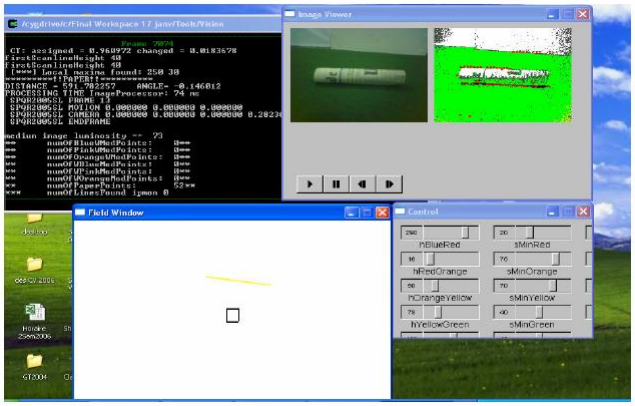


Fig. 5.10: Perception simulation: the Ipmonitor set of windows

Ipmonitor displays three windows. The Image Viewer Window illustrates each real image at the left and the segmented image at the right. It is possible to write some lines of code in the Ipmonitor class, so that it draws in the segmented image, all the edge points detected for an object and the lines found by the Hough transform.

The Field Window represents the position of objects detected with respect to the robot (represented by the square). Once again we can make Ipmonitor draw points or other features like lines.

The Control Window allows varying H and S chromatic components producing a different segmented image. The values already present in the Control Window, are those generated by our color segmentation algorithm.

5.4 Related Problems

AIBO's optical CMOS sensor presents several problems. The image acquiring tool of AIBO ERS-7 is a low cost CMOS sensor. The maximal resolution of the images is 208x160 pixel, converted in hardware at the YCbCr chromatic space, and are available at a rate of 30 fps. The use of such an optical sensor presents several problems:

- Blue cast (figure 5.11) appears on the corners of the images captured by the ERS7's camera, and it is a serious hindrance since the object recognition is mostly based on color classification. As the image is converted in hardware at the YCbCr chromatic space, it appears that the cast is added to the Cb chromatic component of the pixel (blue chrominance) proportionally to the distance from the centre of the image. SPQR solves this problem by operating a filter that separates such a chromatic component [34].

- Motion blur (figure 5.12), appears when the camera is moved (while an image is taken). Each line in the image is captured at a different (time) instant caused while taking an image when the robot moves its head rapidly, it becomes very difficult to detect interesting objects in the image.

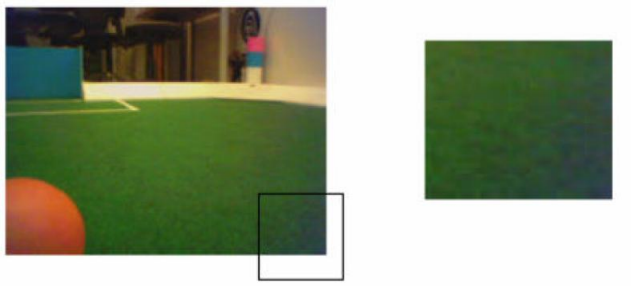


Fig. 5.11: Blue cast defect [34]

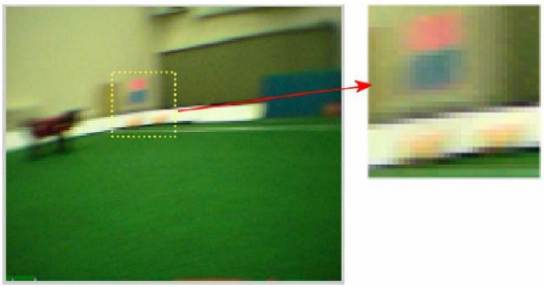


Fig. 5.12: Motion Blur occurrence [34]

- Sometimes stripes of different luminosity (fig.5.13) appear. This problem is treated during object recognition/detection, since pre-filtering of the image is too onerous.

Fig. 5.13: Stripes of different luminosity occurrence [39]

Chapter 6

Object Modeling

During the Perception process, we saw that we first recognize the objects in the image system of coordinates, and afterwards project them into the robot's system. In our work, the robot must walk straight on the lines. However the robot's walk is not perfect and when you give AIBO a motion command to walk straight, after some centimeters of walk he deviates aside the straight axis (figure 6.1).

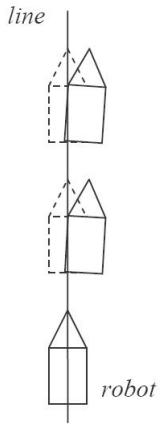


Fig. 6.1: Full line = the actual straight walk of AIBO, dotted shape = expected straight walk of AIBO

Thus AIBO must correct its trajectory to return in front of the line. In order to spare time, we implemented a method that stabilizes the robot along the line, by correcting the trajectory as described in the section: *Path following*. But to follow a line correctly, the robot must check whether the line detected in the image is the same as the one he was following or is a new one. Here the Object Modeling step plays an important role.

Indeed, if the robot perceives a new line, thanks to its LineModel - which keeps information about the previous lines' characteristics (color and position)- he can "understand" if this line is the same as the

one he is following by checking the odometry changes.

Object Modeling is also important when AIBO can't see an object anymore, but still is able to calculate where it is located. In this case when a robot looses track of the line he was following and does not see any other line around, we can make him turn back to the last position where he saw a line (unless he has reached the end of a blue-white line and must search for the newspaper).
Let's see the main steps of our Line Modeling implementation:

- Each new line added to the LinePercept structure, is compared to all the lines already saved in the LineModel which can store at most five lines.
- If a perceived line l is similar to a line ml in the LineModel, i.e. they have same color characteristics, and if odometry changes since l was seen tell that as the robot's body moved, the line still lies at the same place, then l replaces ml.
- If a perceived line is different from all the lines already saved in the model, it is added to the line model.

As the newspaper must only be recognized once, the NewspaperModel only consists of remembering whether the robot has already seen the newspaper or not. The information stored in the ObjectModeling classes. The Behavior Control level only consults these classes.

Chapter 7

Behavior Control

The BehaviorControl (or Deliberative) Module represents the decision making level, based on the world state information provided by the EnvironmentObjectLine and EnvironmentObjectPaper classes associated, to lines and newspaper respectively. They contain high level information over their reliability (i.e. probability that the object was really perceived), their angle and distance with respect to the robot's position. They also give information on the objects' color and their position in the robot's system of coordinates. Then the BehaviorControl Module, outputs motion requests for legs and head, as well as LED requests (e.g. lighting a specific LED when a line is recognized).

In our work we use the hybrid architecture as behavior architecture. This choice is justified by the fact that a deliberative architecture is too expensive in computational terms, since plans are generated on-line, and at the other hand the reactive paradigm does not fit to our working environment, since the robot actions are conditioned by some high-level goals that he pursues (e.g. find the right direction to go to the newspaper), and they must be taken into account.
Indeed, in the chosen architecture, there is a deliberative (or planning) layer maintaining high-level information on the environment and where we choose the right action, and an operational layer with low-level information where conditions are evaluated and actions are executed.

To spare time, we focused on existing tools and formalisms for plan designing which were already used by some of the RoboCup soccer teams, and we chose to work with the Petri Nets Plans (PNP) framework, currently used by SPQR [53].

We chose this modeling language since it allows for high level description of complex actions, such as sensing and conditional actions, action failures and action interruptions, concurrent actions ... [17]. Just like other behavior oriented languages, such as the Extensible Agent Behavior Specification Language (XABSL) [11] used by GT, the PNP framework provides for efficient plan designing, executing and debugging, however as a difference with XABSL, the semantic of Petri Nets is well defined [42], allowing thus for automatic verification of the properties of the behaviors. Furthermore, PNP also differs from the other languages in the higher efficiency of plan execution, due to the absence of computational expensive reasoning procedures during this process [69].

Finally after having a look at the plans produced by the different teams (i.e. SPQR and GT), we

decided to choose the Petri Nets formalism to design plans, since we think that it is more intuitive, expressive and readable than XABSL.

The tool used for plan designing through Petri Nets is Jarp. SPQR selected it among other graphical tools for Petri Nets representation (like PIPE, JFERN, or PNK), for its usability, portability and expressivity [17].

In the following section we will rapidly introduce the Petri Nets formalism, then we will describe the behaviors implemented in our work in section two, and the tools that we used for the creation of plans, will be presented in section three.

7.1 A short overview on Petri Nets

Petri nets are a graphical and mathematical tool for the description of information processing systems'. They consist of two types of nodes; places (figures 7.1(a) and (b)) and transitions (figure 7.1(c)) connected to each other by directed weighted arcs (figures 7.1(d) and (e)).

Fig. 7.1: (a) a place; (b) a place with a token; (c) a transition; (d) an arc labeled with the weight 5 ;(e) a unitary arc

For any kind of system represented through Petri Nets, a place may contain zero tokens or more, and they denote the state of the system. When at least one token is present in a place, that means either that the condition associated to this place is respected, or that some kind of resource is available in the system. The number of tokens in each place is called the marking of the system. A transition generally represents an event that happens in the system. There are two types of places; input and output places (figure 7.2).

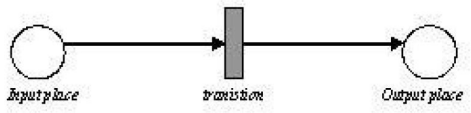


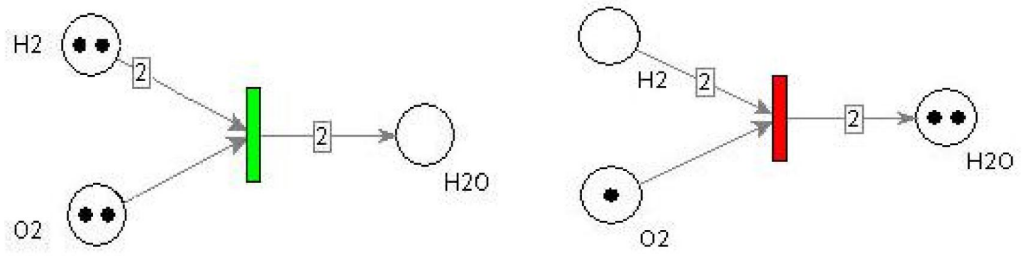
Fig. 7.2: Representation of an input and an output place

The marking of a Petri net changes according to the following rule:

- A transition t is said to be enabled, if its input places p contain at least as much tokens as the weight $w(p,t)$ of the arc linking p to t.

- If the event associated to the transition t happens, then the enabled transition is said to fire. If the related event does not occur, the transition does not fire[1].
- Once t is fired, the $w(p,t)$ number of tokens in the input place p disappears, and the output place p' of t will be filled with as many tokens as the weight $w(t,p')$ of the arc linking t to p'.

Let's apply this rule to the simple example of a water molecule creation as showed in figure 7.3.



(a) (b)

Fig. 7.3: (a) Initial marking of a water molecule creation plan, (b) Final marking of the same plan [17]

7.2 An overview on Petri Nets Plans

PNP is a Petri net which is able to represent the main concepts of the robot's behavior; initial state, ordinary actions, sensing actions (i.e. checking condition validity in order to resolve non-determinism), action failure and action interruption. Let's see how these concepts can be described through this formalism.

The initial state of the robot is described by the initial marking M_0 of the net. An ordinary action (figure 7.4) is expressed by a construct composed by five nodes, where places represent the different phases of the action execution, while transitions represent the related events.

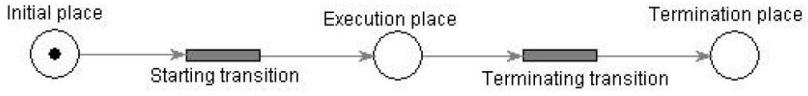


Fig. 7.4: Representation of an ordinary action

[1]When no event is associated to a transition it will fire automatically as soon as it is enabled.

We use a sensing action when we want to resolve a conflict between two or more conflicting transitions. It consists on checking which of the conditions associated to the transitions is verified, in order to fire that transition. In figure 7.5, *ps* represents the place where the property is sensed. In case the sensed property is true then the $tt(p)$ terminating transition is fired, otherwise the $tt(notp)$ termination transition is fired.

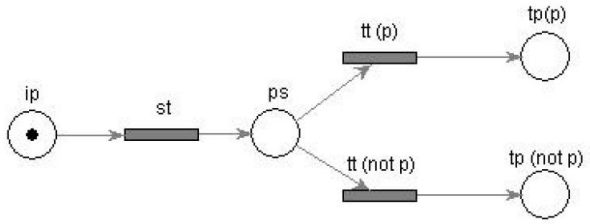


Fig. 7.5: Representation of a sensing action

The action structures presented above are elementary ones, and they can be composed by using the operators; sequence, conditional, loops, concurrent execution and interruption [69]. A sequence of two PNP is obtained by merging the terminating place of one of them and the initial place of the other (figure 7.6).

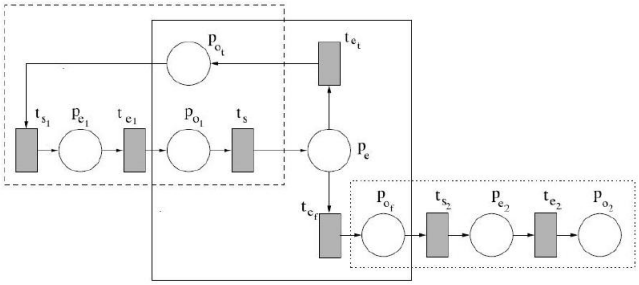
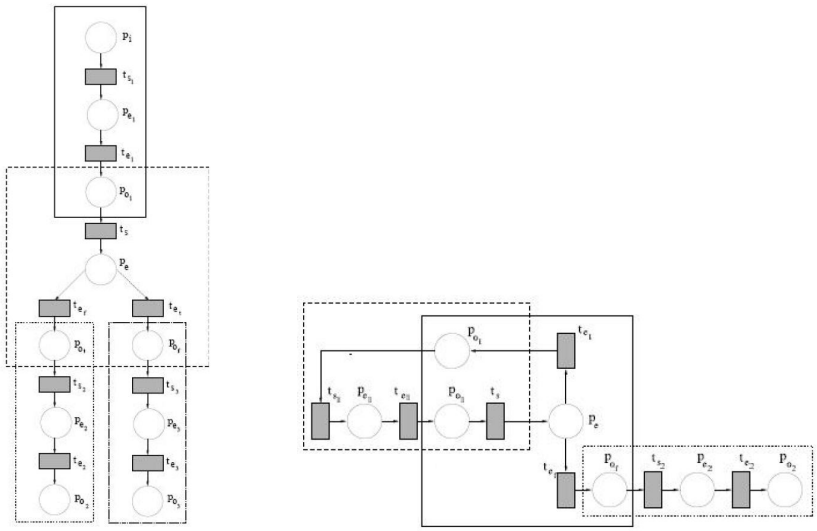


Fig. 7.6: A sequence of two plans [69]

Conditional (figure 7.7.a) and loops (figure 7.7.b) structures are created by using conditional actions to join a plan to others.

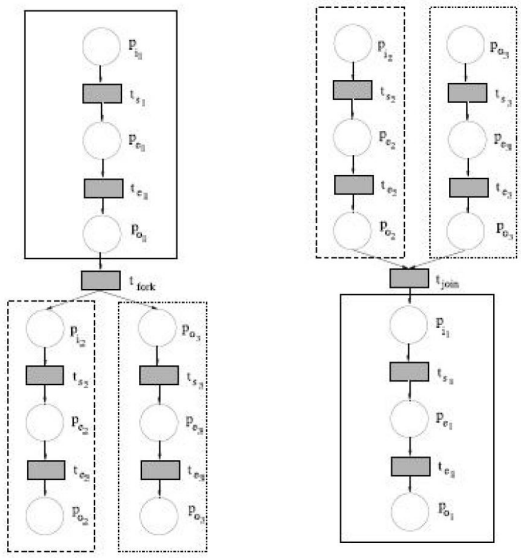
Structures with concurrent actions are created by simply adding a transition and linking arcs. Concurrent actions can be launched by a fork structure (figure 7.8(a)) and stopped by a join structure (figure 7.8(b)).

Interrupt structures are obtained by adding a new transition and arcs to the execution place of a plan (figure 7.9).



(a) (b)

Fig. 7.7: (a) A conditional structure; (b) A loop structure [69]



(a) (b)

Fig. 7.8: (a) A fork structure; (b) A loin structure [69]

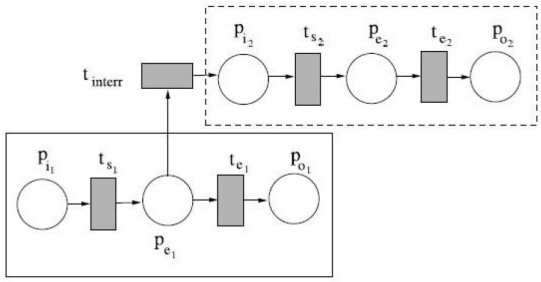


Fig. 7.9: An interrupt structure [69]

Now we will describe the grammar of the Boolean conditions in PNP. Boolean formulas in PNP are a sub-set of the propositional language which has an alphabet consisting of:

- the propositional connectives NOT, OR, AND,
- a finite non empty set of atomic symbols,
- the separation symbols '(' and ')' .

The set of formulas for the propositional language is the sub-set of strings having the following properties:

- An atomic symbol is a formula,
- If F is a formula, then (NOT F) is also a formula,
- If F1 and F2 are two formulas then (AND F1 F2) and (OR F1 F2) are also formulas.

7.3 Our plans

Now that we introduced PNP, we are ready to present our behavioral plans. As we tried to make plans as readable as possible, our plans are strongly structured, i.e. each plan is composed by several sub-plans themselves containing other sub-plans and so on. Here we will present only the most representative plans.

So far in our work we use seventeen plans, which can only be executed by the robot if we associate a file where we specify the conditions that are sensed in the plans. In this file, named conds.txt, to each sensing property; e.g. SeenMedLine or NearToPaper, is associated the object that is concerned by this action; e.g. medline and paper respectively, and the attribute of the object that the sensing action is checking; e.g. the reliability and distance respectively, and the conditions to be satisfied.
For example:

SeenMedLine obj=medline reliability>0.85
NearToPaper obj=paper distance<=200

We use three main plans in our work. Two plans consisting on walking on the lines (one plan for each direction), and a plan to get close to the newspaper[2]
These plans are each composed by other sub-plans as described in the following subsections.

7.3.1 Bi-colored lines plans

If the robot hasn't seen the newspaper yet, the GoToColWhiteLine plan (figure 7.10)[3] is executed, which makes him follow all bi-colored lines that are white on the right. If the robot has recognized the newspaper, to make him come back to its initial position the GoToWhiteColLine plan is executed, which makes him follow all bi-colored lines that are white in the left.

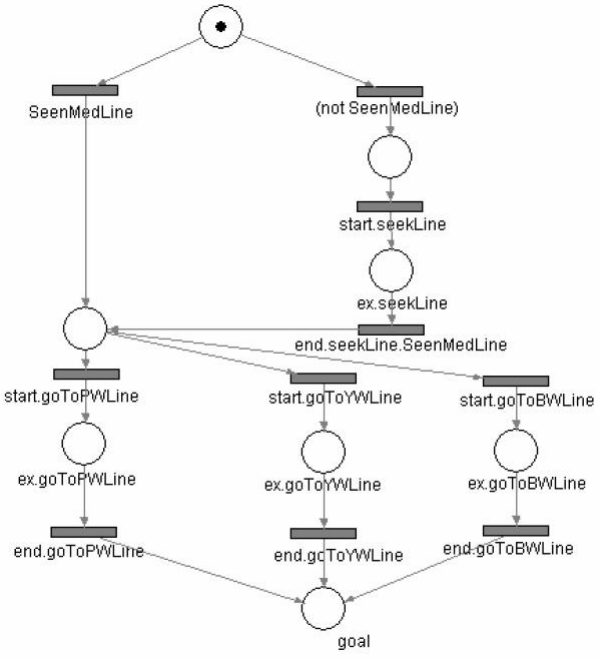


Fig. 7.10: Plan executed to follow the lines when the goal is to find the newspaper

The GoToColWhiteLine is the first plan to be executed when the robot begins its task. Actually when the robot is switched on, he is automatically positioned in an initial state where he stands still and looks straight (figure 7.22(a)).

The ideal process would be to launch the execution of GoToColWhiteLine plan, once the robot hears a signal notifying that the newspaper has arrived. But making AIBO listen and recognize sounds seemed

[2]We didn't have time to implement the newspaper grabbing behavior.
[3]The goal place represents the last terminating place to be marked in the plan. If no goal place appears in the plan, this means either that it is a sub-plan and the goal is in the upper-plan, or that it is a "never-ending" plan.

to be a very challenging and tough job for a few months work and beside that, SPQR does not use AIBO's hearing sense, so we could not benefit from any support or feedback by the researchers around us. Thus, as we did not implement sound recognition, we used the robot's head touch sensor to launch the GoToColWhiteLine plan, which is executed after clicking three times on the robot's head.

SeenMedLine is a sensing property that tests the current reliability that a "medline" environment object was seen. The reliability parameter is calculated in the EnvironmentObjectLine class as a decreasing exponential:

$$reliability = \exp(\frac{-d}{norm})$$

Where d quantifies the time when the last line was seen (in seconds*1000), and norm is an empiric normalization constant 5000 i.e. 5 seconds.
If the robot has not seen a line, he will start seeking one. The seekLine (figure 7.11) plan is only compound by simple actions (which names begin by the "Act" prefix).

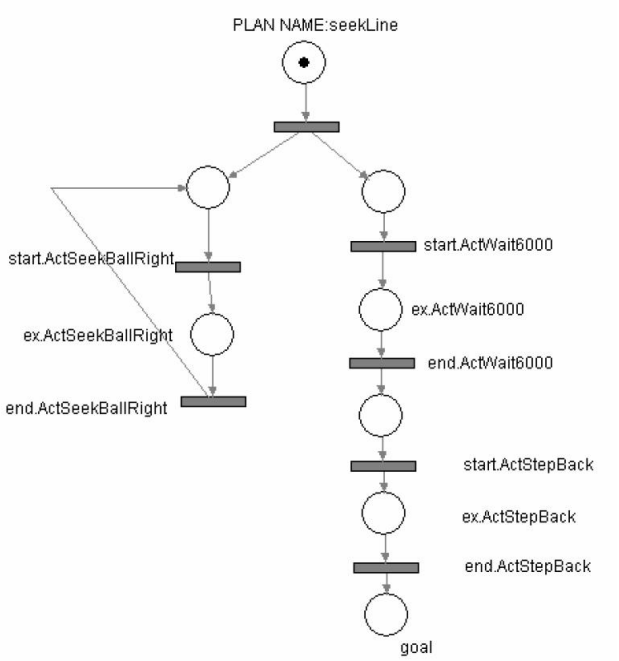


Fig. 7.11: Plan where the robot searches for a bi-colored line by moving its head during 6 seconds

ActSeekBallRight[4] is a head motion action that makes the head turn (and the camera at the same time) from right to left. ActWait6000 is an action that consists on simply counting six seconds, and it is

[4] ActSeekBallRight is a head motion action implemented by SPQR, which is independent from the ball object thus we could use it.

executed in parallel with the ActSeekBallRight action.
If after six seconds the robot's state is still in the seekLine plan, that means that he has lost track of the line (as he can't see it in front or aside), then we make the robot walk some steps back with the leg motion action ActStepBack[5].

If the robot has seen a bi-colored line, he will enter the appropriate sub-plan according to the color of line. Figure 7.12 represents the plan for a yellow-white line.

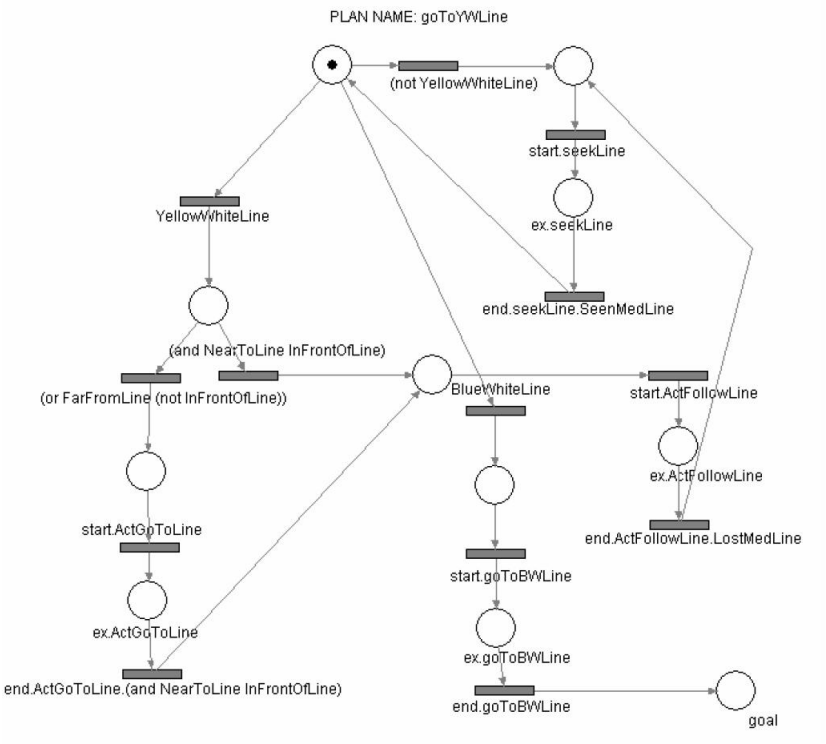


Fig. 7.12: Plan executed when the robot recognizes a yellow-white line

Once the robot enters this plan, he verifies if it is ready to follow the line or not. When the robot is in front of the line (i.e. the angle of the absolute robot pose to the line is higher then 85ř) and near to it (i.e. the distance between the robot and the seen line is inferior to 30 cm), then he is ready to follow the line, and he performs the ActFollowLine action. But if he is far from the line or not in front of he will go toward the line until the conditions for following line become true.

Now suppose that while following it, the line comes out of the robot's sight (LostMedLine). Then the

[5]We can make the robot go at the position where a line was seen for the last time thanks to the LineModel, but it requires more computations. While making the robot simply go back, is less demanding and works very well since the robot is always either going to the line or following the line, so when he looses track of the line, the line is never too far.

robot will enter the seekLine plan to find it. While seeking for a line the robot will either see a yellow-white line and remains in the same plan, or he will see a blue-white line and enter the goToBWLine plan. In the following sub-sections we will explain how the ActGoToLine and ActFollowLine actions are implemented.

Go to line

As we said before, from the object medline, we can extract its distance and angle with respect to the robot. Considering those two values, we want the robot to perform the correct and necessary rotation and translations to align the robot with the line. Formula (7.1) was used to calculate these translations:

$$\begin{aligned} translationX &= translationGain * (d) * \cos(\alpha); && // \ forward\ speed \\ translationY &= translationGain * (d) * \sin(\alpha); && // \ lateral\ speed \\ rotation &= rotationGain * \alpha; && // \ rotation\ speed \end{aligned} \tag{7.1}$$

Where d and α correspond to the distance and the angle of the robot to the line.

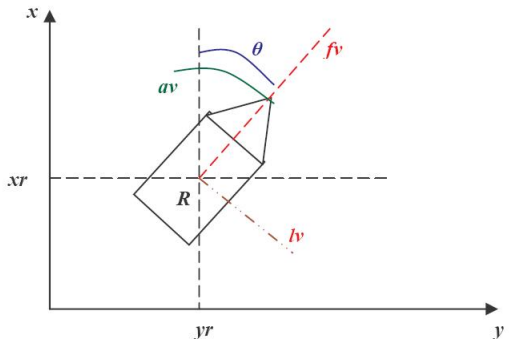


Fig. 7.13: A simple model of a robot

However the robot's joints have limited motions (see the model specifications), i.e. they have maximal translation and rotation values. In figure 7.13 we can see a simplified model of the robot where we can distinguish:

- xr, yr are the coordinates of the reference point of the robot, located in R
- θ represents the heading angle of the robot
- av is the angular velocity
- fv is the forward velocity
- lv is the lateral velocity

The velocities are constant and known values, which express the maximal speed at which the robot can walk forward, walk aside or rotate. For security, when one of the translations or rotations computed as described in (9.1) is higher then its maximal value, then the robot will automatically set this value to the maximum permitted. Let's illustrate the problem by an example.

Suppose we have the following constraints:

fv= 100;

lv= 40;

But we have calculated:

translationX= 100;

translationY= 50;

Then the system will protect itself by considering:

translationX = 100;

translationY= 40;

In this case the motion performed drives the robot in the wrong direction as shown in figure 7.14:

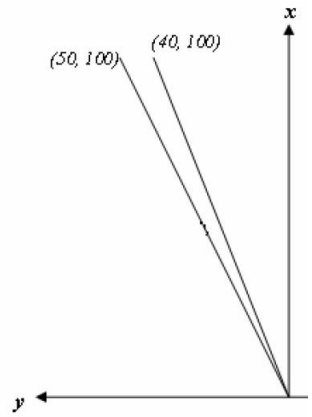


Fig. 7.14: Variation of the robot's trajectory when one of the translations changes

The solution is to "saturate" these values in order to preserve the direction:

$If(|translationY| > maxspeedY)$ // *high lateral speed* {

$Scale = |maxspeedY/translationY|;$

$translationY = sgn(translationY) * maxspeedY;$

$translationX = Scale * translationX;$

$rotation = Scale * rotation;$

}

Where *sgn* is a function that gives 1 if its argument is positive, and -1 if it is negative.
Then we obtain:

translationX = 80;

translationY= 40;

This motion has the same direction as the one with a lateral and forward speed equal to 50 and 100 respectively. Of course the same reasoning is applied when the forward and the rotations speeds are higher then their limits.

Path Following

We can distinguish mainly two categories of localization methods, *landmark-based* and the *model-based* (see section 2.2 on Autonomous robot localization).
Landmark-based approaches are the most popular ones, because of their simplicity, and their resemblance to human navigation in unknown spaces. As we noticed in section 2.2 (Autonomous robot localization), robots working with models generally are equipped with sophisticated and expensive sensors, benefit from significant processing resources, and are hardly suitable for real time applications.

As AIBO was designed so that common people can afford him, its sensors are low-cost ones, the processing resources are limited and above all this we want it to be real-time.
Moreover, as the dog-like robot is supposed to be used as a "pet" in a house, if the robot is given a standard house model, adapting it to one's house, requires considerable time and expertise which can't be provided by the future robot users. On the contrary, if we program the robot to localize itself through floor-marks, the users will just have to place the marks in a certain order and direction (see Annex) in their house. For these reasons, we decided to work with marked-path following.

Our mark-based approach uses bi-colored lines as the path to follow (see figure 5.5). In our work, path following means walking straight on the lines.
Of course, instead of walking straight on it, the robot could simply go on the direction of the line, but then, the robot should be continuously tracking the line using head motions in order not to lose it, and a dynamic head tracking motion is a dangerous (because of the lateral limits of the head motion) and a very accurate task to be performed. Furthermore, in case the robot looses track of the line, he will do superfluous head and walk motions (to correct its position with respect to the line position) that are

time consuming. That's why we decided to make the robot walk straight on the lines, by using a visual servo[6] system for the line following behavior as described in [51].

According to its current position and that of the line, the robot can translate (figure 7.15(a)), rotate (figure 7.15(b)), or do both of them (figure 7.15(c)).

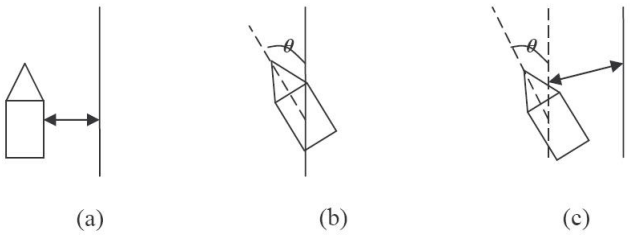


Fig. 7.15: (a) Translation; (b) Rotation; (c) Rotation and Translation

The kinematics model of AIBO when we want him to follow a line is described in figure 7.16. As we want the robot to walk forward, the lateral velocity is not taken into account, thus only the forward and rotation velocities are entered as input to this model. The model will then output the necessary forward, lateral and angular translations that the robot should perform to follow the line, here represented by x, y and θ respectively.

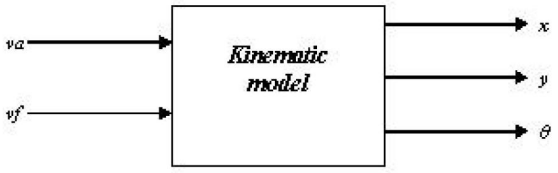


Fig. 7.16: Kinematic model for Line following

[6]Visual servo system is a feedback control system based on visual information. The implemented system works in two steps. The first step consists of line recognition in the image, and the second step corresponds to motion commands sent to the robot.

The kinematics system is described by:

$$x' = vf \cos\theta$$

$$y' = vf \sin\theta$$

$$\theta' = u$$

Where x', y' and θ' are the first derivates of x, y and θ.

When the robot walks straight on the line, rotation and translation are almost equal to 0. So, assuming that θ tends to 0 we have as first-order approximations:

$$x' = vf$$

$$y' = vf * \theta \rightarrow \theta = \frac{y'}{vf}$$

$$\theta' = u$$

It was observed that kinematics is stabilized if:

$u = Kp * y + Kd * \theta$ where Kp is the translation gain and Kd is the rotation gain.

Then we have:

$$\theta' = Kp * y + Kd * \frac{y'}{vf} \rightarrow (\frac{y''}{vf}) = Kp * y + Kd * (\frac{y'}{vf}) \rightarrow y'' - Kd * y' - vf * Kp * y = 0$$

The characteristic polynomial of this system results to be:

$$x_2 - Kd * x - Vf * Kp = f(t) \qquad \rightarrow \qquad x_2 - Kd * x - Vf * Kp = 0 \tag{7.2}$$

Where $f(t)$ is the oscillation of the robot around the line while time t passes by.

And the solution of the linear system has the form:

$$Y(t) = A * e_{xt} \tag{7.3}$$

As in equation (7.3), x is a root obtained by solving (7.2), we can write:

$$Y(t) = A * \exp_{(a_j+b)t} \rightarrow Y(t) = A * \exp_{(a_j)} t * bt \tag{7.4}$$

Where a_j represents the complex part of the root, and b represents the real part of it, and where we have:

$$\exp_{(a_j)t} = \begin{cases} \cos(at) + j * \sin(at) & \text{if } x = b + a_j \\ \cos(at) - j * \sin(at) & \text{if } x = b - a_j. \end{cases}$$

Then we can write:

$$Y(t) = A * (\exp_{bt} * (\cos(at) + j * \sin(at)) + \exp_{bt} * (\cos(at) - j * \sin(at)))$$

According to the sign of the real part of the root, we obtain the system behaviors described in figure 7.17 and figure 7.18.

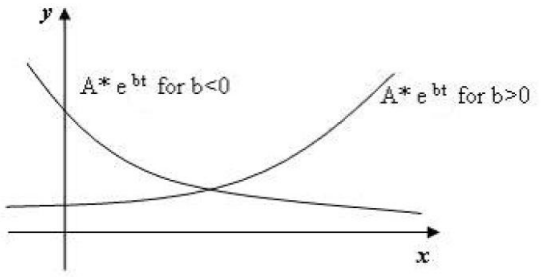
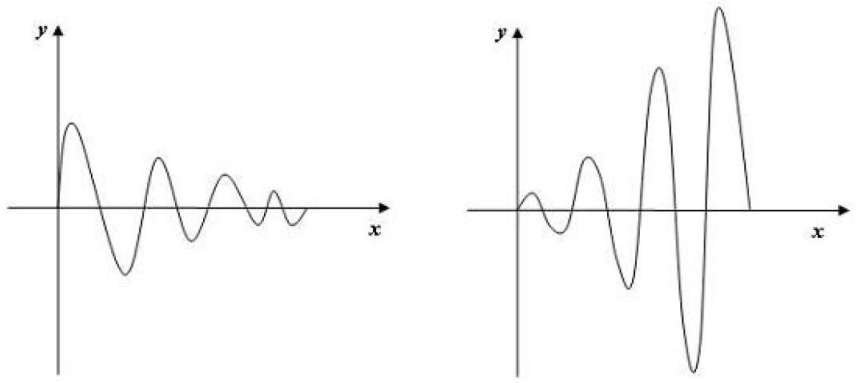


Fig. 7.17: if $b < 0$, $A * \exp_{bt}$ behaves like a decreasing exponential stabilizing around the x-axis, if $b > 0$. $A * \exp_{bt}$ behaves as an increasing exponential moving away from the x-axis

The behaviours we're interested in, are those represented by figure 7.18(a). We can extract some properties for the Kd and Kp values, that will help us obtain an exponentially decaying sinusoid;

$$Kd < 0 \qquad and \qquad Kp < -(\frac{Kd^2}{8 * vf})$$

Of course the solution of this system only gives an idea of the values we can choose for Kd and Kp, so the robot was tested with different values for those parameters as well as for the forward velocity parameter until the best values were found. Indeed if the absolute value of Kp is very big than the number of oscillations will be big as well, at the other hand if this value is very small, the number of oscillations will decrease but the system needs more time before stabilizing.



(a) (b)

Fig. 7.18: (a) behaviour of the system when $b < 0$; (b) behaviour of the system when $b > 0$

It is then necessary to perform tests with different values, in order to adapt the system's response to the environment specifications, i.e. the length of lines.

The optimal values considered in our work for the ActFollowLine action are:

rotationGain=-1.4; //Kd

translationGain=-(rotationGain*rotationGain)/(8*maxspeedX);//Kp

maxTurnSpeed = fromDegrees(30);

While the optimal values for action ActGoToLine are:

translationGain=0.5;

rotationGain=0.8;

7.3.2 Newspaper plans

Figure 7.19 describes the GoToPaper plan, which is executed when the robot sees the newspaper. Here we can distinguish two different seeking plans: seekPaper and SeekLostPaper.

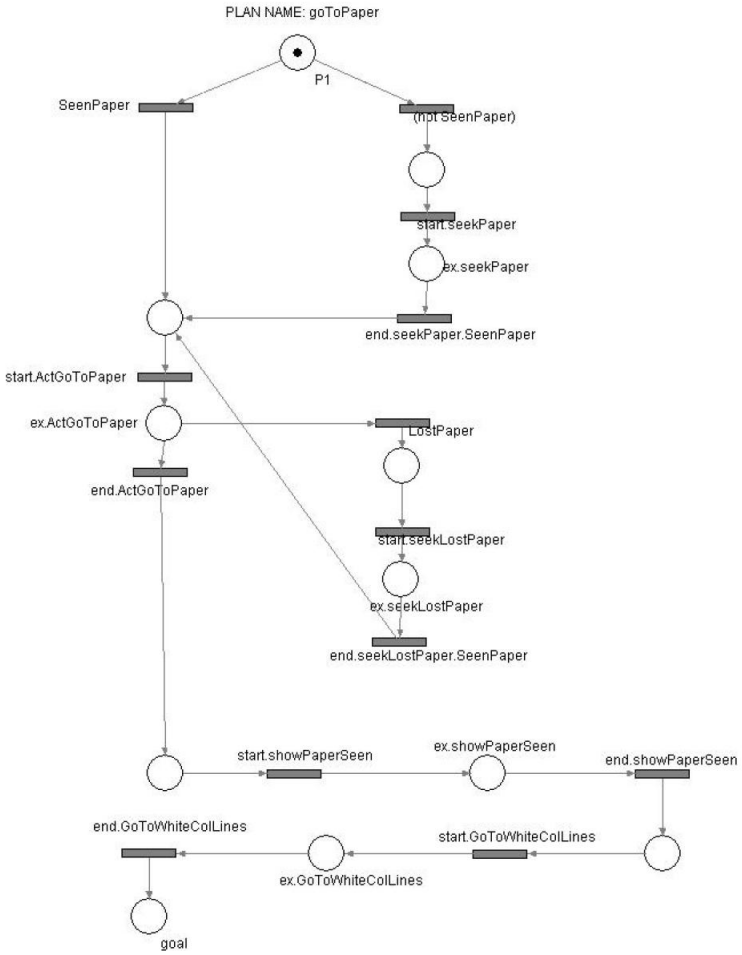


Fig. 7.19: Plan for going to the newspaper

The seekPaper[7] plan (figure 7.20) is used when the robot has reached the end of the blue-white line, and can't see the newspaper. Generally this happens because the newspaper is too far[8]. Then we just have to make the robot go closer by making him go one step forward.

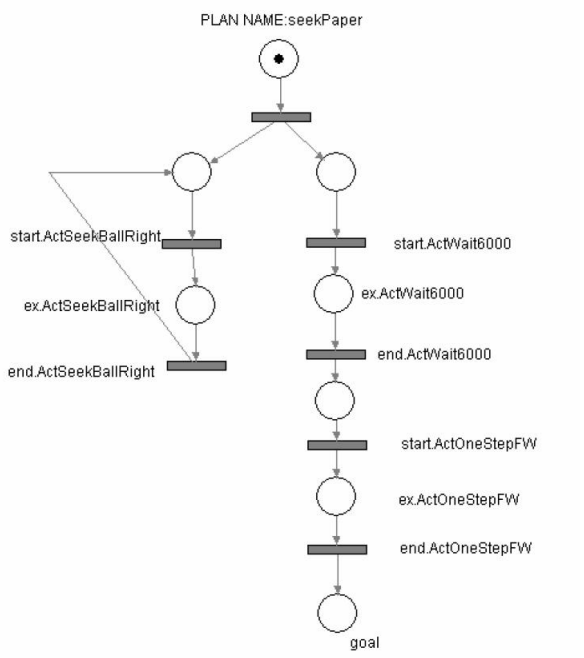


Fig. 7.20: Plan where the robot searches the newspaper by moving its head during 6 seconds

SeekLostPaper is the plan used when the robot looses track of the newspaper. This plan is slightly different form seekPaper. Indeed here the robot cannot afford to walk forward because he might walk on the paper. Thus, in SeekLostPaper the robot just searches the newspaper with its head, keeping its body motionless until the newspaper is seen again.
Once the robot goes close enough to the paper by executing its ActGoToPaper action, in order to show that he recognized the paper and is close to it, the robot executes the plan showPaperSeen, which makes the robot rise its head and open its mouth for several seconds, and then turn around (180ř), to be ready to follow the lines in the reverse direction. Now all he has to do is enter the plan GoToWhiteColLines which will make the robot follow all lines that are white on the left. This plan and its sub-plans have the same structure as GoToColWhiteLines and the corresponding sub-plans.

[7]The seekPaper plan is very similar to the seekLine one, at the difference that after 6 seconds of no object-detection in seekPaper the robot walks forward, while in seekLine he steps back.

[8]The newspaper might appear in the image but as only the bottom half of the image is processed, this object is not recognized.

7.4 Related Tools

Now that we introduced the Petri Net formalism, we will have a look at the tool that implements it: Jarp (http://jarp.sourceforge.net/). Figure 7.21 presents the window that appears when Jarp is launched, and a line of explanation for the lateral buttons is given.

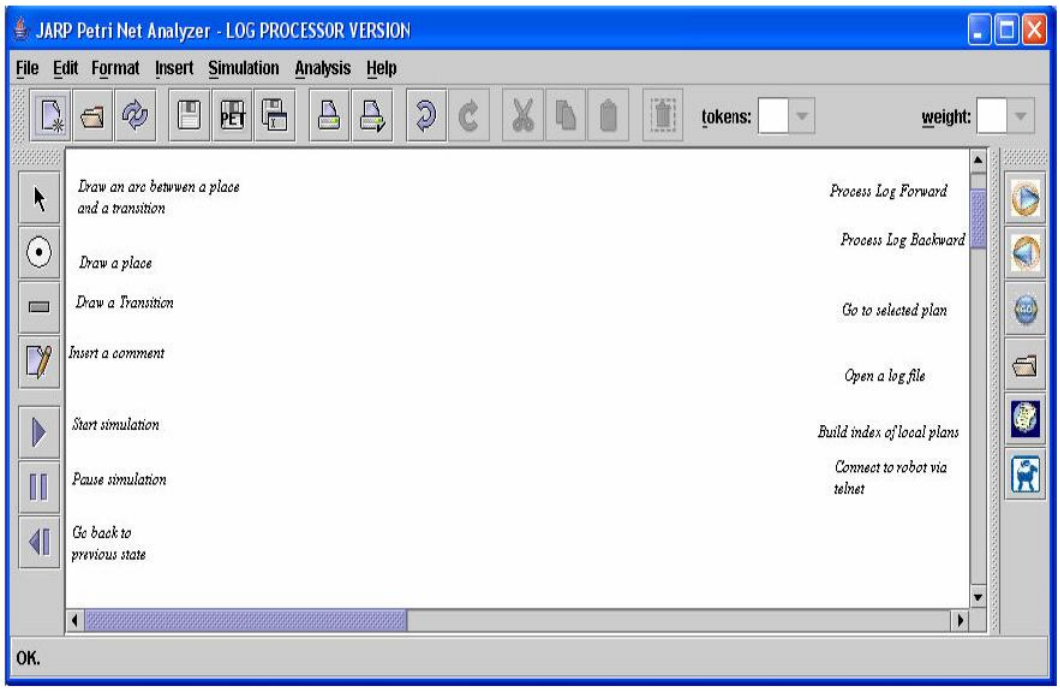
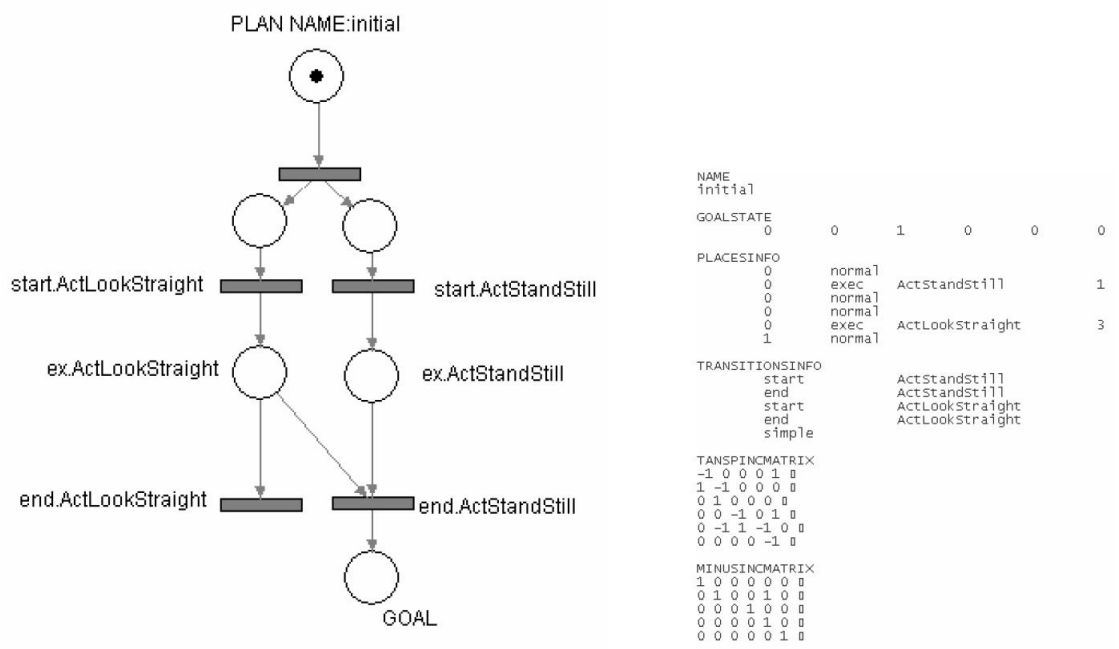


Fig. 7.21: Jarp windows

As usual, we find the File Menu allowing to open or save a plan in different formats: PPM, JPEG, GIF, PNG, ARP 2.4, JARP 1.1, and PNML. None of these formats is supported by AIBO's plan executor. This plan executor receives as input the incidence matrix representation of the PNP which is called the Pet format. For example, to the plan of figure 7.22(a) corresponds the PET file in figure 7.22(b). As we can see, producing such a matrix is tedious and more error prone than simply drawing a plan.

That's why SPQR created a tool to translate a plan saved under PNML format into PET format [17]. In the right side of the Jarp window in figure 7.21, appear six buttons, which don't belong to the standard Jarp tool. Jarp was extended by SPQR in order to permit plans debugging on-line. When the robot is executing plans, a log file is produced where information about this process is stored. This log can be parsed by the extended Jarp, so that we can see how the robot "navigates" from one plan to another, with the scope to identify wrong behaviors. This debugging tool is user friendly and rapid, but unfortunately too rapid, indeed the robot switches so fast from one plan to another that it is almost impossible to follow tokens propagation.




```
NAME
initial

GOALSTATE
        0       0       1       0       0       0

PLACESINFO
        0       normal
        0       exec    ActStandStill           1
        0       normal
        0       normal
        0       exec    ActLookStraight         3
        1       normal

TRANSITIONSINFO
        start           ActStandStill
        end             ActStandStill
        start           ActLookStraight
        end             ActLookStraight
        simple

TANSPINCMATRIX
-1 0 0 0 1 ▯
1 -1 0 0 0 ▯
0 1 0 0 0 ▯
0 0 -1 0 1 ▯
0 -1 1 -1 0 ▯
0 0 0 0 -1 ▯

MINUSINCMATRIX
1 0 0 0 0 0 ▯
0 1 0 0 1 0 ▯
0 0 0 1 0 0 ▯
0 0 0 0 1 0 ▯
0 0 0 0 0 1 ▯
```

(a) (b)

Fig. 7.22: (a) A plan giving motion commands; (b) corresponding Pet file

Chapter 8

Debate on robots

The future described in science fiction books or movies, with intelligent robots being part of the daily life in society, be it as perfect soldiers, companions or slaves, will have to wait a few years before becoming true.
This debate is based on the following articles and websites: [4], [27], [2], [28], [49], [41] and [40].

8.1 Technical limits

After our experience with AIBO, we can point point out some technical factors which impose practical limitations on robots.

Robot sensors suffer from perceptual limitations, as the perceptual range of most sensors (such as ultrasonic transducers, cameras) is limited to a small range around the robot. Sensor measurements are typically corrupted by noise, and often, the distribution of this noise is not known.

At the other hand robot motion is inaccurate, and odometric errors accumulate over time. It gets even worse in cases of drift or slippage. As the evolving environments are complex and dynamic, making it principally impossible to maintain exact models and to predict accurately.

Real-time requirements demand for accurate models to be appropriate in real-time situations, but the most accurate the model is, the less is it suitable for such situations.

There are also technical reasons that can discourage user's acceptance of robots.
Some robots are big and heavy, especially when loaded with supplies, that's why structural variables must be evaluated; e.g., elevator weight limits, load bearing capacity of floors, or storage space.
Furthermore, like any mechanical devices and computer systems, robots will occasionally require maintenance and service.
They also pose problems in terms of security; since robots may be used to transport valuable supplies and regulated items (e.g., prescription pharmaceuticals, medical records), potential users should carefully evaluate a robot's defenses against theft and tampering.
And finally we can mention the problem of cost; not everyone can afford a robot at home, that's why for

the moment they are only used by research and very little by common people.

However besides their defects robots have much success, and we will now expose some the reasons to this success.

8.2 Beyond technique

As we already said, besides being house maintenance slaves, valuable instruments for science promotion, or human lives savers [40], robots are having a growing success in the health care field.

But what are the reasons of such a success? Why using robots in health care?
Using robots for health care brings benefits at different levels. The most evident benefits are measured in economic terms. In fact in medicine, a remote surgery robotic device like Zeus, by providing better quality of surgery allows reducing hospital stays, and eventually the amount of personnel required.

The pharmacy market also benefits from robots, which can assist or replace usual white-coated pharmacists, in performing some tasks such as filling and labeling vials, or handling prescriptions. These tasks are not all more accurate or efficient than those performed by a usual pharmacist, but they are performed faster and then money is saved since less technicians and pharmacists are employed.

In the same range of ideas, nursing robots provided with Internet connection, through which a doctor can observe, hear and talk to patients at remote locations, allows reducing travel costs and increase productivity, since the doctor can visit patients more often. Furthermore as it is too expensive to pay individual nurses for residents that need round-the-clock care, providing them with a monitoring and nursing robot is a cheaper alternative. At the other hand this alternative does not only accommodate for money problems, it also solves the problem of staff shortages in nursery fields.

But those that financially try to benefit the most, are the big robotic manufacturing companies that consider society as a net of consumers and take profit of its blind belief in technology. However invent products just to make us consume as much as possible (e.g. dog clothes, or cell jewelry), do not the same impact as robots which change our way of living!

Some people would agree with this financial point of view, and they are not totally wrong since we live in a society where money "rules". But generally those people are not those that fear to lose their jobs because they are likely of being replaced by robots.

No worry, the robotic manufacturers and researchers can also list a range of practical advantages for those who think they could be victims.
Indeed, robotics is still in an experimental stage and robots are far form being a substitute for human beings. Either way, they will just assist people in their work and not replace them.
Assistance means allowing workers to perform more fulfilling tasks, for example instead of being a delivering person, a pharmacist visit patients. Thanks to robotics more quality work will be provided, for example in medicine, robotic surgery is more accurate and makes patients loose less blood during operations and recover faster. It has also been observed in some studies, that having robots around patients, not only decreased nurses' stress levels but also gave them something to discuss with their elderly patients.

But the principle reason of robots use in health care is the increasingly aging population in the world. According to the Census Bureau's projections, the elderly population will more than double by the year 2050.

As a society we have neither the financial commitment nor the trained medical personnel required to care for the growing number of elderly, unless we embrace a new paradigm, that of providing health care for the elderly at home. Once again it's not only a financial problem; most elderly people report that they prefer to remain living in their homes for as long as possible. Thus, robotic technology is the key solution that allows elderly people to remain in their homes longer.

The quality of health care and well-being of elderly remains an open question, but the Japan society seems less skeptical.
"Japan developed a postwar society that worships economic and technological development," says Yokoyama, the Yamato City Hospital doctor. "Therefore it is absolutely natural to rely on technology to fill in for the breakdown of traditional family roles, which accompanies rapid economic growth in our society." But the "belief" in technology is not the only reason that pushes Japan society to adopt robots; these are also expected to be better than people as confidants. "Japanese culture encourages people not to show their emotions to each other," says Yokoyama. "So it's actually easier for them to express their emotions to a robot - a cold, technological creation."

But something can also be done for elderly in nurseries who don't have any immediate relatives nearby, or no companion to share affection. In this direction, hundreds of clinical reports show that when animals enter the lives of aged patients with chronic brain syndrome, the patients smile and laugh more, and become less hostile to their caretakers and more socially communicative. Other studies have shown that in a nursing home or residential care center, a pet can serve as a catalyst for communication among residents who are withdrawn, and provide opportunities (petting, talking, walking) for physical and occupational rehabilitation. More generally, the research literature has established that the physiological health and emotional well-being of the elderly are enhanced by contact with animals [16].

However, many elderly live in places where pets are prohibited either because some psychological conditions of the patients make the ownership of an animal difficult, or because animals are prompted to cause allergies or carry diseases.

That's why electronics giants including Sony and Matsushita Electric Industrial and other companies have developed A.I. cats, dogs, Koalas, jellyfish and other interactive pets ranging in price from several hundred dollars to several thousand.

Robotic animals don't provide the same well-being as real domesticated animals, but this is yet a positive step forward. Once again we shouldn't see this problem as a future replacement of animals by robots; of course people having the choice can still prefer real animals.

We believe that robotics is a promising technology. However we are sceptical about the idea of expecting from it a solution to social problems. We believe that technology is a solution for scientific problems and culture is a solution for society issues.

The wrong solution to the right issue

What is amazing is the fact that the robotic technology, is not even questioned, it looks like obvious that technology must solve the problems.
Take for example the issue of staff shortage in the nursery filed. The reaction of overcoming this lack by the use of robots is likely to continue cooking with a broken cooker while it can be repaired. We mean that the issue is not well tackled. Indeed, if the youth is not motivated by the nursery field, is maybe because it is not enough well paid for the amount of work and sacrifices that this job requires. Or maybe it is not motivating because nowadays nursery is under-estimated and it is more "pricing" to study law or medicine. If we tackle the problem from such point of view, the issue should be resolved by the governments that should promote this trade. Another solution could be training and employing foreign nurses, who suffer from unemployment in their countries. Thus the problem can be solved in "human" manner, by promoting the future of human beings and helping poor people.

The same remark is valid for projects consisting on equipping elderly houses with robots, which connected to Internet, allow to family, friends, and community members to log onto the robots and spend quality time with their elderly relatives or acquaintances. We believe that if family members wanted to contact this elderly, they can simply do it by phone. And if phoning is not good enough (because one can't see his interlocutor); then by using a webcam!
Thereby, some would say that a webcam won't fit, since elderly would feel observed and would loose their privacy. While the robot is perfect, because when it is not with you, your privacy is preserved. Hence it seems to us much easier to turn off or cover the webcam!

Anyway this little notice is only aimed to show you what excuses people are able to invent, just to force you agree with their ideas, the real problem we wanted to point out in this example, is that if there is few communication between people, it is not because the technology of the communication middle is not sophisticated enough, it is because people don't "wish" to communicate. The problem consists thus on making people care about their older family members.

Hidden effects of robotics technology

We raised several aspects hidden in the propaganda fostering robotics.It claims that robots will allow human to do only qualified jobs, since machines will be performing the "devaluing" work. This idea sounds to us a bit discriminative. It is evident that not every human being has the financial and intellectual means to learn and perform qualified jobs, and those human beings have to live and supply for a family, while a robot will only supply for its already rich manufacturer.

Robots in elderly houses will be connected to Internet so that the person can contact family, friends or caretakers. What if a hacker takes control of the robot and gives it commands to harm the person it is in charge of? What if a thief comes to know every corner of the person's house? What if swindlers come to get all personal information on the person? All these questions aim to show that Internet in care robots, exposes the patient to the dangers that implies the wild virtual world.

Another phenomenon we would like to point out is the decreasing sense of responsibility at work. The same phenomenon can also be observed with computers; when a problem occurs and a bad or no solution is given, people tend to accuse the computer or the program for its dysfunction. Is not the use of robots

in health care likely to provide excuses to get rid of law suits that heavily affect the field?

The paradoxes

We also observed that the robotics technology impact, presents several paradoxes. In fact, while some people are likely to loose their jobs, others will have to work even more. Remember the example of the doctor who can remotely visit its patients thanks to an Internet connection provided robot. Before he could only visit his patients once a month, now he can do that once a week no matter where he is! This argument reminds us of the cell-phone. One main reason, for which people chose to adopt cell phones, is the possibility of being permanently available for contact. Hence, many businessmen complain about the fact that they are "too easily" reachable avoiding them to rest and quietly enjoy their holidays. Doesn't this new technology risk leading to similar problems?

A second paradox, lies on the project of using robots for sick, elderly or disabled persons. Hence, generally sick, old and disabled persons can badly care for their needs, but a robot does need maintenance, how could they care for the robot? It looks like a new generation of nurses is coming; the robot nurses, probably technicians that shall come to fix or update the robot!

Another paradox was observed on the impact of robots, on elderly physical efforts. Sophisticated mobile robots use in medical centers, provide opportunities for physical and occupational rehabilitation and recreational therapy. But it is not the case at home, where people won't walk after the robot, because they will be afraid of having an accident especially if the robot is big and encumbering. Thus letting elderly live at home with a robot, will maybe avoid loneliness, but won't help its physical conditions get better.

8.3 Conclusion on debate

Although literature seems optimistic on health care robots, a lot of work is still to be done. Indeed, robots still suffer from technical limitations, for example those able of connecting to Internet are far from being suitable for several reasons like having a web browser interface used to control the robot which is so intuitive that little or no training is needed, so that the operator can focus on the social interaction, not the robot. And the robot must also be designed so that people feel comfortable with it, and do not need to learn any new skills to have gratifying interactions with it. iRobot has already created several robots of this kind, and will continue working more on them.
At the other hand users (patients or employees) are skeptical on adopting robots in their lives or works, and experts intend to do their possible to foster robots' adoption.

Still most studies of robot performance in health care organizations have evaluated the use of only a single unit. Hence, additional analysis is needed to consider the implications of using two or more different robots in the same space in order to make sure that they do not collide or otherwise interfere with each other and with humans.

In the past few years though, research in technology has proved to evaluate tremendously rapidly, overcoming the physical limits faster than the time a user needs to get used to a technology. Thus, the

real problem is to know how to manage with robots when they will be ready to invade our everyday lives! The future is uncertain but society tries to get prepared to such a future; the proof appears in the consideration of laws of robotics' introduced in the 40's by Isaac Asimov, scientist and science fiction author:

Law Zero: A robot may not injure humanity, or, through inaction, allow humanity to come to harm.

Law One: A robot may not injure a human being or through inaction allow a human being to come to harm.

Law Two: A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law.

Law Three: A robot must protect its own existence, as long as such protection does not conflict with the First or Second Laws.

Conclusion

At the beginning of the project, as we head no idea about the complexity of the work, we specified our problem in a very extensive way, foreseeing many methods to implement, but slowly we realized that we where short of time and decided to reduce the project's scope to only some methods which seemed to be the most important. This is due to the challenging characteristic of our internship, which did not focus on a single part of the robotic process but almost on the whole of it (as described in figure 1 in the introduction). Indeed, within the SPQR team, each researcher was a specialist in one of the robot's architecture modules; vision, deliberation, or motion, whereas we had to understand and use all of them.

It was very funny and instructive to program a robot controller. However, as we explain in the debate chapter, we don't think that using a robot to amuse people, gives the solution to fundamental problems of our society. We told that we want to use robots to solve the solitude problem among elderly persons. This means that humans don't know how to live "well" together! And we believe that it would be better to learn humans to live together instead of learning to robots living with humans.

However we did a lot of job, therefore we would like to describe in the following paragraphs, our solution and some ideas on future work.

We implemented a Perception module, which allows the robot to understand its environment by extracting information from the images captured through its camera. The recognition of objects in the image is based on their colors and uses a color look-up or color table, which is manually created before running the robot, but is automatically updated during the robot's work.

We only implemented the object recognition using color edge-segmentation, while the automatic update of the color-table was implemented by SPQR. The points found from the edge-detection will be used to calculate the size and position of the objects in the image. This information forms the input of the Deliberative module, by which the robot performs the best action according to its goals.

Actions are then generated while executing plans, previously designed in Petri Nets formalism and stored into the robot's Memory stick just like the compiled program. The output of this module consists on a set of low-level commands given to the robot's actuators. We used the existing actuator commands implemented by SPQR and GT, and only defined the robot's head position while for line tracking (the head is not straight but a bit lowered).

If the preconditions are fulfilled (i.e. the user correctly places the lines, the newspaper and the robot, and respects the lighting condition requirements), the user can light AIBO if it is not already lighted, and to make it start performing the implemented task, he/she must click three times on its head. Then AIBO works on its own:

- It can recognize the shape and color of the lines (serving as landmarks), and follows them correctly (i.e. follows them in the right direction).
- As the position of the newspaper in the world is known, the robot stops following the line when it reaches the end of the white-blue line, and it starts searching a newspaper.
- Once AIBO has recognized the newspaper it goes close enough to catch it. But as it can't carry the object, it only raises its head and then turns around, to go back at the beginning of the path.

Missing capabilities

Fetch the newspaper. Before starting to implement, we experienced many ways of making AIBO bring back the newspaper, just like a real dog should do. The idea that is most likely to work, is equipping AIBO with a gadget, which is a kind of newspaper's container (figure 8.1). The container will be

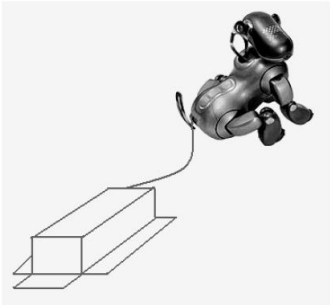


Fig. 8.1: AIBO equipped with a newspaper container

hanged to AIBO's tail and must be long and large enough to contain the newspaper. The bottom of the container (i.e. the part of the container that touches the floor) is not covered, and there is an entry for the newspaper. The wings at the sides of the container enable it not to reverse aside while AIBO walks and turns. Once AIBO has detected the lying newspaper, he should place itself at the same direction as this latter, and when he is close enough and in front of it (AIBO already does such actions when positioning itself in front of a line). Then the robot should walk a fixed number of steps, walking over the newspaper without crushing it, until the newspaper enters in the container. Then AIBO turns back. We have already manufactured the container and tested it. It works very well if it's used in a an environment without carpets.

Return at the initial position. AIBO should also remember to which room (identified by a bi-colored line) it has to go back. For the moment, when AIBO returns back, it follows lines until it reaches the last one. It should also be possible to make AIBO turn back to its initial room, i.e. to the first line it sees when it is asked to go and fetch the newspaper. In such a case, the user sends AIBO find the newspaper in one of the rooms, and waits for it at the same place.

Future work

Here over we described some missing methods in our project, hence, we think that if the project is extended the following capabilities and improvements should be brought.

Sound detection. As we already told, to make the robot fetch the newspaper, the user must click three times on the robot's head. The initial idea was to have AIBO starting the task when it hears a sound, like the bell, or a voice command from the user. This is an important ability for AIBO to have, especially if we include deaf people in the aimed users.
The person could make AIBO store the sounds of the bell or the phone, and choose which actions it must perform to notify each of these sounds. However, the idea would work if only vocal commands are considered and if the user always speaks loud and close to the robot.
But recording other sounds such as a bell, won't always work. Indeed if the robot hears the bell ringing and it is located at a different place then the place he was when the sound was recorded, the robot won't recognize the sound, and even if he is located at the right place, if there is noise it would hardly recognize the bell sound. Thus the idea is attractive, but a lot of work and experiments have to be done in this direction.

Obstacle detection and collision avoidance. As a domestic environment is generally equipped with many objects, it seems normal to make the robot avoid these objects, so that the user will not have to move the furniture to save the robot from collision. Obstacle detection and collision avoidance, is not necessary in our work since the robot mainly walks on lines that are located in a free space, but if the project is extended and AIBO is expected to perform other tasks (e.g. following a moving ball), this feature should be taken into consideration. However, this is also far from being easy. We have the choice between vision system detection and the distance sensing detection. Using vision means that the robot is supposed to know what is the color or shape of a potent obstacle, the walls and doors are the most interesting, hence walls and doors have different colors from one household to another. While using distance sensor, seems to work good for wall detection, as reported by experiences in [50].

Lighting changes adaptability. For the pixel-color association, we used a color-mapping table calibrated for environments with ideal light conditions, i.e. two sources of artificial light. If we start AIBO and only one source of light is available, the image segmentation process will not give good results (figure 8.2).

This means that we should have calibrated the color-mapping table according to this new lighting conditions, to have the image segmented properly.
But the user should not have to do such mappings. If he wants AIBO to work properly he must start it in a much lightened place before, and then light conditions can change and AIBO will adapt, but a lot of work is still to be done. Further works should concentrate on having the robot automatically calibrate its color-mapping table, based on a standard table. For example we could think of implementing a method, which tries several mappings until it reaches a maximum in the number of pixels that are classified in the image. Hence, this idea must be tested.

Looking back at this job, and wandering about what we should have done better, we believe that we should have asked to be given the mission of only implementing one of the modules (either Perception or Behavior), in order to have at least one module working properly. Indeed, the robot performs the most

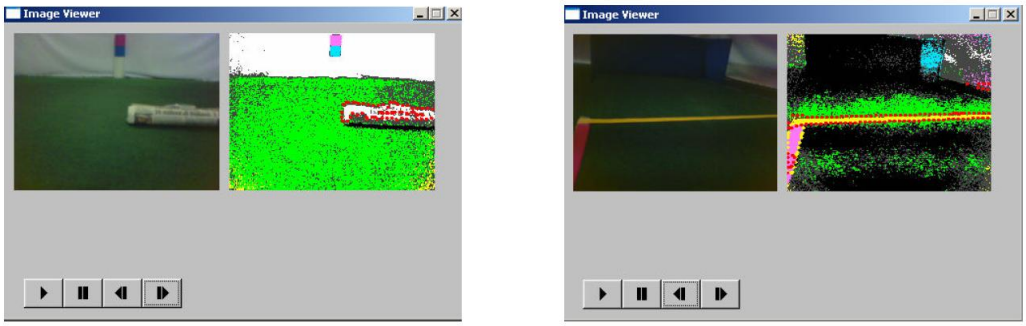


Fig. 8.2: (a) the right image shows the result of segmentation for the left image captured while two source of lights were available; (b) when only one source of light is available for the same color-mapping table the walls are not recognized!

important and significant tasks within good reaction times, but still some errors occur and because of the lack of time, the techniques that we used are very primitive and simple. Furthermore the project is not completely finished since we did not have the time to do experiences in a domestic environment, and some of the capabilities we had foreseen were not implemented. In the following paragraphs, we will talk about some capabilities that the robot should own but were not implemented, and we will also mention some suggestions for making the project even more useful and performing.

Bibliography

[1] *Behavior-Based Robotics*, chapter 9: Social Behavior. MIT Press, 1998.

[2] C. Angle and A. Gruber. Carebots: the future of home healthcare. *MIT Alumni Association*, 2006.

[3] D. H. Ballard. Generalizing the hough transform todetect arbitrary shapes. *Pattern Recognition, 13(2)*, 1981.

[4] J. C. Bauer. Service robots in health care: The evolution of mechanical solutions to human resource problems. *Bon Secours Heaalth System, Inc. Technology Early Warning System Ũ White Paper*, 2003.

[5] J. M. Beaulieu and M. Goldberg. Hierarchy in picture segmentation: a stepwise optimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:150–163, 1989.

[6] A. Belhdadi and A. Khellaf. A noise-filtering method using a local information measure. *IEEE Trans. Image Processing*, 6(6):879–882, June 1997.

[7] S. Bie and J. Persson. Behavior-based control of the ers-7 aibo robot. Master's thesis, Faculty of Science, Lund University, Sweden, 2004.

[8] D. R. K. Brownrigg. The weighted median filter. *Communications of the ACM*, 27(8):807–818, Augustus 1984.

[9] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[10] A.H. Vagnucci D.C.C. Wang and C.C. Li. Gradient inverse weighted smoothing scheme and the evaluation of its performance. *Computer Graphics and Image Processing*, 15:167–181, 1981.

[11] XABSL developer team. The extensible agent behavior specification language. `http://www2.informatik.hu-berlin.de/ki/XABSL/`, 2002-2006.

[12] E.C.Hildreth D.Marr. Theory of edge detection. *Proceedings of the royal society of London B2007*, 1980.

[13] WIKIPEDIA The Free Enciclopedia. Yuv. `http://en.wikipedia.org/wiki/YUV`, 2006.

[14] M. Lang et al. Noise reduction using an undecimated discrete wavelet transform. In *IEEE Signal Processing Letters*, 1995.

[15] V. Zharkova et al. Feature recognition in solar images. *Artificial Intelligence Review*, 23/3:209–266, 2005.

[16] Centre for Human Animal Bond. `http://www.vet.purdue.edu/chab/indcon.htm`.

[17] F. Giannone. Rappresentazione e traduzione di piani. Technical report, Facoltà di Ingegneria dell'Università degli Studi di Roma ŞLa SapienzaŤ, 2003-2004.

[18] H.Bakstein. A complete dlt-based camera calibration with a virtual 3d calibration object. Master's thesis, Faculty of Mathematics and Physics, Charles University, Prague, 1999.

[19] P. V. C. Hough. Method and means for recognizing complex patterns. Technical report, U.S. Patent, 1962.

[20] L. Iocchi. Robust color segmentation through adaptive color distribution transformation. Technical report, Dipartimento di Informatica e Sistemistica Universit'a ŞLa SapienzaŤ, Rome, Italy, 2006.

[21] L. Iocchi. Robust color segmentation through adaptive color distribution transformation. Technical report, Dipartimento di Informatica e Sistemistica Universita ŞLa SapienzaŤ, Rome, Italy, 2006.

[22] I.Sobel. An isotropic 3x3 image gradient operator. *Machine Vision for Three-Dimensional Scenes*, pages 376–379, 1990.

[23] B. Walter J. Knutzon. Object recognition in image processing. `http://www.public.iastate.edu/~knutzonj/ee424projectMain.htm`, 2006.

[24] M. Jiang. Digital image processing. `http://iria.math.pku.edu.cn/~jiangm/courses/dip/html/dip.html`, 2003.

[25] J.M.S.Prewitt. Object enhancement and extraction. *Picture Processing and Psychopictorics*, 1970.

[26] M. Jüngel. Using layered color precision for a self-calibrating vision system. Technical report, Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany, July 4-5, 2004.

[27] F. Jossy. Robostaff: Some pharmacy and nursing tasks don't need the human touch. *healthcare Informatics online*, 2004.

[28] S. Kakuchi. Robot lovin. `http://www.pathfinder.com/asiaweek/magazine/life/0,8782,182326,00.html`, 2001.

[29] A. Kleiner and V. A. Ziparo. Robocuprescue - simulation league team rescuerobots freiburg (germany). Technical report, Institut für Informatik Foundations of AI Universität Freiburg, 2006.

[30] J. K. Mui K.S. Fu. A survey on image segmentation. *Pattern Recognition*, 1981.

[31] Sang Uk Lee, Seok Yoon Chung, and Rae Hong Park. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52:171–190, 1990.

[32] G.R. Leone. Estrazione e fusione di informazioni da sistemi di visione eterogenei. Master's thesis, Facolta di ingenieria dell'Universita degli studi di Roma "La Sapienza", 2001-2002.

[33] L.Marchetti. Localizzazione in ambiente robotico tramite l'uso di filtri partucellari. Master's thesis, Facolta di ingenieria dell'Universita degli studi di Roma "La Sapienza", 2004-2005.

[34] F. Macrì. Classificazione automatica dei colori e riconoscimento oggetti in ambiente robocup. Master's thesis, Facolta di Ingenieria deel'Università degli Studi di Roma "La Sapienza", Rome, Italy, 2005.

[35] D. Marshall. Region splitting. `http://www.cs.cf.ac.uk/Dave/`, 1994-1997.

[36] Bülent Sankur Mehmet Sezgin. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146Ũ165, January 2004.

[37] A. Mehnert and P. Jackway. An improved seeded region growing algorithm. *PRL*, 18:1065–1071, 1997.

[38] W. Nisticò. Segmentazione di immagini in tempo reale per robot tetrapodi. Master's thesis, Facoltà di Ingegneria dell'Università degli Studi di Roma ŞLa SapienzaŤ, 2001-2002.

[39] W. Nisticò and T. Röfer. Improving percept reliability in the sony four-legged robot league. Technical report, Institute for Robot Research (IRF), Universität Dortmund and Center for Computing Technology (TZI), FB 3, Universität Bremen, 2004.

[40] University of Texas at Austin. History. `http://www.robotics.utexas.edu/rrg`, 2006.

[41] A. Beck et al. P. Kahn, B. Friedman. Value sensitive design. *VSD*, 2005.

[42] C.A. Petri. Communication with automata. Technical report, New York: Griffiss Air Force Base, RADC-TR-65-377 vol. 1, Suppl. 1,, 1966.

[43] J. Piater. Visual feature learning. Master's thesis, University of Massachusetts, 2001.

[44] L. Bischof R. Adams. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-16::641–647, June 1994.

[45] A. Walker R. Fisher, S. Perkins and E. Wolfart. Thresholding. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/threshld.htm`, 2003.

[46] L.G. Roberts. Machine perception of 3-d solids. *Optical and Electro-Optical Information Processing*, page 159Ũ197, 1965.

[47] RoboCup. `http://www.robocup.org/overview/21.html`.

[48] RoboCup@Home. `http://www.ai.rug.nl/robocupathome/`.

[49] L. Rose. Nanto city, japan: Robotic companions. `http://www.forbes.com/2005/06/08/cx_lr_0608japan.html`, 2005.

[50] J. Roux. The pursuit ability of a robot 'pet'. Master's thesis, Mathematical and Computer Sciences of Heriot Watt University, Edinburgh, 2004-2005.

[51] D. Maiwand S. Skaff, G. Kantor and A. A. Rizzi. Inertial navigation and visual line following for a dynamical hexapod robot. *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vol. 2*, October 2003.

[52] W. Burgard S. Thrun, D. Fox and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.

[53] S.P.Q.R. + Sicilia. Italian robot team spqr+sicilia. Technical report, Dipartimento di Informatica e Sistemistica Universit‘a di Roma ŞLa Sapienza, Dipartimento di Ingegneria Informatica Universit‘a degli Studi di PalermoŤ, 2005.

[54] W. Skarbek and A. Koschan. Colour image segmentation — a survey. Technical report, Institute for Technical Informatics, Technical University of Berlin, October 1994.

[55] Black Ice Software. The hsi color space. `http://www.blackice.com/colorspaceHSI.htm`, 1999-2006.

[56] Black Ice Software. Hsv color space - color space conversion. `http://www.blackice.com/colorspaceHSV.htm`, 1999-2006.

[57] Black Ice Software. The rgb color space. `http://www.blackice.com/colorspaceRGB.htm`, 1999-2006.

[58] Sony. Aibo web site. `http://www.aibo.com/`, 2006.

[59] Open-R Sony. Aibo software environment development. `http://openr.aibo.com`, 2001-2002.

[60] OPEN-R SDK Sony Corporation. *Model Information for ERS-7*, 2004.

[61] SPQR-Legged. `http://www.dis.uniroma1.it/~spqr`.

[62] M. Sridharan and P. Stone. Autonomous color learning on a mobile robot. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

[63] N. Pears T. Heseltine and J. Austin. Evaluation of image pre-processing techniques for eigenface based face recognition. Research paper, Advanced Computer Architecture Group Department of Computer Science: The University of York, 2002.

[64] German Team. `http://www.germanteam.org`.

[65] German Team. Robocup 2005. Technical report, Center for Computing Technology,FB 3 Informatik,Universität Bremen;Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin; Fachgebiet Simulation und Systemoptimierung, FB 20 Informatik,Technische Universität Darmstadt; Institute for Robot Research, Information Technology Section, Dortmund University, 2005.

[66] S. Thrun. Robotic mapping: A survey. `citeseer.ist.psu.edu/thrun02robotic.html`, February 2002.

[67] M. Ulrich and C. Steger. Performance evaluation of 2d object recognition techniques. `citeseer.ist.psu.edu/ulrich02performance.html`.

[68] Sony AIBO web site. `http://support.sony-europe.com/aibo/1_1_3_aibo_story.asp?language=en`.

[69] V. A. Ziparo and L. Iocchi. Petri net plans. *ourth International Workshop on Modelling of Objects, Components, and Agents (MOCA'06)*, 2006.

Appendix A

Current controller

There are many preconditions that the user must respect to have this task performed correctly:

- Place the bi-colored lines with respect to the logic of directions (white-colored lines follow and are followed by white-colored lines as shown in fig.A.1), don't place them on stairs, and don't change their position while the robot is performing the task in question.

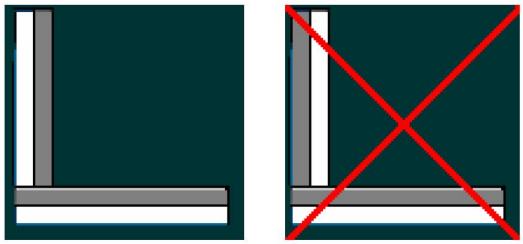


Fig. A.1: Placing the lines

- Place the newspaper close to the end of the blue-white, but not on the line (fig.A.2).

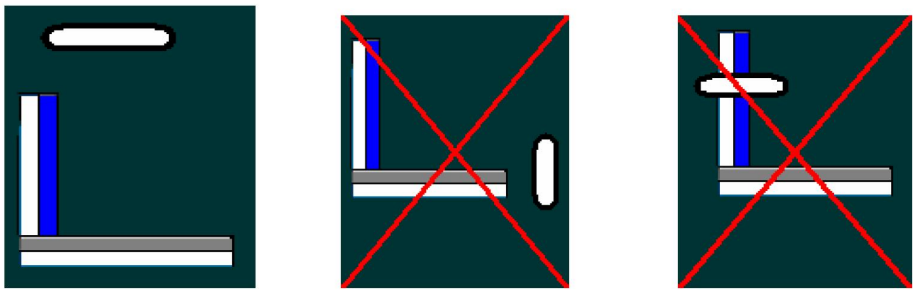


Fig. A.2: Placing the newspaper

- Make sure that AIBO is located not too far from a line, and from a position he could be able to detect the lines by moving its head right and left (fig.A.3).

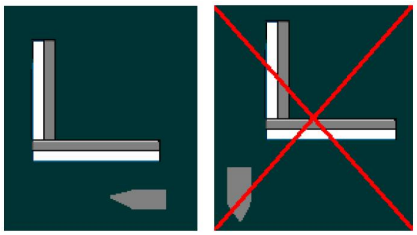


Fig. A.3: Placing the robot

- Insure that the lighting conditions are good, i.e. light as many lights as possible and avoid shadows.
- Don't place any obstacle on or around the lines or to the newspaper, otherwise the robot will collide.

The great number of preconditions is due to the lack of time for implementation, and they can be decreased if we implement some more features and improve others.

Appendix B

S.U.S.A.N: non-linear filter

This method is a powerful non-linear image processing method, which performs Edge and Corner Detection and Structure Preserving Noise Reduction as well, where the use of controlling parameters is much simpler and less arbitrary and therefore easier to automate than other edge detection algorithms. It is generally necessary if edges are to be localized to sub-pixel accuracy.

The concept of such an algorithm is the construction of a sub-region of points called USAN (Univalue Segment Assimilating Nucleus), having similar brightness to the central point called nucleus of a circular mask which dimensions are fixed (generally 37 pixels or a 3x3 mask).

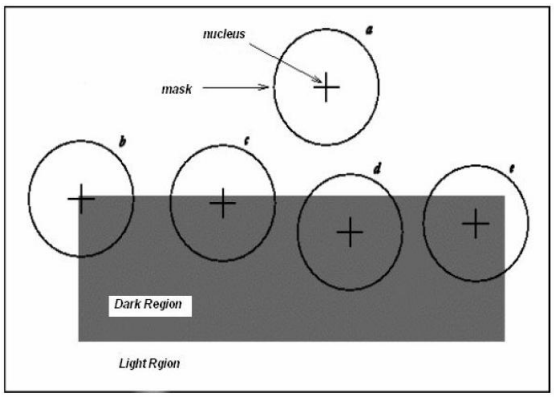


Fig. B.1: Application of SUSAN on a dark and white image

For each point, a correlation function is applied to the intensity difference between the nucleus and any other point in the mask. The USAN space is composed by all the points having a correlation result different form 0.

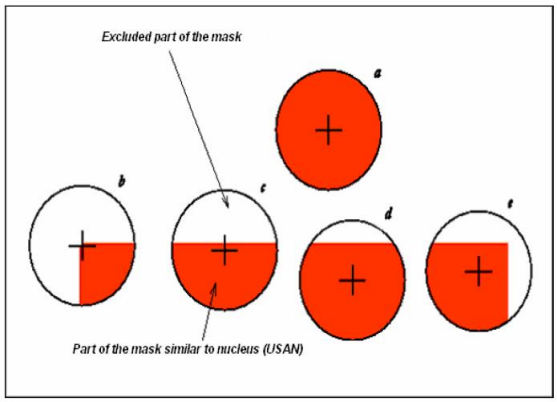


Fig. B.2: the red area is the USAN space associated to each nucleus

SUSAN used the following Gaussian correlation function:

$$c(p, p_0) = \exp -(\frac{I(p) - I(p_0)}{T})_2$$

Where:

- p is any pixel of the mask.
- p_0 is the nucleus (the central point).
- $I(p)$ is the brightness.
- T is the brightness threshold that controls the smoothing scale. For small values of T we obtain higher selectivity of points to be included in the USAN that means only points having a closer brightness value to the nucleus will be included. When T grows the USAN extends to the whole mask. Anyway T is not a crucial choice for correct filtering.
- c is the output of the comparison.

SUSAN smoothing

The SUSAN smoothing (Smoothing over USAN) method, smooths only similar pixels within a local region known, the USAN.
To explain this type of smoothing in a more intuitive way we use the following example. Suppose that we have a gray point in the transition zone between a white region and a black one. If this point is close enough to one of these regions, it will be filtered as a point belonging to that region. But if this point is an isolated one, not close enough to one of the regions, a median operator will then be applied to decide to which region it shall be associated. Clearly, we can see that geometric properties are not only

preserved but are also improved.

SUSAN edge detection

As seen in the fig.B.2, the USAN area is at a maximum when the nucleus lies in a flat region of the image surface. It falls to half of this maximum very near a straight edge and falls even further when inside a corner.
The main steps of Edge detection with the SUSAN method are described here:

Step1: A circular mask of for example 37 pixels or a 3x3 mask, is placed at each point in the image and, for each point, the brightness of each pixel within the mask is compared with that of the nucleus (the center point).

Step 2: If the difference of the brightness values is less than a threshold T, then the output of the comparison is 1 else its 0. The comparison is done for each pixel within the mask and is summed up. This total N is the number of pixels in the USAN: the USAN's area. T determines the maximum contrast of features which will be detected, and also the minimum amount of noise which will be ignored.

Step 3: N is compared with a fixed threshold g. This value is calculated from analysis of the expectation value of the response in the presence of noise only. The use of g should not result in incorrect dismissal of correct edges. With the SUSAN principle, false positive problems due to noisy images are solved.

Appendix C

Working context

Robots in society

The notion of "robot" first appeared in the 1923, when Karl Capek, a well-known Czech science-fiction writer at that time, wrote a futuristic thriller entitled "Rossum's Universal Robots", where robots control human society. Thus the word "robot", has a Czech origin and stands for forced labour or serf [40]. Indeed the first robots that were created, where applied in the industries, to accomplish jobs that were hot, heavy, dangerous or repetitive. Nowadays robots are still used in industrial applications, but they also serve new purposes, where they are no more used as serfs, but as interesting and wonderful tools in education, agriculture, rescue operations or space exploration [40]. Hence recently, robots have been attributed a new critical mission aiming to bring a solution to the modern society new crises; caring for our aging population. And it is in this context that our work was developed. This problem motivated the creation of the RoboCare Project at ISTC, which is the research group which asked our contribution.

Research on robotics

We were invited to work in the ISTC Institute (Institute of Cognitive Science and Technology) (www.istc.cnr.it), which belongs to the institutes' network of CNR (National Research Council) in Italy. The institute, located in Rome (Italy), was created in 2001, and is the result of a fusion of various institutions such as: the former Institute of Psychology, the former Institute of Phonetics and Dialectology in Padova and some groups from Biomedical Technologies in Rome, LADSEB in Padova and from the Solid State Electronics group in Rome. The Institute is involved in research, enhancement, technological transfer and training activities in scientific areas such as, artificial intelligence, artificial life, artificial societies; cognitive technologies, neural networks, autonomous robotics; social cognition; decision-making, cooperation technologies and many others (www.istc.cnr.it).

RoboCare (http://robocare.istc.cnr.it) is focused on the issue of "independence" and "aging at home" of the aging European population, which needs support tools to be developed to ensure that elderly and partially impaired people can continue to enjoy an independent lifestyle. They use autonomous robots

and distributed computing technologies for the implementation of a user service generating system in a closed environment such as a health-care institution or a domestic environment. Research topics tackled in the RoboCare Project are: the production of different robotic platforms, the supervision of activities in complex environments, the problem of human-robot interaction and in general of technology acceptance and usability by real users, Multi-agent systems, Autonomous robotics, Robot coordination, Planning, Scheduling and Execution, Monitoring Ě As the aim of the project is to provide services for the elderly, using a dog-like robot as an entertainment pet seemed to reduce the acceptance issue. Of course, Sony already provides AIBO with pet-like capacities, like playing with the ball, recognizing the owner... but the task we were asked to implement has a larger scope. Indeed AIBO should be able to autonomously locate itself in an indoor lighted environment. Furthermore, as the RoboCare project was planning to participate to a new challenging competition named Robocup@Home [48], our new task could have been presented in the challenge.

RoboCup@Home is a competition that focuses on real-world applications and human-machine interaction with autonomous robots. The aim is to foster the development of useful robotic applications that can assist humans in their daily life. This competition is accessible to anyone with an autonomous robot, be it self-constructed or a pre-manufactured, and it is application oriented, which means that participants are not judged on the technical properties of their robot, but on the software skills it develops according to its technical capabilities and limits [48]. The competition consists on a set of pre-defined tests and an open challenge test where freely chosen abilities of the robot can be shown. The tests are performed in a real world scenario, initially a home-looking environment with a living room and a kitchen, but it will evolve and tend to operate in common areas of daily life such as parks, shops, streets or other public places. In order to take part to the tests, the tasks presented must follow the spirit of the competition, i.e. include human machine interaction, be socially relevant and interesting to watch, be application oriented, be scientifically challenging but simple, be easy to set up, low in costs and short, and have self-explaining rules [48]. All these criteria are respected in our work. However members of the RoboCare staff, didn't have any experience with the AIBO robot, thus could not entirely help us in our work. That's why we went to work at the Faculty of Engineering of the University of Rome "La Sapienza", where we were supervised by the S.P.Q.R team [61]. S.P.Q.R. (Soccer Player Quadruped Robots) is a group of researchers from the Faculty of Engineering that is involved in the RoboCup [47] competitions since 1998.

The RoboCup Competition is an international research and education initiative, which aims to promote Artificial Intelligence and Robotics research, by providing a standard problem where a wide range of technologies can be examined and integrated. RoboCup initially started with the soccer competition in 1997, with the ultimate goal stated by the RoboCup Federation as follows: "By 2050, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official FIFA rules, against the winner of the most recent World Cup of Human Soccer."[47] Since 2000 RoboCup competitions have extended by creating the RoboCupRescue competition, to foster research and development on Robotic rescue, and the RoboCupJunior initiative which sponsors local, regional and international robotic events for young students. S.P.Q.R has successfully participated to the RoboCup Soccer competitions since its beginnings, but since the GermanTeam [64] published its source code and documentation in 2004, many other teams in the world, SPQR included, began to use them as bases for their work.

The GermanTeam is the German national robotic soccer team participating in the Four-Legged League of RoboCup, result of the cooperation of several German universities; the Humboldt-Universität zu Berlin, the Universität Bremen, and the Technische Universität Darmstadt [64].