GENERIC OBJECT DETECTION FOR AUTONOMOUS ROBOTS

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Carlos Gallardo García
i dirigit per
Sergio Escalera i Petia Radeva
Bellaterra, **29** de **Enero** de **2007**

Abstract

Artificial Intelligence for Autonomous robots is a growing line of interest for researchers. Computer Vision techniques for Object Detection and Recognition are widely applied in this domain to increase to robots interaction. In this paper, we present a generic Object Detection System for autonomous robots simulated in the Aibo robot of Sony. The detection process is done by means of cascades of classifiers using the Gentle Addaboost with Haar-like features estimated over the integral image. The system has been tested in different environments to test the Aibo interaction and behaviors. The system works at real-time and has been applied to other real applications, such as the traffic sign detection problem. The system has been exposed in different social events with great interest.

Resumen

La inteligencia artificial de robots autónomos es un campo de amplio interés para los investigadores. Las técnicas de visión por computador para la detección y reconocimiento de objetos se aplican ampliamente en este dominio para conseguir mayor interacción robótica. En este artículo presentamos un sistema genérico de detección de objetos para robots autónomos, el cual hemos integrado en el Aibo de Sony llamado Aibo. El proceso de detección se realiza mediante cascadas de clasificadores utilizando Gentle Adaboost con características Haar-like calculadas sobre la imagen integral. El sistema ha sido testeado en diferentes entornos para probar la interacción y el comportamiento del robot.Este sistema trabaja a tiempo real y ha sido utilizado en otras aplicaciones reales tales como el problema de detección de señales de tráfico. El sistema ha sido expuesto en diversos acontecimientos sociales despertando gran interés.

Resum

La intel·ligència artificial de robots autònoms és un camp de gran interés per als investigadors. Les tècniques de visió per computador per la detecció i reconeixement d'objectes s'apliquen àmpliament en aquest domini per assolir una major interacció dels robots. En aquest article presentem un sistema genèric de detecció d'objectes per a robots autònoms, el qual hem simulat amb un robot de Sony anomenat Aibo. El procés de detecció es realitza mitjançant cascades de clasificadors utilitzant Gentle Adaboost amb característiques Haar-like estimades sobre la imatge integral. El sistema ha estat testejat a diferents entorns per provar la interacció i el comportament del robot.Aquest sistema treballa a temps real i ha estat utilitzat en altres aplicacions reals tals com el problema de la detecció de senyals de trànsit. El sistema ha estat exposat en diversos esdeveniments socials despertant gran interès.

Index

Abstract **1**
Keywords **1**
1. Introduction **1**
2. Aibo Robot **4**
2.1 What is the Aibo entertainment robot? 4
2.2 Autonomous activities of the Aibo robot 4
2.3 The Aibo colour camera 4
3. Object Detection **5**
3.1 Adaboost detection 5
3.2 Haar-like features 6
3.3 Detectors cascade 6
4. System **9**
4.1 Generating the training set 9
4.2 Training classifiers 9
4.3 Object detection 10
4.4 Object classification 10
4.5 Aibo Behavior 10
5. Results **13**
5.1 Aibo detection 13
5.2 Traffic sign detection 14
5.3 Discussion 16
6. Conclusions **16**
7. Acknowledgments **17**
Referentes **17**
8. Appendix **19**
8.1 Appendix A Manual de XABSL 19
8.2 Appendix B The AIBO® Entertainment Robot ERS-7M3 36
8.3 Appendix C Computer Vision 40
8.4 Appendix D Mobile mapping adquisition 46
8.5 Appendix E Artificial Intelligence History 49
8.6 Appendix F Apropat a la Ciència 59

Generic Object Detection for Autonomous Robots

Carlos Gallardo García

Informatics, ETSE, UAB, Campus UAB, 08193 Bellaterra, Spain

Resum

Artificial Intelligence for Autonomous robots is a growing line of interest for researches. Computer Vision techniques for Object Detection and Recognition are widely applied in this domain to increase to robots interaction. In this paper, we present a generic Object Detection System for autonomous robots simulated in the Aibo robot of Sony. The detection process is done by means of cascades of classifiers using the Gentle Adaboost with Haar-like features estimated over the integral image. The system has been tested in different environments to test the Aibo interaction and behaviors. The system works at real-time and has been applied to other real applications, such as the traffic sign detection problem. The system has been exposed in different social events with great interest.

Key words: Autonomous Robots, Aibo, Artificial Intelligence, Computer Vision, Object Detection, Adaboost.
PACS:

1 Introduction

From the beginning of the civilization, humans has tried to delegate hard tasks to other people instead of do them by ourself. Without any doubt, the human ambition has increased practically at same time that the technology. Slowly, robots are replacing tasks that we have always done manually, such the industry assembly line.

On of the main problems for autonomous robots is its ability to interact with its environment. Computer Vision is a wide line of research that treats to deal the problem of visual perception. Object detection and recognition is a pervasive activity in our lives. We are constantly looking for, detecting and recognizing objects: people, streets, buildings, tables, chairs, desks, sofas, beds,

automobiles, etc. It still remains a mystery how we perceive objects so accurately and with so little apparent effort. In order to recognize an object, one must know, at least, something about the object. The central problem is how to deal with a huge amount of variation in visual appearance. That is, how we can obtain a universal representation of an object that is able to cope with both, the variation within the object and with the diversity of visual imagery that exist in the world.

In most robot applications, a robot continuously searches for objects in images acquired by a camera. In object-based video annotation, video frames are automatically labelled with symbolic descriptions which may be connected to the presence of certain objects in the sequence. Recognition may be also useful to catalogue searching in art, trademark and other commercial applications. Recently, web-based systems have been developed, searching on the internet for images showing desired objects.

The Aibo robot from Sony (fig.1) is a perfect tool to implement and test artificial intelligent techniques in robotics. The AIBO robot combines a body (hardware) and mind (the Aibo Mind 3 software) that allow it to move, think, and display the lifelike attribute of emotion, instinct, learning and growth. It establishes communication with people by displaying emotions, and assumes various behaviors (autonomous actions) based on information which it gathers from its environment. The Aibo robot is not only a robot, but an autonomous robot with the ability to complement your live. While living with you, the behavior of the Aibo robot patterns develops as it learns and grows. Also, it lets you to implement new complex behaviors depending on the environment. So it is the best tool to try and test a signs recognition system in real environments. There is no other so developed technology embedded in a simple but intelligent robot.

Figura 1. Sony AIBO robot

Object recognition process basically is composed by three steps: detection of a region of interest (ROI), model matching, and classification. Each of these steps can be done by a great different number of algorithms. Depending on the object we want to recognize, different techniques offer different performance depending on the domain. Adaboost [1] has been at last years one of the most used technique for object detection, feature selection, and object classification. Usually, the problem of object recognition (e.g. person identification) needs a previous addressing the category detection (e.g. face location). According to the way objects are described, three main families of approaches can be considered [2]: part-based, patch-based and region-based methods. Part-based approaches considered that an object is defined as a specific spatial arrangement of the object parts. An unsupervised statistical learning of constellation of parts and spatial relations is used in [3] In [4] and [5] a representation integrating Boosting with constellations of contextual descriptors is defined, where the feature vector includes the bins that correspond to the different positions of the correlograms determining the object properties. Patch-based methods classify each rectangular image region of a fixed aspect ratio (shape) at multiple sizes, as object (or parts of the target object) or background. In [6] objects are described by the best features obtained using masks and normalized cross-correlation. Finally, region-based algorithms segment regions of the image from the background and describe them by a set of features that provide texture and shape information. In [4], the selection of feature points is based on image contour points. Model matching normally is adapted depending on the domain we are working on, and finally, object classification involves a lot of techniques to solve the problem of discriminability between different types of objects (classes).

In this paper, we present a generic object detection system applied to autonomous robots. In particular, we use the Aibo Robot of Sony to test the techniques. The generic detection system is based on learning a set of object categories by means of a cascade of classifiers. The classifier used is the Gentle Adaboost with decision stumps and the Haar-like features estimated on the integral images. As we show, this technique is very suitable for real-time detection problems and quit robust to object variations in the scene. Besides, we generate a simulation environment in which an interface shows to robot vision and object detection. The robot interacts depending on the presence of objects in the scene and shows different behaviors. We exposed the system is social and scientific events with great success and interest. Besides, the system was tested in a real detection problem: the detection of traffic signs.

The paper is organized as follows: section 2 comments the Aibo robot used to simulate the system. Section 3 comments the detection methodology applied in the present architecture, and section 4 explains the whole system. Finally, experimental results are shown in section 5 and section 6 concludes the paper.

2 Aibo Robot

2.1 ***What is the AIBO® Entertainment Robot?***

The AIBO robot is the name which Sony® has given to its family of entertainment robots, robots that are designed with the goal of presenting a vision for a new type of lifestyle in which human beings derive enjoyment from mutual existence with robotic creatures. The name itself is a play on the words "artificial intelligence"(AI) and "robot", or a robot with eyes. In its home country, Japan, the word "AIBO"also means "partner"or çompanion". [1] See fig.29

2.2 ***Autonomous activities of the AIBO robot***

The AIBO robot combines a body (hardware) and mind (the AIBO MIND 3 software) that allow it to move, think, and display the lifelike attributes of emotion, instinct, learning, and growth. It establishes communication with people by displaying emotions, and assumes various behaviors (autonomous actions) based on information which it gathers from its environment, the AIBO robot's behavioral patterns will develop as it learns and grows,the AIBO robot undergoes changes in spirit that display themselves in the form of emotional expression. The AIBO robot possesses the following five basic instincts: Love instinct,Search instinct,Movement instinct,Recharge instinct,Sleep instinct.It really seems to be an electronic life.

2.3 ***The AIBO® COLOR CAMERA***

About the pictures o Pictures are stored on the "Memory Stick"media in JPEG format. o The picture resolution is 416 x 320 pixels. o Depending on lighting conditions at the time the picture is taken, flicker (horizontal stripes) may appear in pictures, or pictures may have red or blue hues. o Fast movement may result in distortion of pictures. See fig.2

[1] For the latest information on the AIBO robot, visit the following Web site: http://www.aibo.com

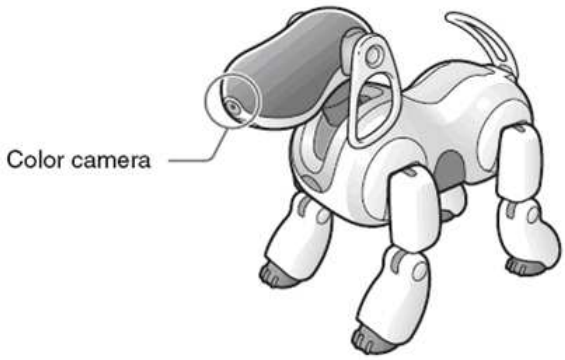


Figura 2. Sony AIBO Robot

3 Object Detection

In this chapter, we explain the techniques used to solve the object detection step.

3.1 ***Adabost Detection***

Gentle Adaboost (fig.3) is a modified version of the Real AdaBoost algorithm, using Newton stepping rather than exact optimization at each step [13].Gentle Adaboost is the most frequently used variants of Adaboost and improves the other ones.

The conventional AdaBoost procedure can be easily interpreted as a greedy feature selection process.Consider the general problem of boosting, in which a large set of classification functions are combined using a weighted majority vote. The challenge is to associate a large weight with each good classification function and a smaller weight with poor functions. AdaBoost is an aggressive mechanism for selecting a small set of good classification functions which nevertheless have significant variety. Drawing an analogy between weak classifiers and features, AdaBoost is an effective procedure for searching out a small number of good "features"which nevertheless have significant variety. One practical method for completing this analogy is to restrict the weak learner to the set of classification functions each of which depend on a single feature. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples.[12]

In order to compute these features very rapidly at many scales we introduce the integral image representation for images (the integral image is very similar to the summed area table used in computer graphics [10] for texture mapping).

The integral image can be computed from an image using a few operations per pixel. Once computed, any one of these Harr-like features can be computed at any scale or location in constant time.

Within any image sub-window the total number of Harr-like features is very large, far larger than the number of pixels. In order to ensure fast classification, the learning process must exclude a large majority of the available features, and focus on a small set of critical features. Motivated by the work of Tieu and Viola, feature selection is achieved through a simple modification of the AdaBoost procedure: the weak learner is constrained so that each weak classifier returned can depend on only a single feature. As a result each stage of the boosting process, which selects a new weak classifier, can be viewed as a feature selection process. AdaBoost provides an effective learning algorithm and strong bounds on generalization performance [11].

The speed of the detector by focussing attention on promising regions of the image increases combining successively more complex classifiers in a cascade structure.The notion behind focus of attention approaches is that it is often possible to rapidly determine where in an image an object might occur. More complex processing is reserved only for these promising regions. The key measure of such an approach is the "false negative"rate of the attentional process. It must be the case that all, or almost all, object instances are selected by the attentional filter. Now, we discuss the Haar-like features and the cascade of classifiers structure.

Gentle AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weights w_i.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$
   (c) Update $w_i \leftarrow w_i e^{-y_i f_m(x_i)}$ and renormalize.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$

Figura 3. Gentle Adaboost algorithm

3.2 ***Haar-like features***

In order to obtain a robust detector, instead of using directly the image pixels, we use Haar-like features, which are differences between the sum of all the pixels in some contiguous rectangular regions of the image. This type of features are robust in front of the noise and illumination changes.

Our object detection procedure classifies images based on the value of simple features. There are many motivations for using features rather than the pixels directly. The most common reason is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. For this system there is also a second critical motivation for features: the feature-based system operates much faster than a pixel-based system.

In [7] Lienhart and Maydt extended the Haar-like features set used by Viola and Jones, adding the rotated versions of each feature type fig.4. All these features can be calculated using the integral image or SAT1 and the 45o rotated integral image or RSAT2. Both auxiliary images can be calculated using only one pass from left to right and top to bottom over all pixels. In the SAT image, each pixel SAT(x, y) contains the sum of all pixels of the upright rectangle ranging from the top-left corner to the bottom-right corner at (x, y) fig.5. The RSAT image is defined as the sum of the pixels of a 45o rotated rectangle with the bottom most corner at (x, y) and extending upwards till the boundaries of the image fig.5. Given a training window size, our feature set will be composed by all the possible features that we can put on this window changing the size and position.

Using the integral image any rectangular sum can be computed in four array references fig.5. Clearly the difference between two rectangular sums can be computed in eight references. Since the two rectangle features defined above involve adjacent rectangular sums they can be computed in six array references, eight in the case of the three-rectangle features, and nine for four-rectangle features. One alternative motivation for the integral image comes from the "boxlets" work of Simard, et al. [9]. The authors point out that in the case of linear operations,(e.g.fog) any invertible linear operation can be applied to f or g if its inverse is applied to the result.

3.3 ***Detectors Cascade***

The most important contribution of Viola and Jones work was the definition of the Attentional cascade. It is a degenerated decision tree where at each stage a detector is trained to detect almost all objects of interested while rejecting a certain fraction of the non-objects patterns fig.6.

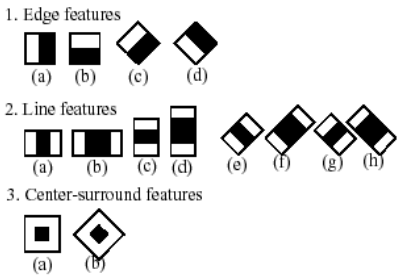


Figura 4. Extended set of Haar-like features. The white rectangle corresponds to the positive region and the black one to the negative.

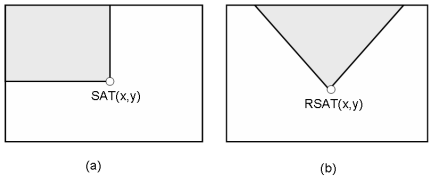


Figura 5. Definition of (a) Summed Area Table (SAT) and (b) Rotated Summed Area Table (RSAT).

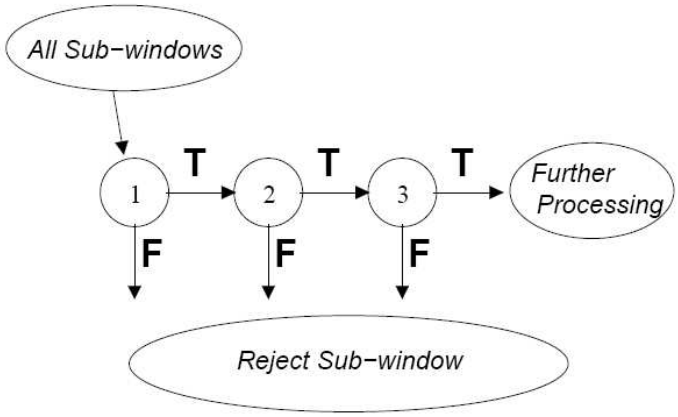


Figura 6. The attentionat cascade

Detection systems must accomplish hard restrictions both, on hit and false alarm ratios. If we train a simply detector with this restrictions, the number of hypotheses that the boosting method must combine to obtain the committee is enormous. Using the cascade structure, the false alarm ratio restriction is shared along false alarm of the detectors cascade will be FORMULA , where f is the false alarm ratio fixed for each stage of the cascade and n is the number of stages. The same can be applied to the hit ratio.

Each stage analyzes only the objects accepted by the previous stages, and thus, the non-objects are analyzed only until they are rejected by a detector.

The number of applied classifiers is reduced exponentially due to the cascade architecture. We use GentleAdaboost to learn each stage of the cascade, and change the rejected objects by other non-objects that the previous trained stages classify by correct objects.

4 System

To explain the system architecture of the detection process see the fig.10. Our propose is to detect a set of objects and test the system in the Sony Aibo robot. The detection steps are: generate a training set of positive samples, train a classifier for each group of objects, use the input frames from the Aibo robot to detect objects in the scene, use the process of [8] to classify by the object category, and finally control the Aibo behavior depending on the detected and classified object.

4.1 Generating the training set

To obtain a robust classifier, first we need to define a representative training set that contain the high variability of object appearance. For our implementation, we designed the set of objects of fig.7, categorized in two main groups, circular and triangular. The training set images were obtained by recording video sequences of the Aibo robot at different environments that contain the generated objects.

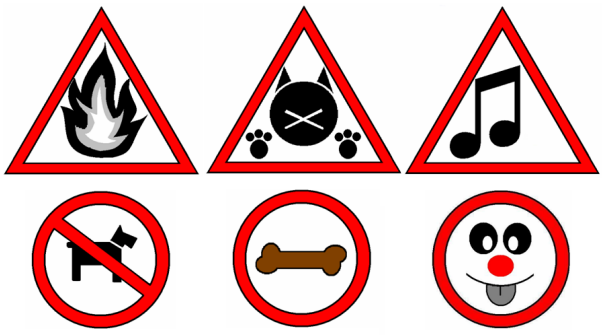


Figura 7. Designes

4.2 Training classifiers

We trained two detectors, one for the triangular objects, and the other for the circular ones. Each of these two groups were trained using a set of positive regions that contains samples of the objects, and also a negative set of samples selected randomly from images that do not contain the desired objects. Same examples of the set of positive and negative images used to train the cascades of classifiers are shown in fig.8.

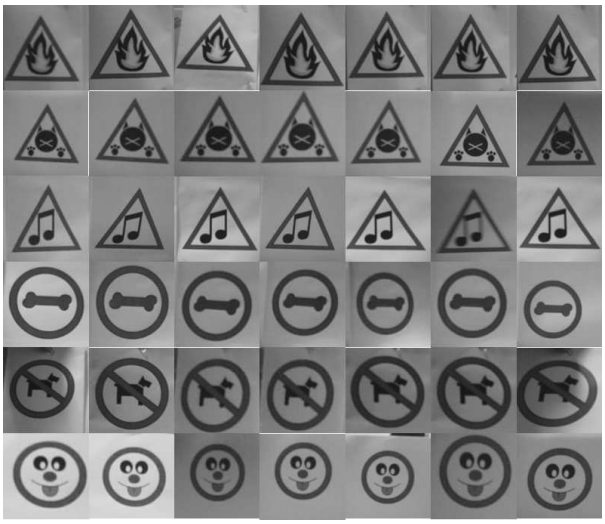


Figura 8. Positive regions

4.3 Object detection

Object detection step consists on applying a windowing strategy over the Aibo frames and test the trained cascades in order to classify each image region as object or non-object. In this way, robust trained cascades rejects regions that do not contain the object, and detect as positive the regions with instances of the object trained by the cascade.

4.4 Object classification

Once we have the detected regions, detected as circular or triangular group, next step is to classify the contain based in the approach of [8].

4.5 Aibo behavior

The final step consists in to generate physic behaviors to the Aibo robot depending on the detection and classification steps of the frames captured by the Aibo video camera.

The Aibo robot provides a set of implemented behaviors, such as dancing, singing, etc. We take use from this set of states to easily identify the output of our system. We defined a set of behaviors that Aibo runs depending of the detected and classified object of fig.9 by our system.To deal with this problem, we have updated the state machine integrated in the Aibo robot to define the execution of the robot depending on the results obtained by our system .Our machine of states are commented in the chapter of the results.

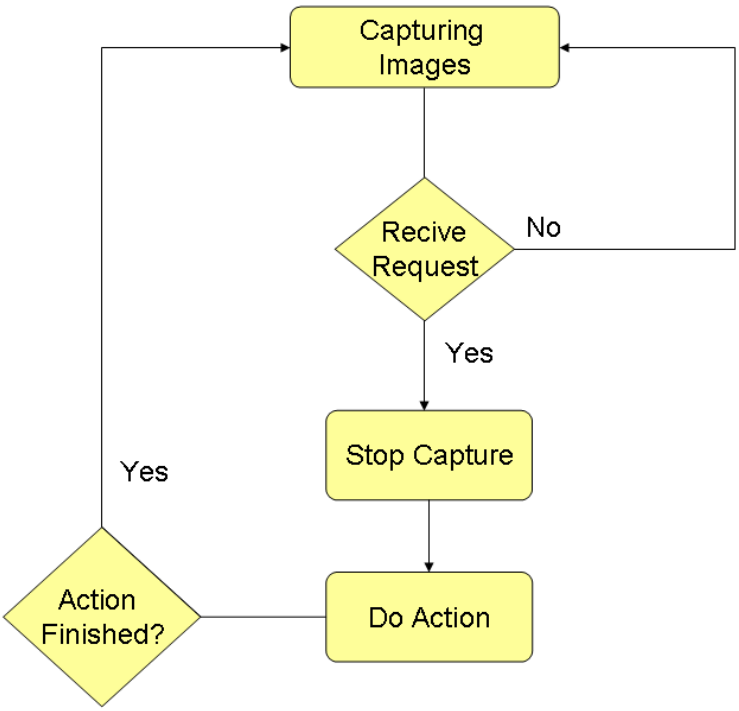


Figura 9. State Machine of the Aibo robot

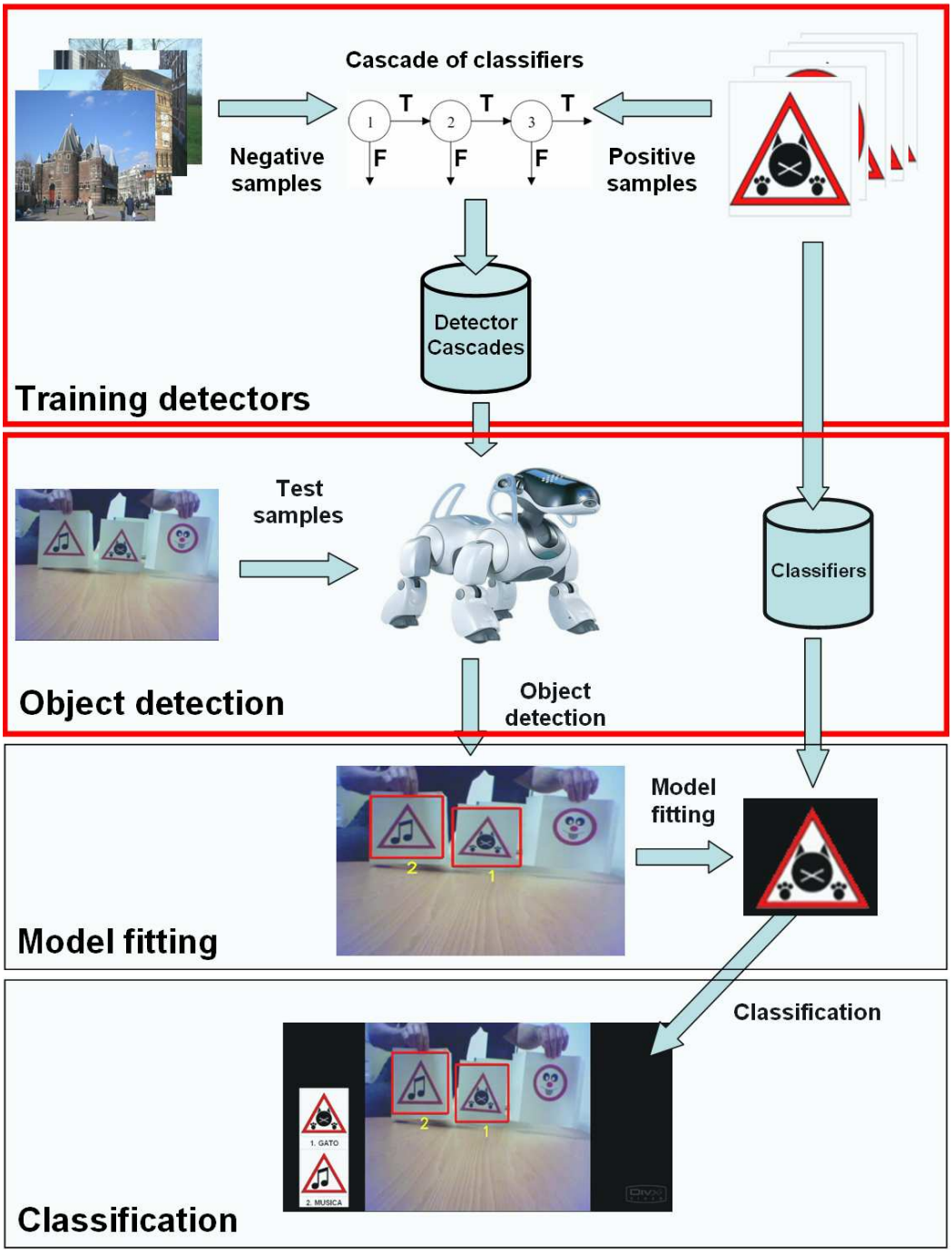


Figura 10. System Architecture

5 Results

Before showing the results, first, we comment the parameters of our system. The cascades of classifiers are trained using a set of 1000 positive samples of (32 x 32 pixel resolution) for each group, and 2000 negative samples at each stage of the cascade. In particular, we have experimentally tested that generating 6 cascade levels obtains good results. The classifier used to train the cascade is 50 runs of Gentle Adaboost with decision stumps using the Haar-like features estimated over the integral image.

The experiments are divided in: Aibo sign detection, traffic sing categorization, and exhibitions of the system.

5.1 ***Aibo detection:***

To test the system in the Aibo robot, we designed a C++ interface as shown in fig.13(a). Testing the trained cascades with the parameters commented before in a video set of total 2000 frames labeled manually we obtained the results shown in fig.14(a) for triangular and circular signs. The results are very high, being able to detect signs with low resolution (in 416 x 320 pixel frames) at different conditions, such as illumination changes, partial occlusions or rotation, obtaining results are upon 97% for both, circular and triangular signs.

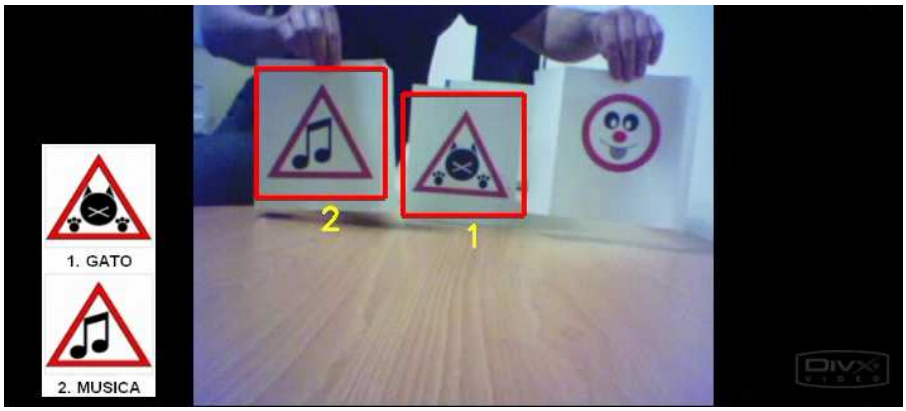


Figura 11. Our Test's interface

Our behavior architecture using the internal Aibo machine state is shown in fig.12. Different behaviors: dance, bark, Alarm, slad, eat, etc. are considered in the machine state.For more details of the Aibo machine state architecture see Apendix A.To illustrate the behaviors integration, in fig.13 the Aibo interface an example of the Aibo behavior in its environment is shown.

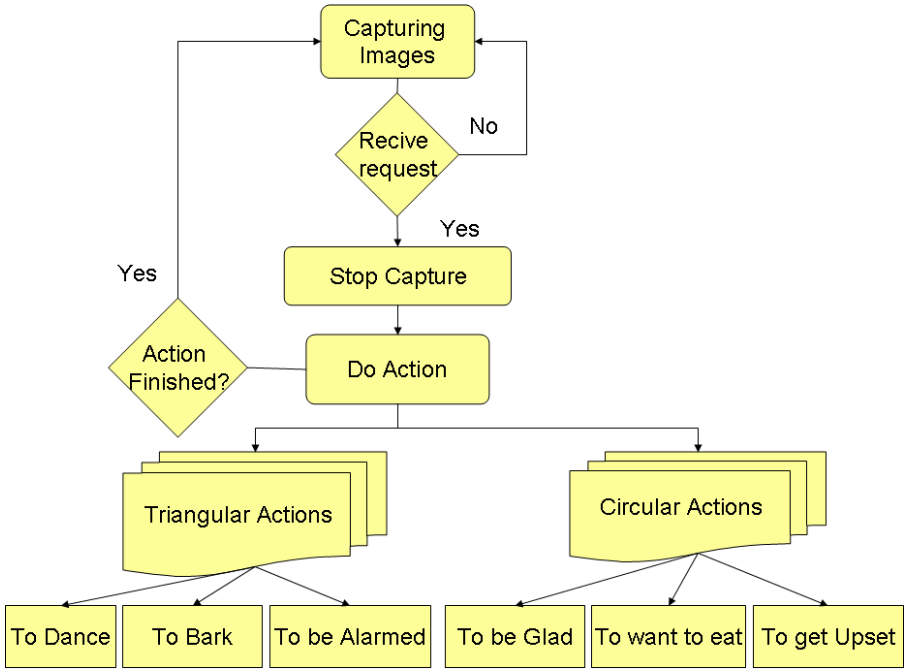
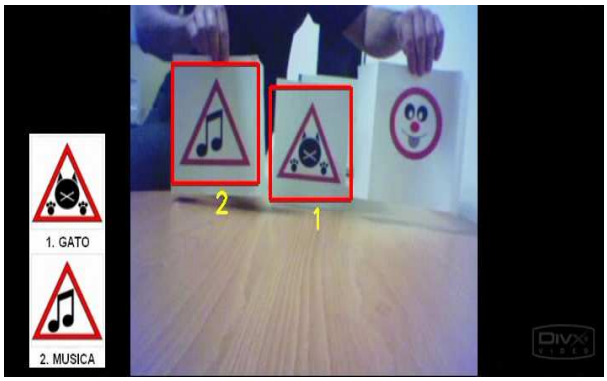


Figura 12. Aibo State Machine



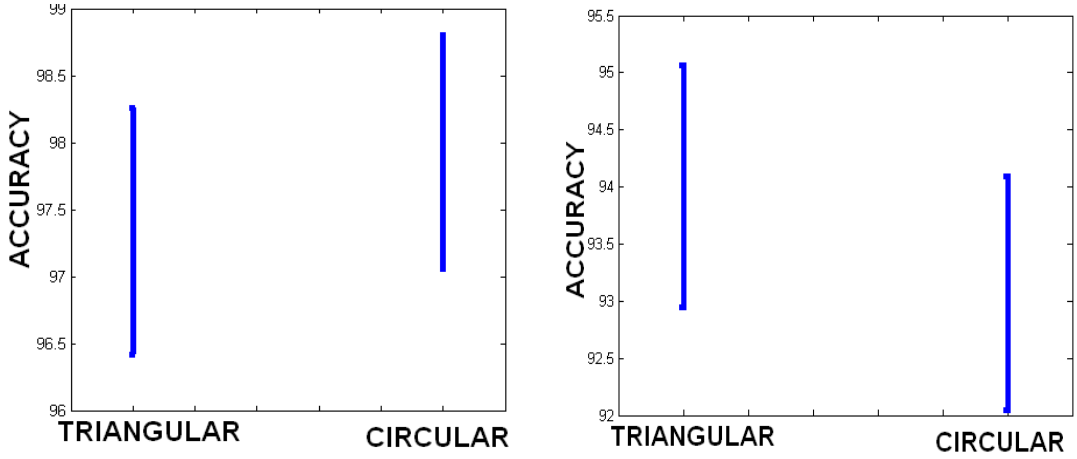
(a) Aibo Interface

(b) Aibo

Figura 13. System behavior

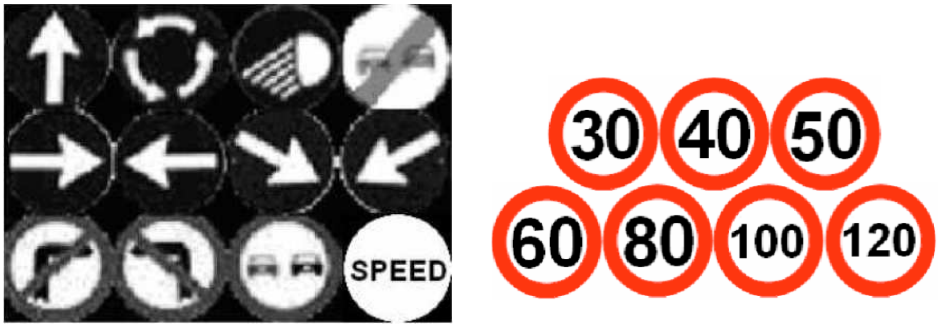
5.2 ***Traffic sign detection:***

We used the video frames from the Institute Cartographic of Catalonia [16] to test our detection system on other real applications. We used a wide set of classes (fig.15 and reffig:trianClasses) to train the circular and triangular cascades. The parameters of the cascade are the same than in the previous experiments.The detection results are shown in fig.14(b). In this case, we are also able to detect the traffic signs with results upon 90% for the two types of signs.



(a) (b)

Figura 14. (a) Aibo system detection results. (b) Traffic detection results



(a) (b)

Figura 15. (a) Circular classes. (b) Speed group.

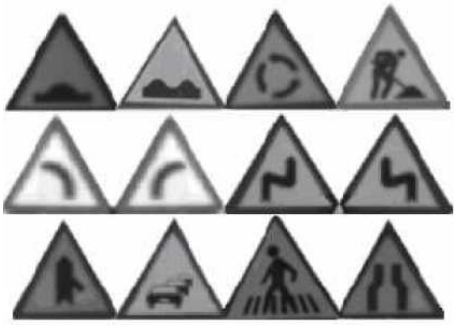
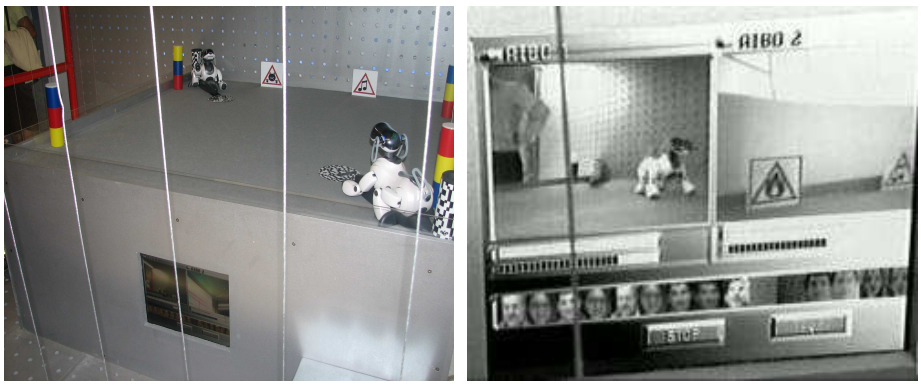


Figura 16. Triangulares classes



(a) Aibo environment (b) Aibo interface

Figura 17. Pictures of exhibition

5.3 Discussion

In this chapter, we comment different exhibitions of the present system on different social event.

5.4 ***"Apropat a la ciencia"***

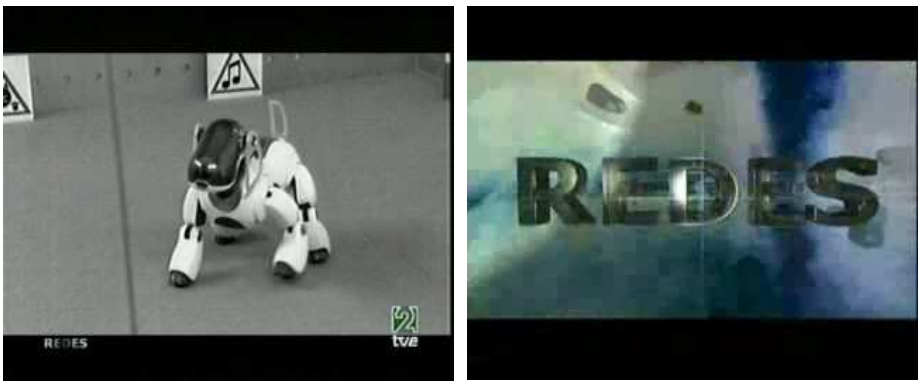
Robotics is a focus of attention for a high number of scientists. Now, so years after the birth of the Artificial Intelligence, we presented during the year 2006/2007 in the "Apropa't a la Ciència"event organized by the Generalitat de Catalunya, a simulation of the Aibo robot using our system. Different illustrations from the event are shown in figures fig. 17. For more details of the event see Appendix X or enter in website
http:www10.gencat.net/probert/catala/exposicio/ex14_ciencia.htm

5.4.1 ***"Redes"***

Beside, our work was emitted of a part of the documental "REDES"of TV2 from the little "programs emotions", in date of 7.1.2007 .Different images from the show are shown in figures fig.18.For more details enter in website
http:www.rtve.es/tve/b/redes/semanal/prg418/index.html

6 Conclusions

In this paper, we presented a real-time system for generic object detection applied to autonomous robots. The detection process is done by means of a



(a) Program logo (b) Aibo

Figura 18. Redes TV program

cascade of classifiers using Gentle Adaboost with the Haar-like features estimated over the integral image. The system was tested in the Aibo robot of Sony. Real simulations showed the high robustness of the robot on detecting objects in uncontrolled environments and adverse conditions with great success. Besides, the system was applied to a different real detection problem: the detection of traffic signs.

7 Acknowledges

I want to acknowledge Sergio, Petia, Bogdan, Santi, Jordi, Raúl, Ppl, and Xevi, between others, by their support and patience during this work. Also, thanks to my family!

Referències

[1] Yoav Freund and Robert E. Schapire. "A decision-theoretic generalization online learning and an application to boosting," Computational Learning Theory: Eurocolt '95, pages 23-37. Springer-Verlag, 1995.

[2] K. Murphy, A.Torralba and W.T.Freeman, Üsing the Forest to See the Trees: A Graphical Model Relating Features, Objects, and Scenes", Advances in NIPS, MIT Press, 2003.

[3] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning", CVPR, 2003.

[4] J. Amores, N. Sebe and P. Radeva, "Fast Spatial Pattern Discovery Integrating Boosting with Constellations of Contextual Descriptors", CVPR, 2005.

[5] S. Escalera, O. Pujol, and P. Radeva, "Boosted Landmarks of contextual descriptors and Forest-ECOC: A novel framework to detect and classify objects in cluttered scenes", International Conference on Pattern Recognition, Hong Kong, 2006.

[6] A. Torralba, K. Murphy and W. Freeman, "Sharing Visual Features for Multiclass and Multiview Object Detection", CVPR, pp. 762-769, vol. 2, 2004.

[7] R.Lienhart and J.Maydt, "An extended set of haar-like features for rapid object detection,"Proc. of the IEEE Conf. On Image Processing, pp. 155-162, 2002.

[8] R.Perez, "Generic Object classification for Autonomous Robots"thecnical reports UAB 2007

[9] Haffner, and Yann Le Cun. Boxlets: a fast convolution algorithm for signal processing and neural networks. In M. Kearns, S. Solla, and D. Cohn, editors, Advances in Neural Information Processing Systems, volume 11, pages 571–577, 1999.

[10] F. Crow. Summed-area tables for texture mapping. In Proceedings of SIGGRAPH, volume 18(3), pages 207–212, 1984.

[11] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In Proceedings of the Fourteenth International Conference on Machine Learning, 1997.

[12] Paul Viola and Michael Jones ,Robust Real-time Object Detection VANCOUVER, CANADA, JULY 13, 2001.

[13] ADDITIVE LOGISTIC REGRESSION: A STATISTICAL VIEW OF BOOSTING By Jerome Friedman,TrevorHastie and Robert Tibshirani Stanford University

[14] URL: http:www10.gencat.net/probert/catala/exposicio/ex14_ciencia.htm

[15] URL: http:www.bcn.es/ciencia2007/

[16] URL: http:www.icc.es

8 Apendix A Manual de XABSL

El llenguatge XABSL és un llenguatge especificat en l'equematica XML.

Els disseny del llenguatge XABSL ens assegura:

- interoperabilitat amb els editors i eines de XML.
- Que no necessitem cap altre validadror o compilador que no sigui l'estandard XSLT.
- Una escabilitat de les solucions dels comportmaents. Els agents dels comportaments són fàcils d'extendre.
- Gran velocitat de validació i compilació.

8.1 Modularitat

Els agents de comportament de XABSL són distribuits en diversos fitxers, això ens ajuda a gruardar una descripció sobre els agents grans, així com la possibilitat de treballar en paral·lel.

La figura 19 mostra els diferents fitxers que formen part dels agents de comportament del XABSL

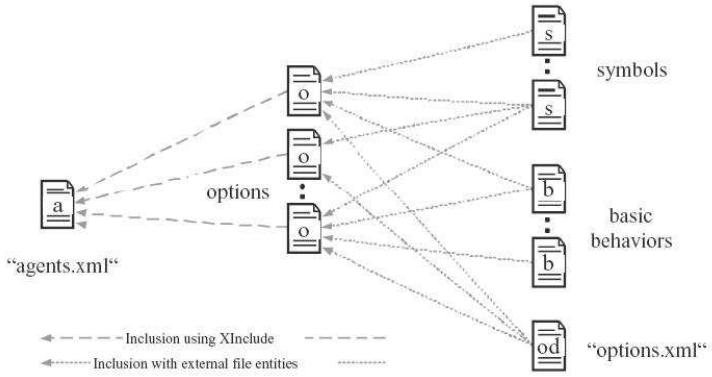


Figura 19. Els diferents tipus de fitxers en l'esquema de XABSL

- Els fitxers de símbols contenen les definicions dels símbols. Aquest símbols són utilitzats en els opcions.
- Els fitxers de comportaments bàsics conten els prototips per els comportaments així com també els seus paràmetres. Els comportaments són referenciats des dels estats que tenen un subseqüent comportament bàsic.
- El fitxers d'opcions contenen les opcions senzilles.

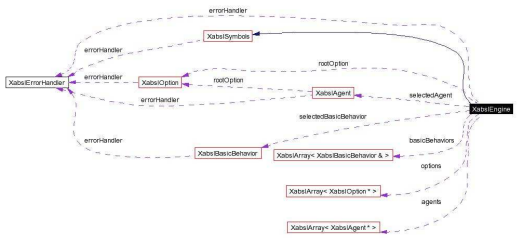


Figura 20. Relació entre les classes de XABSL

- El fitxer 'options.xml' defineix el prototipus de cada opció i els seus paràmetres. Aquests prototips són necessaris per comprovar en el fitxer d'opcions, quan una existeix una referència subseqüent.
- El fitxer 'agents.xml' inclou tots els fitxers d'opcions. Els agents i les opcions principals es defineixen aquí.

Així com els agents pot ser distribuït en molts fitxers, els esquemes també estàn modularitzats:

- xabsl-2.1.agent-collection.xsd Element principal del comportament XABSL. Definició de l'agent.
- xabsl-2.1.basic-behaviors.xsd Prototipus per el comportament bàsic.
- xabsl-2.1.expressions.xsd Arxiu per les expressions decimals i booleanes.
- xabsl-2.1.option-definitions.xsd Prototipus per les opcions.
- xabsl-2.1.option.xsd Definicí de l'opció.
- xabsl-2.1.parameter.xsd Parametres per les opcions, comporatments bàscis i altres funcions.
- xabsl-2.1.symbols.xsd Definició dels simbols.
- xinclude-1.0.xsd Esquema simplificat per l'estàndard XInclude.
- xmlbase.xsd Esquema simplificat pel XML estàndard.

La figura 20 ens mostra les relacions que tenen les diferents classes de XABSL.

8.2 Definició dels simbols

Tots els símbols que siguin utilitzats dins les opcions primer ha de ser definits en un fitxer apart. Per exemple el fitxer 'my-symbols.xml' pot ser així:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<symbols xmlns="http://www.ki.informatik.hu-berlin.de/XABSL2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.1
xabsl-2.1.symbols.xsd"
id="my-symbols" title="My Symbols"
description="My most used symbols">
<boolean-input-symbol name="something-wrong" description="a boolean symbol"/>
<decimal-input-symbol name="foo" description="a decimal symbol" measure="mm"/>
```

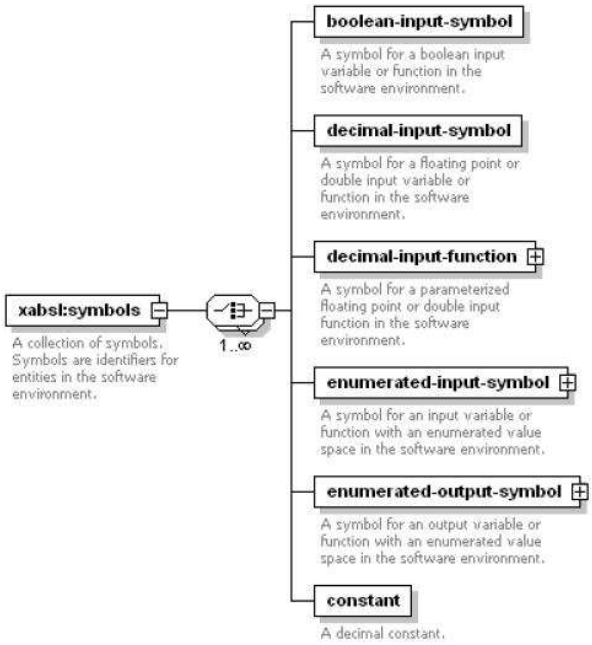


Figura 21. L'estructura del l'element símbol

```
<decimal-input-function name="abs" description="the absolute value of a number"measure="">
 <parameter name="abs.value" measure="" range="decimal" description="The value for that abs() is calculated"/>
</decimal-input-function>
<enumerated-input-symbol name="type-of-recognized-pet" description="Which pet was seen by the robot">
 <enum-element name="dog"/>
 <enum-element name="cat"/>
 <enum-element name="guinea-pig"/>
</enumerated-input-symbol>
<enumerated-output-symbol name="op-mode" description="The mode how fast the robot shall act">
 <enum-element name="op-mode.slow"/>
 <enum-element name="op-mode.fast"/>
 <enum-element name="op-mode.very-fast"/>
</enumerated-output-symbol>
<constant name="pi" description="The value of pi"
measure="rad" value="3.14"/>
...
</symbols>
```

Els atributs dels símbols són:

- 'id': Un identificador per la col·lecció de símbols. Ha de ser identic que el fitxer però sense la extensió.
- 'title': Un titul per la documentació.
- 'description': La descripció és requerida per la documentació.

La figura 21 mostra l'estructura dels símbols. Tenim 6 tipus diferents de símbols dins de l'element símbol.

- **boolean-input-symbol:** El símbol per a les funcions booleanes.
- **decimal-input-symbol:** El símbol per a les variables o funcions decimals.

- **decimal-input-function:** El prototip per a les funcions decimals parametritzades.
- **enumerated-input-symbol:** Un símbol per poder enumerar variables o funcions. Cada element enum ha de ser definit com un un símbol de fill del enum.
- **enumerated-output-symbol:** Aquest símbol pot ser utilitat per influir a l'agent mentres estem executatnt algun comportament bàsic. Així com el enumerated-input-symbol també té elements fill.
- **constant:** Defineix una costant decimal.

8.3 Prototipus dels comportaments bàsics

Per a cada comportament bàsic, hem de definir un prototipus. En el fitxer següent es mostra un exemple del que pot ser un comportament bàsci:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<basic-behaviors xmlns="http://www.ki.informatik.hu-berlin.de/XABSL2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.1
xabsl-2.1.basic-behaviors.xsd"
id="my-basic-behaviors" title="My Basic Behaviors" description="My common basic behaviors">

<basic-behavior name="move-to" description="Lets the agent move to a point">
 <parameter name="move-to.x" measure="mm" range="-1000..1000" description="X of destination position"/>
 <parameter name="move-to.y" measure="mm" range="-1000..1000" description="Y of destination position"/>
</basic-behavior>
<basic-behavior name="wait" description="The agent performs no action"/>
</basic-behaviors>
```

Els atributs per els basic-behaviors són:

- 'id': Un identificador per la col·lecció de comportaments bàsics. Ha de ser identic que el fitxer però sense la extensió.
- 'title': Un títol per la documentació.
- 'description': La descripció és requerida per la documentació.

8.4 Prototipus per a les opcions

Cada opció pot ser encapsulada dins el seu propi fitxer. Per tal de validar una opció símbol, hem de tenir un prototipus per a totes les altres opcions. Per tant, en cada agent de comportaments de XABSL he de tenir un fitxer anomenat 'options.xml'. Serà semblant a aquest.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<option-definitions
xmlns="http://www.ki.informatik.hu-berlin.de/XABSL2.1"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.1xabsl-2.1.option-definitions.xsd">
```

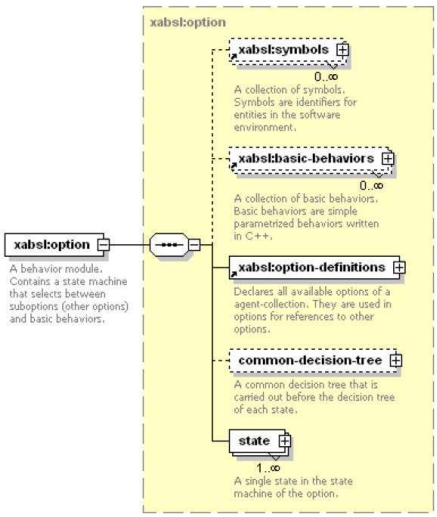


Figura 22. L'element opció

```
<option-definition name="nice-behavior" description="A nice behavior"/>
<option-definition name="move-around" description="A behavior for randomly moving around">
 <parameter name="move-around.speed" measure="mm/s" range="0..500"
       description="The speed with that the robot shall move"/>
</option-definition>
...
</option-definitions>
```

8.5 Opcions

Cada opció ha d'estar definida en un fitxer separat. Això seria un fitxer d'opcions:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<option-definitions
xmlns="http://www.ki.informatik.hu-berlin.de/XABSL2.1"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.1xabsl-2.1.option-definitions.xsd">
<option-definition name="nice-behavior" description="A nice behavior"/>
<option-definition name="move-around" description="A behavior for randomly moving around">
 <parameter name="move-around.speed" measure="mm/s" range="0..500"
       description="The speed with that the robot shall move"/>
</option-definition>
...
</option-definitions>
```

L'element opció és l'element root de l'arxiu d'opcions, té els següents atributs:

- 'name': El nom de l'opció. Ha de ser igual que el fitxer però sense l'extensió.
- 'initial-state': El nom de l'estat inicial. Aquest està serà activat quan entrem per primer cop des de l'execució del graf s'opcions.

Hem d'incloure tots els fitxers de definicions, de comportaments bàsics, i les definicions de les opcions. Com mostrem en l'exemple, podem declarar tot els

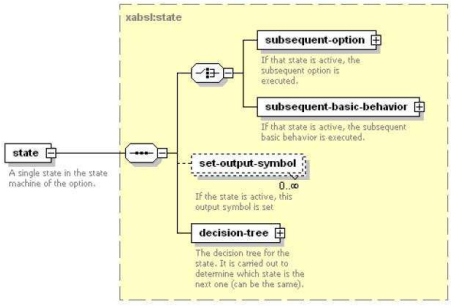


Figura 23. L'element estat

fitxers com a fitxers externs en el principi del document.

Després d'incloure els símbols, els comportaments bàsics i la definició d'opcions, podrem incloure un arbre de decisió comú.

Si tenim transicions amb les mateixes condicions en cada estat, aquestes condicions les podem posar dins un arbre de decisió comú. Aquestes condicions seran realitzades abans d'activar algun estat. Si cap de les condicions de l'arbre com és certa, continuarem aquí fins que alguna es compleixi.

8.6 Els estats

L'element estat (Figura 23 representa un estat simple de la màquina d'estats

```
<state name="first-state" is-target-state="true">
 <subsequent-basic-behavior ref="move-to">
 <set-parameter ref="move-to.x">
  <decimal-value value="42"/>
  </set-parameter>
 </subsequent-basic-behavior>
 <set-output-symbol ref="op-mode" value="op-mode.fast"/>
 <decision-tree>
  <if>
   <less-than>
    <decimal-input-symbol-ref ref="foo"/>
    <decimal-value value="14"/>
   <less-than>
   <transition-to-state ref="second-state"/>
  </if>
  <else>
   <transition-to-state ref="first-state"/>
  </else>
 </decision-tree>
</state>
```

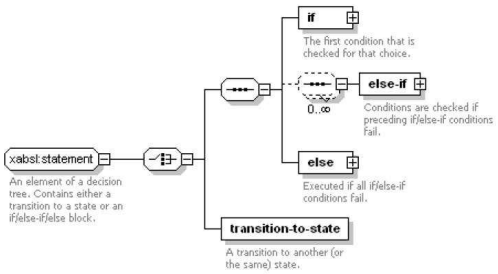


Figura 24. L'element arbre de decisió

Cada estat té una opció subsequent o be un subseqüent comportament bàsic. Això determina quina acció serà executada si entrem en aquest estat.

Si tenim un comportament que té un paràmetre, però en aquest estat no inicialitzem el paràmetre, el motor l'inicialitzara a 0.

8.7 Arbres de decisió

Cada estat té un arbre de decisió. La tasca d'aquest arbre de decisió s determinar la transició a un altre estat depenent del símbols d'entrada. Per tant les fulles de l'arbre de decisió són transicions.

Si l'element conte sempre un bloc de if/else-if/else o una transició a un estat. Un bloc if/else-if/else consisteix amb un element if, un element else-if opcional i un element else. La condició d'elements té una expressió booleana.

8.8 Expressions booleanes

Les expresions booleanes poden ser un dels element que mostrem en la figura 25.

- **boolean-input-symbol-ref**: Una referència a un símbol booleà.
- **enumerated-input-symbol-comparison:** Compara el valor de un enumerated-input-simbol amb el valor donat.
- **and, or:** L'operador booleà i (&&) i o (— —).
- **not:** L'operador booleà de negació (!).
- **equal-to, not-equal-to, less-than, less-than-or-equal-to, greater-than, greater-than-orequal- to:** Els operador igual, diferent, major que, menor que, major o igual que, i menor o igual que. Compraren dos elements.

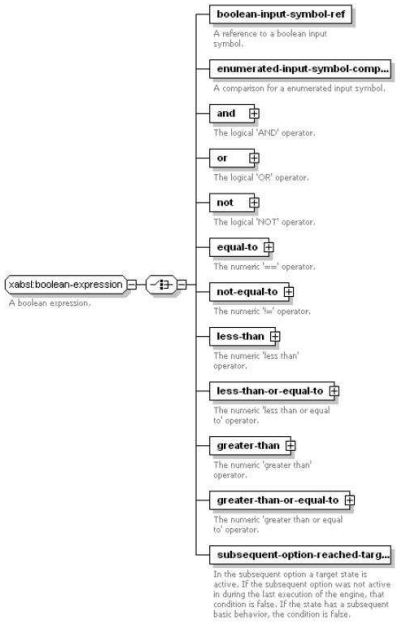


Figura 25. Les expresions booleanes

- **subsequent-option-reached-target-state:** Aquesta sentencia serà certa quan l'estat subsequent de la opció sigui una opció o bé quan l'estat actiu de la subsequent opció sigui una opció. En cas contrari serà fals.

8.9 Expressions decimals

La expresió decimal pot ser utilitzade dins d'una expressió booleana i per a la prametrització dels comportamnts subsequents. Pot ser un dels elements de la figura 26

- **decimal-input-function-call:** Crida a una funció decimal.
- **constant-ref:** Una referència a una costant que va està definida dins la col·lecció de símbols.
- **decimal-value:** Un valor decimal.
- **plus, minus, multiply, divide, mod:** Les operacions aritmètiques de suma, resta, multiplicació, divisió i mòdul.
- **time-of-state-execution:** El temps en que l'estat ha estat actiu aquest estat. Aquest cop és resetejat quan aquest estat no fos actiu des de l'execució del motor.
- **time-of-option-execution:** El temps en que aquesta opció esta sent activa. Aquest temps serà resetejat quant aquesta opció no sigui activa en l'ultima execució del motor.
- **conditional-expression:** Aquesta opció funciona com el una l'operador de C. La condició és comprovada, si és certa, retornem l'expressió decimal 1 en cas contrari l'expressió decimal 2. Els elements de la condició contenen una

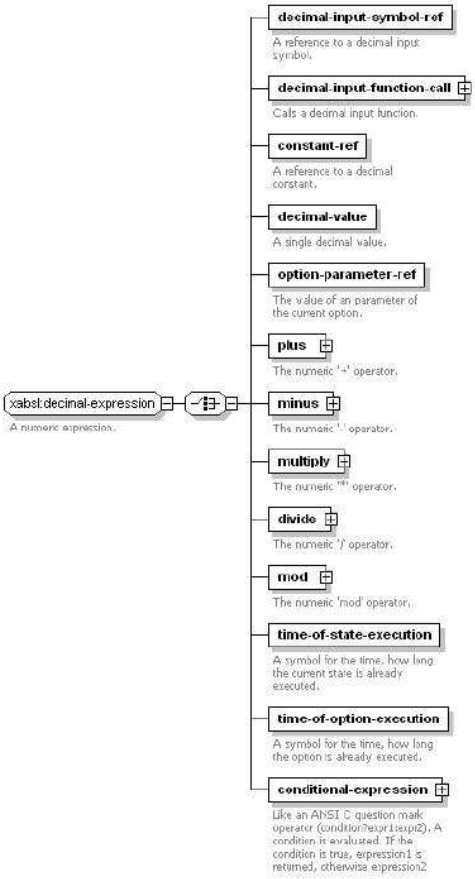


Figura 26. Les expresions decimals

expressió booleana i dues expressions decimals.

8.10 Agents

El fitxer 'agents.xml' és el fitxer principal del document XABSL. Ell inclou totes les opcions i defineix els agents. Aquí teniu un exemple d'aquest fitxer:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE options [
<!ENTITY options SYSTEM "options.xml">
]>
<agent-collection xmlns="http://www.ki.informatik.hu-berlin.de/XABSL2.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.1
xabsl-2.1.agent-collection.xsd"
xmlns:xi="http://www.w3.org/2001/XInclude">
<title>My XABSL behavior application</title>

<platform>My robot/agent platform.</platform>
<software-environment>My software platform</software-environment>
<agent id="default-agent" title="Default" description="The default agent behavior" root-option="foo"/>
<agent id="test-behavior" title="Test" description="A test environment for the option bla" root-option="bla"/>
```

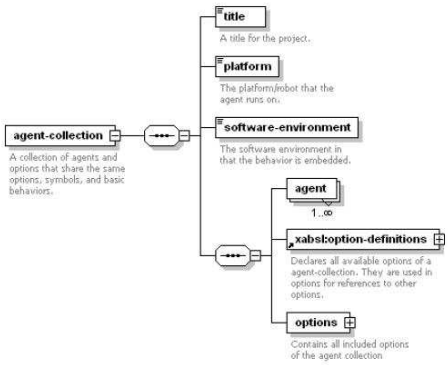


Figura 27. L'element agent

```
...
&options;
<options>
<xi:include href="Options/foo.xml"/>
<xi:include href="Options/bla.xml"/>
...
</options>
</agent-collection>
```

8.11 Integració de la màquina al codi C++

El següent pas és la integració d'aquest màquines d'estat escrites en XML en el nostre codi C++. Hem de tindre hem compte que s'han definir tots els comportaments bàsics, registrar els símbols dels comportaments, registrar els símbols de consulta del WorldState, i associar les funcions de C++ amb aquests símbols.

BasicBehaviors (Comportaments).

Comencem per els comportaments bàsics:

Hem de definir una classe per a cadascun dels comportaments on aquests heretin de la classe Xabsl2BasicBehavior i tinguin implementat el mètode virtual execute(), en aquesta funció hem de posar el valor de l'acció a executar en la variable action.

El nom del basicBehavior de l'arxiu XML l'hi hem de passar al constructor de la classe base, com podem veure en l'exemple "standUp"
.

```
class BasicBehaviorEstatStandUp : public Xabsl2BasicBehavior
```

```
{
public:
  BasicBehaviorEstatStandUp(Xabsl2ErrorHandler& errorHandler,
    WorldState& worldState, int& action)
    : Xabsl2BasicBehavior("standUp",errorHandler),
    worldState(worldState),
    action(action)
  {
  }

  virtual void execute();
private:
  WorldState& worldState; // el world state on es basa l'acció
  int& action; // l'accio a ser generada
};

void BasicBehaviorEstatStandUp::execute()
{
 //printf("void BasicBehaviorEstatStandUp::execute() \n");
 action=6;
}
```

Veure exemple MyBasicBehavior

En l'arxiu ../AiboXabsl/xml/my-basic-behaviors.xml

Hem de definir cadascun dels basic-behaviors.

Exemple:

```
<basic-behavior name="standUp" description="el gos s'aixeca de
l'estació de recarrega"/>
```

WorldState (estat de l'entorn)

Aquesta classe ha d'heretar de la classe Xabsl2FunctionProvider. Aquí definírem totes les funcions necessàries per conèixer des de les maquines d'estat l'estat en que es troba el gos AIBO. S'ha d'entendre aquesta classe com la classe que té la informació de tots els sensors, la posició i tots els aspectes rellevants de la visió del robot AIBO. Quan l'AIBO detecti un canvi en el seu estat ha de fer-ho saber-ho a aquesta classe, per tant hem de tenir un mètodes per actualitzar cadascuna de les variables. Aquesta classe també donarà la informació a la màquina d'estats així com ella ho vagi sol·licitant, per tant també s'ha de tenir uns mètodes per poder recollir cadascuna de les variables.

En l'arxiu ../AiboXabsl/xml/my-symbols.xml , definírem cadascun dels símbols que utilitzarem , després explicarem com associar aquests símbols amb la funció corresponent del WorldState.

```
<boolean-input-symbol name="cara" description="Hay cara"/>
<decimal-input-symbol name="tecla-apretada" description="Valor de
la tecla apretada" measure="px"/> <decimal-input-symbol
name="fiBallar" description="Variable que ens indica si ha acabat
de ballar" measure="px"/> <decimal-input-symbol name="fiCaminar"
description="Variable que ens indica si ha acabat de ballar"
measure="px"/> <decimal-input-symbol name="estatAibo"
description="Variable que ens indica si el gos esta en l'estació
de recarrega" measure="px"/> <decimal-input-symbol name="x"
description="The x position of the player" measure="px"/>
```

En aquest arxiu també definirem les constat que ens serviran per les comprovacions, per exemple:

```
<constant name="FIBALLAR" description="ens indica si s'ha acabat
de ballar" measure="int" value="11"/> <constant name="FICAMINAR"
description="ens indica si s'ha acabat de ballar" measure="int"
value="12"/>
```

PROCÉS.

Finalment en el nostre programa principal hem de fer les següents coses:

```
stat = new WorldState(); //Fem una instancia del WordlState.

int nextAction; //Variable on ficarem el valor de l'acció a
ejecutar. MyErrorHandler myErrorHandler;

//Definim cadascun del BasicsBehaviors
BasicBehaviorGetBehindBall
basicBehaviorGetBehindBall(myErrorHandler, worldState,
nextAction); BasicBehaviorGoTo basicBehaviorGoTo(myErrorHandler,
worldState, nextAction); BasicBehaviorSimpleAction
basicBehaviorSimpleAction(myErrorHandler, worldState, nextAction);
BasicBehaviorEstat2 basicBehaviorEstat2(myErrorHandler,
worldState, nextAction); BasicBehaviorEstatPara
basicBehaviorEstatPara(myErrorHandler, worldState, nextAction);
BasicBehaviorEstatSaluda basicBehaviorEstatSaluda(myErrorHandler,
worldState, nextAction); BasicBehaviorEstatBalla
basicBehaviorEstatBalla(myErrorHandler, worldState, nextAction);
BasicBehaviorEstatAnarEstacioRecarga
basicBehaviorEstatAnarEstacioRecarga(myErrorHandler, worldState,
nextAction); BasicBehaviorEstatNothing
basicBehaviorEstatNothing(myErrorHandler, worldState, nextAction);
BasicBehaviorEstatStandUp
basicBehaviorEstatStandUp(myErrorHandler, worldState, nextAction);
//Cream un nou motor
pEngine = new Xabsl2Engine(myErrorHandler, &getCurrentSystemTime);

// Registrem tots els BasicBehaviors amb el nostre motor.
pEngine->registerBasicBehavior(basicBehaviorGetBehindBall);
pEngine->registerBasicBehavior(basicBehaviorGoTo);
pEngine->registerBasicBehavior(basicBehaviorSimpleAction);
pEngine->registerBasicBehavior(basicBehaviorEstat2);
pEngine->registerBasicBehavior(basicBehaviorEstatPara);
pEngine->registerBasicBehavior(basicBehaviorEstatSaluda);
pEngine->registerBasicBehavior(basicBehaviorEstatBalla);
pEngine->registerBasicBehavior(basicBehaviorEstatAnarEstacioRecarga);
pEngine->registerBasicBehavior(basicBehaviorEstatNothing);
pEngine->registerBasicBehavior(basicBehaviorEstatStandUp);

// Registrem els símbols que utilitzarem i els diem quina serà la funció del WorldState que
// utilitzaran per poder obtenir el valor.
```

```
pEngine->registerDecimalInputSymbol("tecla-apretada", &worldState,
     (double (Xabsl2FunctionProvider::*)())&WorldState::getTeclaApretada);
pEngine->registerDecimalInputSymbol("fiBallar", &worldState,
     (double (Xabsl2FunctionProvider::*)())&WorldState::getFiBallar);
pEngine->registerDecimalInputSymbol("fiCaminar", &worldState,
     (double (Xabsl2FunctionProvider::*)())&WorldState::getFiCaminar);
pEngine->registerDecimalInputSymbol("estatAibo", &worldState,
     (double (Xabsl2FunctionProvider::*)())&WorldState::getEstatEstacio);

MyFileInputSource input("intermediate-code.dat");  //Agafem
l'arxiu generat pels xml pEngine->createOptionGraph(input);
//Cream ef graf


While(..) { if (!myErrorHandler.errorsOccurred) {
    pEngine->execute();
    worldState.printField(myErrorHandler);
    player_move= nextAction;
} else {
    printf("Hi ha hagut un error;");
    player_move= 1;
}



//En aquest punt tenim a la variable nextAction , el valor de d'acció a realitzar, haurem de
//de comprovar si la variable és igual o no a l'estat anterior, per tal de dir-li
//al gos que executi una nova acció o no.
}
```

Options:

En aquest arxiu d'XML definirem totes les opcions que tenim, per cada opció haurem de definir l'arxiu ".Options/nomOpcio.xml"
En el nostre exemple únicament tenim una opció definida, ïnici".

Podria ser interessant tenir una opció per quan el gos està en l'estació i una altre per quan esta aixecat.

En l'inici serà on definirem la maquina d'estats d'aquesta opció.

Exemple Maquina d'estats:

```
<?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE
dummy-doc-type [
    <!ENTITY my-symbols SYSTEM "../my-symbols.xml">
    <!ENTITY my-basic-behaviors SYSTEM "../my-basic-behaviors.xml">
    <!ENTITY options SYSTEM "../options.xml">
]> <option xmlns="http://www.ki.informatik.hu-berlin.de/XABSL2.2"
xmlns:xi="http://www.w3.org/2003/XInclude"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ki.informatik.hu-berlin.de/XABSL2.2
../../../Xabsl2/xabsl-2.2/xabsl-2.2.option.xsd" name="inici"
initial-state="estatInici">
    &my-symbols;
```

```
    &my-basic-behaviors;
    &options;

<state name="estatInici">
    <subsequent-basic-behavior ref="estat2"/>
    <decision-tree>
        <if>
            <condition description="tecla apretada es amunt">
            <equal-to>
                <decimal-input-symbol-ref ref="tecla-apretada"/>
                <constant-ref ref="CARA"/>
            </equal-to>
            </condition>
            <transition-to-state ref="estatStandUp"/>
        </if>
        <else>
            <transition-to-state ref="estatInici"/>
        </else>
    </decision-tree>
</state>

<state name="estatBalla">
    <subsequent-basic-behavior ref="estatBalla"/>
    <decision-tree>
        <if>
            <condition description="ha acabat de ballar">
            <equal-to>
                <decimal-input-symbol-ref ref="fiBallar"/>
                <constant-ref ref="N"/>
            </equal-to>
            </condition>
            <transition-to-state ref="estatAnarEstacioRecarga"/>
        </if>
        <else>
            <transition-to-state ref="estatBalla"/>
        </else>
    </decision-tree>
</state>

<state name="estatCamina">
    <subsequent-basic-behavior ref="get-behind-ball"/>
    <decision-tree>
        <if>
            <condition description="Ha acabat de caminar?">
            <equal-to>
                <decimal-input-symbol-ref ref="fiCaminar"/>
                <constant-ref ref="N"/>
            </equal-to>
            </condition>
            <transition-to-state ref="estatCap"/>
        </if>
        <else>
            <transition-to-state ref="estatCamina"/>
        </else>
    </decision-tree>
</state>

<state name="estatCap">
    <subsequent-basic-behavior ref="nothing"/>
    <decision-tree>
        <if>
            <condition description="M'han tocat el cap , han apretat tecla C ?">
            <equal-to>
                <decimal-input-symbol-ref ref="tecla-apretada"/>
                <constant-ref ref="C"/>
            </equal-to>
            </condition>
            <transition-to-state ref="estatBalla"/>
```

```
        </if>
        <else>
            <transition-to-state ref="estatCap"/>
        </else>
    </decision-tree>
</state>

<state name="estatAnarEstacioRecarga">
    <subsequent-basic-behavior ref="anarEstacioRecarga"/>
    <decision-tree>
        <if>
            <condition description="Esta en l'estació de recarga">
                <equal-to>
                    <decimal-input-symbol-ref ref="tecla-apretada"/>
                    <constant-ref ref="ESTACIO"/>
                </equal-to>
            </condition>
            <transition-to-state ref="estatEstacio"/>
        </if>
        <else>
            <transition-to-state ref="estatAnarEstacioRecarga"/>
        </else>
    </decision-tree>
</state>

<state name="estatEstacio">
    <subsequent-basic-behavior ref="nothing"/>
    <decision-tree>
        <if>
            <condition description="Esta Carregat o han apretat tecla B">
            <equal-to>
                <decimal-input-symbol-ref ref="tecla-apretada"/>
                <constant-ref ref="B"/>
            </equal-to>
            </condition>
            <transition-to-state ref="estatStandUp"/>
        </if>
        <else>
            <if>
                <condition description="tecla apretada es amunt">
                <equal-to>
                    <decimal-input-symbol-ref ref="tecla-apretada"/>
                    <constant-ref ref="CARA"/>
                </equal-to>
                </condition>
                <transition-to-state ref="estatSaluda"/>
            </if>
            <else>
                <transition-to-state ref="estatEstacio"/>
            </else>
        </else>
    </decision-tree>
</state>

<state name="estatSaluda">
    <subsequent-basic-behavior ref="estatSaluda"/>
    <decision-tree>
        <if>
            <condition description="ha acabat de ballar">
                <equal-to>
                    <decimal-input-symbol-ref ref="fiBallar"/>
                    <constant-ref ref="N"/>
                </equal-to>
            </condition>
            <transition-to-state ref="estatEstacio"/>
        </if>
        <else>
            <transition-to-state ref="estatSaluda"/>
```

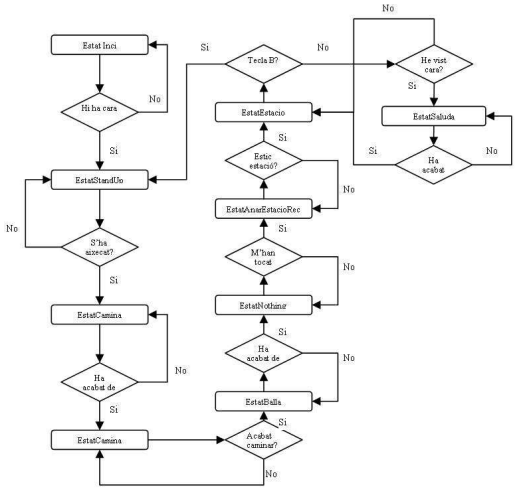


Figura 28. Graf del comportament

```
        </else>
    </decision-tree>
</state>


<state name="estatStandUp">
    <subsequent-basic-behavior ref="standUp"/>
    <decision-tree>
        <if>
            <condition description="El gos s'hauria d'aixecar de l'estació de recarga">
            <equal-to>
                <decimal-input-symbol-ref ref="estatAibo"/>
                <constant-ref ref="N"/>
            </equal-to>
            </condition>
            <transition-to-state ref="estatCamina"/>
        </if>
        <else>
            <transition-to-state ref="estatStandUp"/>
        </else>
    </decision-tree>
</state>

</option>
```

Passos per compilar.

Ens hem de situar en la carpeta ../AiboXabls/xml/ Fem "make all".
Això validara la nostra maquina d'estats i en cas d'èxit ens generà l'arxiu 'intermediate-code.dat' i 'debug-symbols.dat', que després aquests arxius seran utilitzats posteriorment en el projecte de visual.

Hem de tenir hem compta que cada cop que fem una modificació en els arxius XML, haurem de tornar a generar aquests fitxers.

Un cop tenim aquest arxius generats podem compilar i executar el projecte de Visual.

En l'hora d'execucío el programa genererà un arxiu anomenat 'messages.log' en la mateixa carpeta de l'executable. Si tot fos correcta aquest arxiu hauria d'estar buit, en cas contrari vol dir que s'han trobat els errors que l'arxiu ens indica.

9 Appendices

9.1 ***Appendix B The AIBO® Entertainment Robot ERS-7M3 parts***

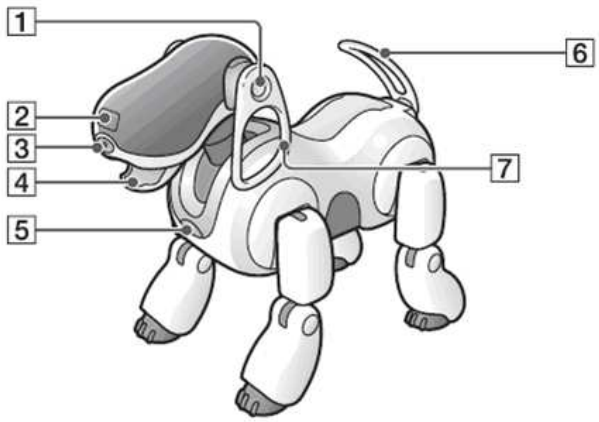


Figura 29. Sony AIBO Robot

[1] Stereo microphones Allow the AIBO® Entertainment Robot to listen to the surrounding environment.

[2] Head distance sensor Measures the distance between the AIBO robot and other objects.

[3] Color camera Detects the color, shape, and movement of nearby objects.

[4] Mouth Picks up the AIBOne toy and expresses emotions.

[5] Chest distance sensor Measures the distance between the AIBO robot and other objects.

[6]Tail Moves up, down, left, and right to express the AIBO robot's emotions.

[7] Ears Indicates the AIBO robot's emotions and condition. See fig.29

[8] Head sensor Detects and turns white when you gently stroke the AIBO robot's head.

[9] Wireless light (on the back of the AIBO robot's head) Indicator used with the wireless LAN function. This light turns blue when the AIBO robot is connected to the e-mail server.

[10] Pause button When pressed, the AIBO robot's activity will pause or resume.

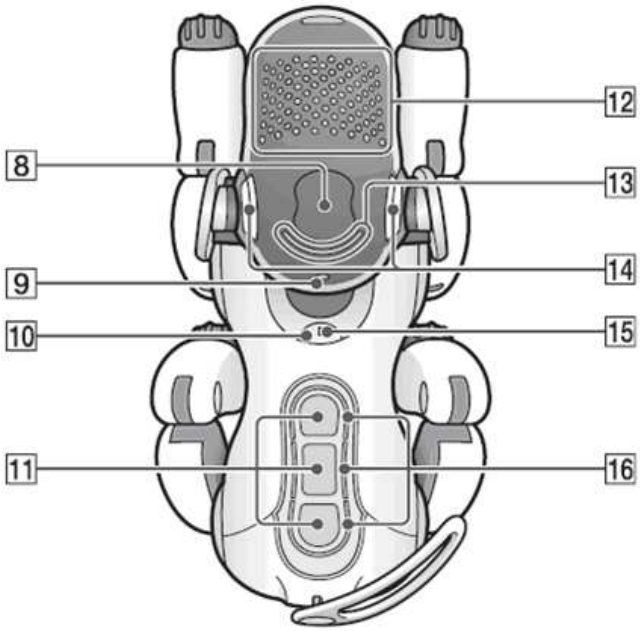


Figura 30. Sony AIBO Robot

[11] Back sensors (front, middle, and rear) Detect and turn white when you gently stroke the AIBO robot's back.

[12] Face lights (illuminated face) These lights turn various colors to show the AIBO robot's emotions and conditions.

[13] Head light Detects and turns white when you touch the head sensor. Lights / flashes orange when one of the AIBO robot's joints is jammed (page 43).

[14] Mode indicators (inner side of ears) These indicate the present mode and condition of the AIBO robot (page 42).

[15] Operation light During operation: turns green. During preparation for shutdown: flashes green. During charging: turns orange. When a charging error occurs: flashes orange. When operation stops: turns OFF. Outside hours of activity (Sleeping on the Energy Station): slowly flashing green.

[16] Back lights (front, middle, and rear) Detect and turn white when you gently touch the AIBO robot's back sensors. These lights also turn blue (front), orange (middle), and red (rear) to indicate a variety of actions.See fig.30

[1] Paw sensors These are located on the bottom of the AIBO® Entertainment Robot's paws, and detect contact with any surface it touches. When the AIBO robot extends one of its paws, it will react with happiness if you touch it.

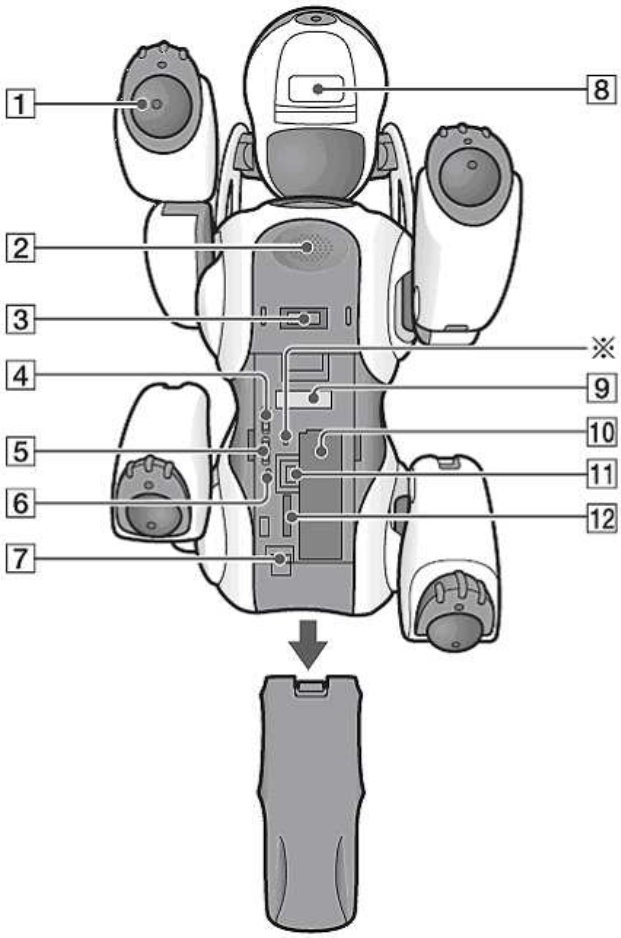


Figura 31. Sony AIBO Robot

[2] Speaker Emits music, sound effects, and voice guide.

[3] Charging terminal When you place the AIBO robot on the Energy Station, this part makes contact with the station to allow charging of the AIBO robot's battery.

[4] Volume control switch (VOLUME) Adjusts the volume of the speaker to one of four levels (including no sound).

[5] Wireless LAN switch (WIRELESS) This turns the AIBO robot's wireless LAN function ON or OFF.

[6] "Memory Stick^{TM}"media access indicator This indicator turns red while the AIBO robot is reading or writing to a "Memory Stick"media.

[7] Battery pack latch (BATT Z) Flip this latch to the rear when you want to remove the battery.

[8] Chin sensor Senses when you touch the AIBO robot's chin.

[9] FCC ID/MAC address label Indicates the FCC ID and MAC address of the AIBO robot's wireless unit.

[10] Battery slot Holds the AIBO robot's lithium-ion battery.

[11] "Memory Stick"media eject button (Z) Press to eject the "Memory Stick"media. L "Memory Stick"media slot This is where you insert the provided AIBO-ware "Memory Stick"media. See fig.31

9.2 ***Appendix C Computer Vision***

The field of computer vision can be characterized as immature and diverse. Even though earlier work exists, it was not until the late 1970's that a more focused study of the field started when computers could manage the processing of large data sets such as images. However, these studies usually originated from various other fields, and consequently there is no standard formulation of the "computer vision problem". Also, and to an even larger extent, there is no standard formulation of how computer vision problems should be solved. Instead, there exists an abundance of methods for solving various well-defined computer vision tasks, where the methods often are very task specific and seldom can be generalized over a wide range of applications. Many of the methods and applications are still in the state of basic research, but more and more methods have found their way into commercial products, where they often constitute a part of a larger system which can solve complex tasks (e.g., in the area of medical images, or quality control and measurements in industrial processes).

Computer vision is by some seen as a subfield of artificial intelligence where image data is being fed into a system as an alternative to text based input for controlling the behavior of a system. Some of the learning methods which are used in computer vision are based on learning techniques developed within artificial intelligence.

Since a camera can be seen as a light sensor, there are various methods in computer vision based on correspondences between a physical phenomenon related to light and images of that phenomenon. For example, it is possible to extract information about motion in fluids and about waves by analyzing images of these phenomena. Also, a subfield within computer vision deals with the physical process which given a scene of objects, light sources, and camera lenses forms the image in a camera. Consequently, computer vision can also be seen as an extension of physics.

A third field which plays an important role is neurobiology, specifically the study of the biological vision system. Over the last century, there has been an extensive study of eyes, neurons, and the brain structures devoted to processing of visual stimuli in both humans and various animals. This has led to a coarse, yet complicated, description of how "real" vision systems operate in order to solve certain vision related tasks. These results have led to a subfield within computer vision where artificial systems are designed to mimic the processing and behavior of biological systems, at different levels of complexity. Also, some of the learning-based methods developed within computer vision have their background in biology.

Yet another field related to computer vision is signal processing. Many existing methods for processing of one-variable signals, typically temporal signals, can be extended in a natural way to processing of two-variable signals or multi-variable signals in computer vision. However, because of the specific nature of images there are many methods developed within computer vision which have no counterpart in the processing of one-variable signals. A distinct character of these methods is the fact that they are non-linear which, together with the multi-dimensionality of the 19 22 signal, defines a subfield in signal processing as a part of computer vision.

Beside the above mentioned views on computer vision, many of the related research topics can also be studied from a purely mathematical point of view. For example, many methods in computer vision are based on statistics, optimization or geometry. Finally, a significant part of the field is devoted to the implementation aspect of computer vision; how existing methods can be realized in various combinations of software and hardware, or how these methods can be modified in order to gain processing speed without losing too much performance.

9.2.1 Related fields

Computer vision, Image processing, Image analysis, Robot vision and Machine vision are closely related fields. If you look inside text books which have either of these names in the title there is a significant overlap in terms of what techniques and applications they cover. This implies that the basic techniques that are used and developed in these fields are more or less identical, something which can be interpreted as there is only one field with different names.

On the other hand, it appears to be necessary for research groups, scientific journals, conferences and companies to present or market themselves as belonging specifically to one of these fields and, hence, various characterizations which distinguish each of the fields from the others have been presented. The following characterizations appear relevant but should not be taken as universally accepted.

Image processing and Image analysis tend to focus on 2D images, how to transform one image to another, e.g., by pixel-wise operations such as contrast enhancement, local operations such as edge extraction or noise removal, or geometrical transformations such as rotating the image. This characterization implies that image processing/ analysis does not produce nor require assumptions about what a specific image is an image of.

Computer vision tends to focus on the 3D scene projected onto one or several images, e.g., how to reconstruct structure or other information about the 3D scene from one or several images. Computer vision often relies on more or less

complex assumptions about the scene depicted in an image.

Machine vision tends to focus on applications, mainly in industry, e.g., vision based autonomous robots and systems for vision based inspection or measurement. This implies that image sensor technologies and control theory often are integrated with the processing of image data to control a robot and that real-time processing is emphasized by means of efficient implementations in hardware and software. There is also a field called Imaging which primarily focus on the process of producing images, but sometimes also deals with processing and analysis of images. For example, Medical imaging contains lots of work on the analysis of image data in medical applications.

Finally, pattern recognition is a field which uses various methods to extract information from signals in general, mainly based on statistical approaches. A significant part of this field is devoted to applying these methods to image data. A consequence of this state of affairs is that you can be working in a lab related to one of these fields, apply methods from a second field to solve a problem in a third field and present the result at a conference related to a fourth field!

9.2.2 Examples of applications for computer vision

Another way to describe computer vision is in terms of applications areas. One of the most prominent application fields is medical computer vision or medical image processing. This area is characterized by the extraction of information from image data for the purpose of making a medical diagnosis of a patient. Typically image data is in the form of microscopy images, X-ray images, angiography images, ultrasonic images, and tomography images. An example of information which can be extracted from such image data is detection of tumours, arteriosclerosis or other malign changes. It can also be measurements of organ dimensions, blood flow, etc. This application area also supports medical research by providing new information, e.g., about the structure of the brain, or about the quality of medical treatments.

A second application area in computer vision is in industry. Here, information is extracted for the purpose of supporting a manufacturing process. One example is quality control where details or final products are being automatically inspected in order to find defects. Another example is measurement of position and orientation of details to be picked up by a robot arm. See the article on machine vision for more details on this area.

Military applications are probably one of the largest areas for computer vision, even though only a small part of this work is open to the public. The obvious examples are detection of enemy soldiers or vehicles and guidance of missiles to a designated target. More advanced systems for missile guidance

send the missile to an area rather than a specific target, and target selection is made when the missile reaches the area based on locally acquired image data. Modern military concepts, such as "battlefield awareness," imply that various sensors, including image sensors, provide a rich set of information about a combat scene which can be used to support strategic decisions. In this case, automatic processing of the data is used to reduce complexity and to fuse information from multiple sensors to increase reliability.

One of the newer application areas is autonomous vehicles, which include submersibles, land-based vehicles (small robots with wheels, cars or trucks), and aerial vehicles. An unmanned aerial vehicle is often denoted UAV. The level of autonomy ranges from fully autonomous (unmanned) vehicles to vehicles where computer vision based systems support a driver or a pilot in various situations. Fully autonomous vehicles typically use computer vision for navigation, i.e. for knowing where it is, or for producing a map of its environment (SLAM) and for detecting obstacles. It can also be used for detecting certain task specific events, e. g., a UAV looking for forest fires. Examples of supporting system are obstacle warning systems 21 24 in cars and systems for autonomous landing of aircraft. Several car manufacturers have demonstrated systems for autonomous driving of cars, but this technology has still not reached a level where it can be put on the market. There are ample examples of military autonomous vehicles ranging from advanced missiles to UAVs for recon missions or missile guidance. Space exploration is already being made with autonomous vehicles using computer vision, e. g., NASA's Mars Exploration Rover.

Other application areas include the creation of visual effects for cinema and broadcast, e.g., camera tracking or matchmoving, and surveillance.

9.2.3 Typical tasks of computer vision

Object recognition

Detecting the presence of known objects or living beings in an image, possibly together with estimating the pose of these objects.

Examples: Searching in digital images for specific content (content-based image retrieval) Recognizing human faces and their location in images. Estimation of the three-dimensional pose of humans and their limbs Detection of objects which are passing through a manufacturing process, e.g., on a conveyor belt, and estimation of their pose so that a robot arm can pick up the objects from the belt. Optical character recognition OCR (optical character recognition) takes pictures of printed or handwritten text and converts it into computer readable text such as ASCII or Unicode. In the past images were acquired with a computer scanner, however more recently some software can

also read text from pictures taken with a digital camera.

Tracking

Tracking known objects through an image sequence.

Examples: Tracking a single person walking through a shopping center. Tracking of vehicles moving along a road.

Scene interpretation

Creating a model from an image/video.

Examples: Creating a model of the surrounding terrain from images, which are being taken by a robot-mounted camera. Anticipating the pattern of the image to determine size and density to estimate the volume using tomography like device. The cloud recognition is one the government project using this method.

Egomotion

The goal of egomotion computation is to describe the motion of an object with respect to an external reference system, by analyzing data acquired by sensors onboard on the object. i.e. the camera itself.

Examples: Given two images of a scene, determine the 3d rigid motion of the camera between the two views.

9.2.4 Computer vision systems

A typical computer vision system can be divided in the following subsystems:

Image acquisition

The image or image sequence is acquired with an imaging system (camera, radar, lidar, tomography system). Often the imaging system has to be calibrated before being used.

Preprocessing

In the preprocessing step, the image is being treated with "low-level"-operations. The aim of this step is to do noise reduction on the image (i.e. to dissociate the signal from the noise) and to reduce the overall amount of data. This is typically being done by employing different (digital)image processing methods such as: Downsampling the image. Applying digital filters convolutions, computing a scale space representation Correlations or linear shift invariant filters Sobel

operator Computing the x- and y-gradient (possibly also the time-gradient). Segmenting the image. Pixelwise thresholding. Performing an eigentransform on the image Fourier transform Doing motion estimation for local regions of the image (also known as optical flow estimation). Estimating disparity in stereo images.

Feature extraction

The aim of feature extraction is to further reduce the data to a set of features, which ought to be invariant to disturbances such as lighting conditions, camera position, noise and distortion. Examples of feature extraction are: Performing edge detection or estimation of local orientation. Extracting corner features. Detecting blob features. Extracting spin images from depth maps. Extracting geons or other three-dimensional primitives, such as superquadrics. Acquiring contour lines and maybe curvature zero crossings. Generating features with the Scale-invariant feature transform.

Registration

The aim of the registration step is to establish correspondence between the features in the acquired set and the features of known objects in a model-database and/or the features of the preceding image. The registration step has to bring up a final hypothesis. To name a few methods: Least squares estimation Hough transform in many variations Geometric hashing Particle filtering RANdom SAmple Consensus.

9.3 ***Appendix D Traffic sign recognition mobile mapping adquisition***

The Traffic Sign Recognition (TSR) is a field of applied computer vision research concerned with the automatical detection and classification of traffic signs in traffic scene images acquired from a moving car. Most part of the work done in this field is enclosed in the problem of the Intelligent Transportation Systems (ITS), which aim is to provide Driver Support Systems (DSS) with the ability to understand its neighborhood environment and so permit advanced driver support such as collision prediction and avoidance. Driving is a task based fully on visual information processing. The road signs and traffic signals define a visual language interpreted by drivers. Road signs carry many information necessary for successful driving - they describe current traffic situation, define right-of-way, prohibit or permit certain directions, warn about risky factors, etc. Road signs also help drivers with navigation. Two basic applications of TSR are under consideration in the research community - driver's aid (DSS) and automated surveillance of road traffic devices. It is desirable to design smart car control systems in such a way to allow evolution of fully autonomous vehicles in the future. The TSR system is also being considered as the valuable complement of the GPS-based navigation system. The dynamical environmental map may be enriched by road sign types and positions (acquired by TSR) and so help with the precision of current vehicle position.

Mobile mapping: the Geomobil project on Mobile mapping is a useful technique used to compile cartographic information from a mobile vehicle. The mobile vehicle is usually equipped with a set of sensors synchronized with an orientation system in order to link the obtained information with its position over the map. We are working with the mobile mapping system named Geomobil. The Geomobil is a Land Based Mobile Mapping System (LBMMS) developed by the Institut Cartogr'afic de Catalunya (ICC) (fig.32). It is a modular system that allows the direct orientation of any sensor mounted on a roof platform. The Geomobil system is composed of the following subsystems: orientation subsystem, image subsystem, laser ranging subsystem, synchronization subsystem, power and environmental control subsystem. In our case we only use information from the image and orientation subsystems, which will be briefly explained in the rest of this point.

Geomobil system: the orientation subsystem is responsible for georeferencing the images acquired by the Geomobil. Thus it provides the coordinates (position) and the angles (attitude) of their projection centers. It is a system that combines inertial and GPS observations at a high level of integration, where the GPS derived trajectories are used to correct and calibrate the drifts of the Inertial Measurement Unit (IMU) gyros and accelerometers so that the

position and velocity errors derived from inertial sensors are minimized. This combination of GPS and IMU systems allows the system to calculate the position even when the GPS satellites signals are blocked by terrain conditions (buildings, bridges, tunnels,...). The image subsystem design has been driven by two main requirements: to acquire images of at least 1Mpix and to get 10m stereoscopic overlap at a 10m distance from the van.

Figura 32. Geomobil system.

The stereo overlap is conditioned by two factors: getting the maximum stereoscopic overlap free of obstacles and preserving a B/D ratio (stereoscopic Base - object Distance) as good as possible. The system links the captured images with their position and orientation data, and saves the information to the discs. The acquisition frequency is limited by the storage system capacity, and nowadays is programmed to take a stereo-pair of images each 10 meters or a turn higher than 60 degrees, which corresponds to the camera field of view

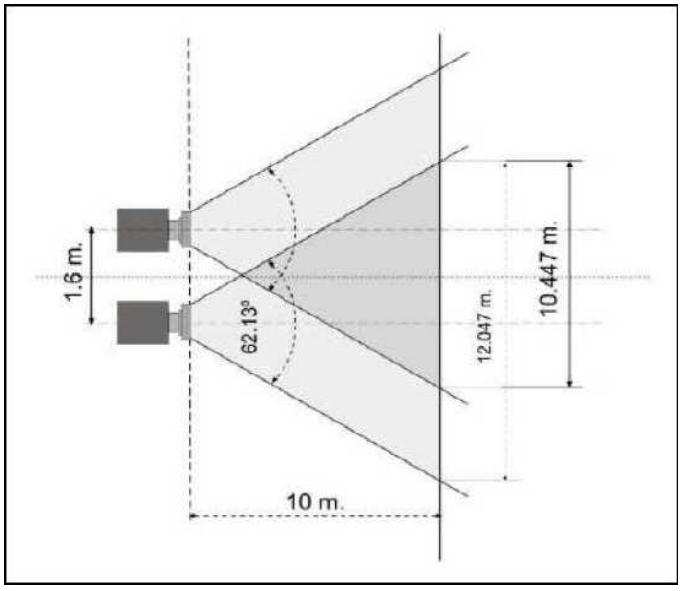


Figura 33. Stereoscopic system diagram. We can see the relation between overlap zone and distance.

Feature	Value
Number of pixels	1024×1020
Pixel size	$12\mu m$
Focal length	$10.2mm$
FOV	62.13^{o}
IFOV	3 min. 38 sec.
Precision at $10m$ (across-track)	$0.8cm$
Precision at $10m$ (along-track)	$5.6cm$

Figura 34. Geovan camera characteristics.

9.4 ***Appendix E Artificial Intelligence History***

9.4.1 Prehistory of AI

Humans have always speculated about the nature of mind, thought, and language, and searched for discrete representations of their knowledge. Aristotle tried to formalize this speculation by means of syllogistic logic, which remains one of the key strategies of AI. The first is-a hierarchy was created in 260 by Porphyry of Tyros. Classical and medieval grammarians explored more subtle features of language that Aristotle shortchanged, and mathematician Bernard Bolzano made the first modern attempt to formalize semantics in 1837.

Early computer design was driven mainly by the complex mathematics needed to target weapons accurately, with analog feedback devices inspiring an ideal of cybernetics. The expression "artificial intelligence"was introduced as a 'digital' replacement for the analog 'cybernetics'.

9.4.2 Development of AI theory

Much of the (original) focus of artificial intelligence research draws from an experimental approach to psychology, and emphasizes what may be called linguistic intelligence (best exemplified in the Turing test).

Approaches to Artificial Intelligence that do not focus on linguistic intelligence include robotics and collective intelligence approaches, which focus on active manipulation of an environment, or consensus decision making, and draw from biology and political science when seeking models of how ïntelligent"behavior is organized. AI also draws from animal studies, in particular with insects, which are easier to emulate as robots (see artificial life), as well as animals with more complex cognition, including apes, who resemble humans in many ways but have less developed capacities for planning and cognition. Some researchers argue that animals, which are apparently simpler than humans, ought to be considerably easier to mimic. But satisfactory computational models for animal intelligence are not available.

Seminal papers advancing AI include "A Logical Calculus of the Ideas Immanent in Nervous Activity"(1943), by Warren McCulloch and Walter Pitts, and "On Computing Machinery and Intelligence"(1950), by Alan Turing, and "Man-Computer Symbiosis"by J.C.R. Licklider. See Cybernetics and Turing test for further discussion. There were also early papers which denied the possibility of machine intelligence on logical or philosophical grounds such as "Minds, Machines and G¨odel"(1961) by John Lucas. With the development of practical techniques based on AI research, advocates of 10 AI have argued that opponents of AI have repeatedly changed their position on tasks

such as computer chess or speech recognition that were previously regarded as ïntelligentïn order to deny the accomplishments of AI. Douglas Hofstadter, in Godel, Escher, Bach, pointed out that this moving of the goalposts effectively defines ïntelligence"as "whatever humans can do that machines cannot". John von Neumann (quoted by E.T. Jaynes) anticipated this in 1948 by saying, in response to a comment at a lecture that it was impossible for a machine to think: "You insist that there is something a machine cannot do. If you will tell me precisely what it is that a machine cannot do, then I can always make a machine which will do just that!". Von Neumann was presumably alluding to the Church-Turing thesis which states that any effective procedure can be simulated by a (generalized) computer.

In 1969 McCarthy and Hayes started the discussion about the frame problem with their essay, "Some Philosophical Problems from the Standpoint of Artificial Intelligence".

9.4.3 Experimental AI research

Artificial intelligence began as an experimental field in the 1950s with such pioneers as Allen Newell and Herbert Simon, who founded the first artificial intelligence laboratory at Carnegie Mellon University, and John McCarthy and Marvin Minsky, who founded the MIT AI Lab in 1959. They all attended the Dartmouth College summer AI conference in 1956, which was organized by McCarthy, Minsky, Nathan Rochester of IBM and Claude Shannon.

Historically, there are two broad styles of AI research - the "neats"and "scruffies". "Neat", classical or symbolic AI research, in general, involves symbolic manipulation of abstract concepts, and is the methodology used in most expert systems. Parallel to this are the "scruffy", or çonnectionist", approaches, of which artificial neural networks are the best-known example, which try to "evolveïntelligence through building systems and then improving them through some automatic process rather than systematically designing something to complete the task. Both approaches appeared very early in AI history.

Throughout the 1960s and 1970s scruffy approaches were pushed to the background, but interest was regained in the 1980s when the limitations of the "neat"approaches of the time became clearer. However, it has become clear that contemporary methods using both broad approaches have severe limitations.

Artificial intelligence research was very heavily funded in the 1980s by the Defense Advanced Research Projects Agency in the United States and by the fifth generation computer systems project in Japan. The failure of the work funded at the time to produce immediate results, despite the grandiose promises of some AI practitioners, led to correspondingly large cutbacks in funding

by government agencies in the late 1980s, leading to a general downturn in activity in the field known as AI winter. Over the following decade, many AI researchers moved into related areas with more modest goals such as machine learning, robotics, and computer vision, though research in pure AI continued at reduced levels.

9.4.4 Micro-World AI

The real world is full of distracting and obscuring detail: generally science progresses by focusing on artificially simple models of reality (in physics, frictionless planes and perfectly rigid bodies, for example). In 1970 Marvin Minsky and Seymour Papert, of the MIT AI Laboratory, proposed that AI research should likewise focus on developing programs capable of intelligent behaviour in artificially simple situations known as micro-worlds. Much research has focused on the so-called blocks world, which consists of coloured blocks of various shapes and sizes arrayed on a flat surface.

9.4.5 Spinoffs

Whilst progress towards the ultimate goal of human-like intelligence has been slow, many spinoffs have come in the process. Notable examples include the languages LISP and Prolog, which were invented for AI research but are now used for non-AI tasks. Hacker culture first sprang from AI laboratories, in particular the MIT AI Lab, home at various times to such luminaries as John McCarthy, Marvin Minsky, Seymour Papert (who developed Logo there) and Terry Winograd (who abandoned AI after developing SHRDLU).

9.4.6 AI languages and programming styles

AI research has led to many advances in programming languages including the first list processing language by Allen Newell et. al., Lisp dialects, Planner, Actors, the Scientific Community Metaphor, production systems, and rule-based languages. GOFAI TEST research is often done in programming languages such as Prolog or Lisp. Bayesian work often uses Matlab or Lush (a numerical dialect of Lisp). These languages include many specialist probabilistic libraries. Real-life and especially real-time systems are likely to use C++. AI programmers are often academics and emphasise rapid development and prototyping rather than bulletproof software engineering practices, hence the use of interpreted languages to empower rapid command-line testing and experimentation.

The most basic AI program is a single If-Then statement, such as Ïf A, then B.Ïf you type an 'A' letter, the computer will show you a 'B' letter. Basically,

you are teaching a computer to do a task. You input one thing, and the computer responds with something you told it to do or say. All programs have If-Then logic. A more complex example is if you type in "Hello.", and the computer responds "How are you today?"This response is not the computer's own thought, but rather a line you wrote into the program before. Whenever you type in "Hello.", the computer always responds "How are you today?". It seems as if the computer is alive and thinking to the casual observer, but actually it is an automated response. AI is often a long series of If-Then (or Cause and Effect) statements.

A randomizer can be added to this. The randomizer creates two or more response paths. For example, if you type "Hello", the computer may respond with "How are you today?"or "Nice weather"or "Would you like to play a game?"Three responses (or 'thens') are now possible instead of one. There is an equal chance that any one of the three responses will show. This is similar to a pull-cord talking doll that can respond with a number of sayings. A computer AI program can have thousands of responses to the same input. This makes it less predictable and closer to how a real person would respond, arguably because living people respond somewhat unpredictably. When thousands of input ("if") are written in (not just "Hello.") and thousands of responses ("then") are written into the AI program, then the computer can talk (or type) with most people, if those people know the If statement input lines to type.

Many games, like chess and strategy games, use action responses instead of typed responses, so that players can play against the computer. Robots with AI brains would use If-Then statements and randomizers to make decisions and speak. However, the input may be a sensed object in front of the robot instead of a "Hello.l·line, and the response may be to pick up the object instead of a response line.

9.4.7 Chronological History

Historical Antecedents

Greek myths of Hephaestus and Pygmalion incorporate the idea of intelligent robots. In the 5th century BC, Aristotle invented syllogistic logic, the first formal deductive reasoning system.

Ramon Llull, Spanish theologian, invented paper "machines"for discovering nonmathematical truths through combinattions of words from lists in the 13th century. By the 15th century and 16th century, clocks, the first modern measuring machines, were first produced using lathes. Clockmakers extended their craft to creating mechanical animals and other novelties. Rabbi Judah Loew ben Bezalel of Prague is said to have invented the Golem, a clay man brought to life (1580).

Early in the 17th century, Ren´e Descartes proposed that bodies of animals are nothing more than complex machines. Many other 17th century thinkers offered variations and elaborations of Cartesian mechanism. Thomas Hobbes published Leviathan, containing a material and combinatorial theory of thinking. Blaise Pascal created the second mechanical and first digital calculating machine (1642). Gottfried Leibniz improved Pascal's machine, making the Stepped Reckoner to do multiplication and division (1673) and evisioned a universal calculus of reasoning (Alphabet of human thought) by which arguments could be decided mechanically.

The 18th century saw a profusion of mechanical toys, including the celebrated mechanical duck of Jacques de Vaucanson and Wolfgang von Kempelen's phony chessplaying automaton, The Turk (1769).

Mary Shelley published the story of Frankenstein; or the Modern Prometheus (1818).

19th and Early 20th Century

George Boole developed a binary algebra (Boolean algebra) representing (some) l·laws of thought.Çharles Babbage and Ada Lovelace worked on programmable mechanical calculating machines.

In the first years of the 20th century Bertrand Russell and Alfred North Whitehead published Principia Mathematica, which revolutionized formal logic. Russell, Ludwig Wittgenstein, and Rudolf Carnap lead philosophy into logical analysis of knowledge. Karel Capek's play R.U.R. (Rossum's Universal Robots)) opens in London (1923). This is the first use of the word "robotïn English.

Mid 20th century and Early AI

Warren Sturgis McCulloch and Walter Pitts publish "A Logical Calculus of the Ideas Immanent in Nervous Activity"(1943), laying foundations for artificial neural networks. Arturo Rosenblueth, Norbert Wiener and Julian Bigelow coin the term çybernetics̈in a 1943 paper. Wiener's popular book by that name published in 1948. Vannevar Bush published As We May Think (The Atlantic Monthly, July 1945) a prescient vision of the future in which computers assist humans in many activities.

The man widely acknowledged as the father of computer science, Alan Turing, published Çomputing Machinery and Intelligence"(1950) which introduced the Turing test as a way of operationalizing a test of intelligent behavior. Claude Shannon published a detailed analysis of chess playing as search (1950). Isaac Asimov published his Three Laws of Robotics (1950).

1956: John McCarthy coined the term "artificial intelligence"as the topic of the Dartmouth Conference, the first conference devoted to the subject. Demonstration of the first running AI program, the Logic Theorist (LT) written by Allen Newell, J.C. Shaw and Herbert Simon (Carnegie Institute of Technology, now Carnegie Mellon University).

1957: The General Problem Solver (GPS) demonstrated by Newell, Shaw and Simon.

1952-1962: Arthur Samuel (IBM) wrote the first game-playing program, for checkers (draughts), to achieve sufficient skill to challenge a world champion. Samuel's machine learning programs were responsible for the high performance of the checkers player.

1958: John McCarthy (Massachusetts Institute of Technology or MIT) invented the Lisp programming language. Herb Gelernter and Nathan Rochester (IBM) described a theorem prover in geometry that exploits a semantic model of the domain in the form of diagrams of "typicalçases. Teddington Conference on the Mechanization of Thought Processes was held in the UK and among the papers presented were John McCarthy's Programs with Common Sense, Oliver Selfridge's Pandemonium, and Marvin Minsky's Some Methods of Heuristic Programming and Artificial Intelligence.

Late 1950s and early 1960s: Margaret Masterman and colleagues at University of Cambridge design semantic nets for machine translation.

1961: James Slagle (PhD dissertation, MIT) wrote (in Lisp) the first symbolic integration program, SAINT, which solved calculus problems at the college freshman level.

1962: First industrial robot company, Unimation, founded.

1963: Thomas Evans' program, ANALOGY, written as part of his PhD work at MIT, demonstrated that computers can solve the same analogy problems as are given on IQ tests. Edward Feigenbaum and Julian Feldman published Computers and Thought, the first collection of articles about artificial intelligence.

1964: Danny Bobrow's dissertation at MIT (technical report from MIT's AI group, Project MAC), shows that computers can understand natural language well enough to solve algebra word problems correctly. Bert Raphael's MIT dissertation on the SIR program demonstrates the power of a logical representation of knowledge for question-answering systems.

1965: J. Alan Robinson invented a mechanical proof procedure, the Resolution Method, which allowed programs to work efficiently with formal logic as a

representation language. Joseph Weizenbaum (MIT) built ELIZA (program), an interactive program that carries on a dialogue in English language on any topic. It was a popular toy at AI centers on the ARPANET when a version that "simulated"the dialogue of a psychotherapist was programmed.

1966: Ross Quillian (PhD dissertation, Carnegie Inst. of Technology, now CMU) demonstrated semantic nets. First Machine Intelligence workshop at Edinburgh: the first of an influential annual series organized by Donald Michie and others. Negative report on machine translation kills much work in Natural language processing (NLP) for many years.

1967: Dendral program (Edward Feigenbaum, Joshua Lederberg, Bruce Buchanan, Georgia Sutherland at Stanford University) demonstrated to interpret mass spectra on organic chemical compounds. First successful knowledge-based program for scientific reasoning. Joel Moses (PhD work at MIT) demonstrated the power of symbolic reasoning for integration problems in the Macsyma program. First successful knowledge-based program in mathematics. Richard Greenblatt (programmer) at MIT built a knowledge-based chess-playing program, MacHack, that was good enough to achieve a class-C rating in tournament play.

1968: Marvin Minsky and Seymour Papert publish Perceptrons, demonstrating limits of simple neural nets.

1969: Stanford Research Institute (SRI): Shakey the Robot, demonstrated combining animal locomotion, perception and problem solving. Roger Schank (Stanford) defined conceptual dependency model for natural language understanding. Later developed (in PhD dissertations at Yale University) for use in story understanding by Robert Wilensky and Wendy Lehnert, and for use in understanding memory by Janet Kolodner. Yorick Wilks (Stanford) developed the semantic coherence view of language called Preference Semantics, embodied in the first semantics-driven machine translation program, and the basis of many PhD dissertations since such as Bran Boguraev and David Carter at Cambridge. First International Joint Conference on Artificial Intelligence (IJCAI) held at Stanford.

1970: Jaime Carbonell (Sr.) developed SCHOLAR, an interactive program for computer assisted instruction based on semantic nets as the representation of knowledge. Bill Woods described Augmented Transition Networks (ATN's) as a representation for natural language understanding. Patrick Winston's PhD program, ARCH, at MIT learned concepts from examples in the world of children's blocks. Early 70's: Jane Robinson and Don Walker established an influential Natural Language Processing group at SRI.

1971: Terry Winograd's PhD thesis (MIT) demonstrated the ability of computers to understand English sentences in a restricted world of children's blocks,

in a coupling of his language understanding program, SHRDLU, with a robot arm that carried out instructions typed in English.

1972: Prolog programming language developed by Alain Colmerauer.

1973: The Assembly Robotics Group at University of Edinburgh builds Freddy Robot, capable of using visual perception to locate and assemble models. The Lighthill report gives a largely negative verdict on AI research in Great Britain and forms the basis for the decision by the British government to discontine support for AI research in all but two universities.

1974: Ted Shortliffe's PhD dissertation on the MYCIN program (Stanford) demonstrated the power of rule-based systems for knowledge representation and inference in the domain of medical diagnosis and therapy. Sometimes called the first expert system. Earl Sacerdoti developed one of the first planning programs, ABSTRIPS, and developed techniques of hierarchical planning.

1975: Marvin Minsky published his widely-read and influential article on Frames as a representation of knowledge, in which many ideas about schemas and semantic links are brought together. The Meta-Dendral learning program produced new results in chemistry (some rules of mass spectrometry) the first scientific discoveries by a computer to be published in a referreed journal.

Mid 70's: Barbara Grosz (SRI) established limits to traditional AI approaches to discourse modeling. Subsequent work by Grosz, BonnieWebber and Candace Sidner developed the notion of çentering", used in establishing focus of discourse and anaphoric references in NLP. David Marr and MIT colleagues describe the "primal sketch"and its role in visual perception.

1976: Douglas Lenat's AM program (Stanford PhD dissertation) demonstrated the discovery model (loosely-guided search for interesting conjectures). Randall Davis demonstrated the power of meta-level reasoning in his PhD dissertation at Stanford.

Late 70's: Stanford's SUMEX-AIM resource, headed by Ed Feigenbaum and Joshua Lederberg, demonstrates the power of the ARPAnet for scientific collaboration.

1978: Tom Mitchell, at Stanford, invented the concept of Version Spaces for describing the search space of a concept formation program. Herbert Simon wins the Nobel Prize in Economics for his theory of bounded rationality, one of the cornerstones of AI known as "satisficing". The MOLGEN program, written at Stanford by Mark Stefik and Peter Friedland, demonstrated that an object-oriented programming representation of knowledge can be used to plan gene-cloning experiments.

1979: Bill VanMelle's PhD dissertation at Stanford demonstrated the generality of MYCIN's representation of knowledge and style of reasoning in his EMYCIN program, the model for many commercial expert system "shells". Jack Myers and Harry Pople at University of Pittsburgh developed INTERNIST, a knowledge-based medical diagnosis program based on Dr. Myers' clinical knowledge. Cordell Green, David Barstow, Elaine Kant and others at Stanford demonstrated the CHI system for automatic programming. The Stanford Cart, built by Hans Moravec, becomes the first computer-controlled, autonomous vehicle when it successfully traverses a chair-filled room and circumnavigates the Stanford AI Lab. Drew McDermott and Jon Doyle at MIT, and John McCarthy at Stanford begin publishing work on nonmonotonic logics and formal aspects of truth maintenance.

1980s: Lisp machines developed and marketed. First expert system shells and commercial applications.

1980: Lee Erman, Rick Hayes-Roth, Victor Lesser and Raj Reddy published the first description of the blackboard model, as the framework for the HEARSAY-II speech understanding system. First National Conference of the American Association for Artificial Intelligence (AAAI) held at Stanford.

1981: Danny Hillis designs the connection machine, a massively parallel architecture that brings new power to AI, and to computation in general. (Later founds Thinking Machines, Inc.)

1982: The Fifth Generation Computer Systems project (FGCS), an initiative by Japan's Ministry of International Trade and Industry, begun in 1982, to create a "fifth generation computer" (see history of computing hardware) which was supposed to perform much calculation utilizing massive parallelism.

1983: John Laird and Paul Rosenbloom, working with Allen Newell, complete CMU dissertations on Soar (program). James F. Allen invents the Interval Calculus, the first widely used formalization of temporal events.

Mid 80's: Neural Networks become widely used with the Backpropagation algorithm (first described by Paul Werbos in 1974).

1985: The autonomous drawing program, AARON, created by Harold Cohen, is demonstrated at the AAAI National Conference (based on more than a decade of work, and with subsequent work showing major developments).

1987: Marvin Minsky publishes The Society of Mind, a theoretical description of the mind as a collection of cooperating agents.

1989: Dean Pomerleau at CMU creates ALVINN (An Autonomous Land Vehicle in a Neural Network), which grew into the system that drove a car

coast-to-coast under computer control for all but about 50 of the 2850 miles.

1990s: Major advances in all areas of AI, with significant demonstrations in machine learning, intelligent tutoring, case-based reasoning, multi-agent planning, scheduling, uncertain reasoning, data mining, natural language understanding and translation, vision, virtual reality, games, and other topics. Rodney Brooks' MIT Cog project, with numerous collaborators, makes significant progress in building a humanoid robot.

Early 90's: TD-Gammon, a backgammon program written by Gerry Tesauro, demonstrates that reinforcement (learning) is powerful enough to create a championshiplevel game-playing program by competing favorably with world-class players.

1997: The Deep Blue chess program (IBM) beats the world chess champion, Garry Kasparov, in a widely followed match. First official RoboCup football (soccer) match featuring table-top matches with 40 teams of interacting robots and over 5000 spectators.

1998: Tim Berners-Lee published his Semantic Web Road map paper [2]. Late 90's: Web crawlers and other AI-based information extraction programs become essential in widespread use of the World Wide Web. Demonstration of an Intelligent room and Emotional Agents at MIT's AI Lab. Initiation of work on the Oxygen architecture, which connects mobile and stationary computers in an adaptive network.

2000: Interactive robopets ("smart toys") become commercially available, realizing the vision of the 18th century novelty toy makers. Cynthia Breazeal at MIT publishes her dissertation on Sociable machines, describing Kismet (robot), with a face that expresses emotions. The Nomad robot explores remote regions of Antarctica looking for meteorite samples.

2004: OWL Web Ontology Language W3C Recommendation (10 February 2004).

9.5 ***Appendix F Apropat a la Ciència***

9.5.1 Apropa't a la ciència.De la Recerca a la Innovació

Figura 35. Apropat a la Ciència event (2006/2007,Barcelona)

The followings information is provided directly from the official website: http://www10.gencat.net/probert/catala/exposicio/ex14_ciencia.htm

Apropa't a la ciència.De la Recerca a la Innovació (fig.35)

Inauguració: 11 d'octubre de 2006 a les 19 h Oberta al públic del 12 d'octubre al 31 de juliol de 2007 L'exposició Com un exponent destacat del Pla de Recerca i Innovació 2005 - 2008 de la Generalitat de Catalunya, i coincidint amb Barcelona Ciència 2007, "Apropa't a la ciència"pretén acostar d'una manera didàctica i atractiva la ciència als ciutadans; entesa aquesta en un sentit ampli. És a dir, com una eina útil per a la gent, malgrat el desconeixement de molts dels seus aspectes, però de la qual se'n deriven evidents repercussions socials i de millora per a la qualitat de vida, és a dir el RETORN SOCIAL DE LA CIÈNCIA. També està adreçada a fomentar l'interés dels més joves

cap a aquesta disciplina.

Any de la ciència Sota el títol de Barcelona Ciència 2007 l'ajuntament de Barcelona commemora el centenari del premi Nobel atorgat a Santiago Ramon y Cajal. Engega un ampli programa cultural que posarà un accent especial en el vincle entre les ciències, la cultura i la societat. A més a més, el 2007 es commemora també el centenari de la creació de l'Institut d'Estudis Catalans i el de la Junta de Ampliación de Estudios, institució precursora del Consejo Superior de Investigaciones Científicas. Per tal de donar un impuls decisiu a la política en relació amb la promoció de la cultura científica i crear una definitiva major sensibilitat social i cultural cap a la ciència, aquest programa cultural anuncia entre les seves línies concretes d'acció la proposta que l'any 2007 sigui declarat "Any de la Ciència"a la ciutat de Barcelona. website: http://www.bcn.es/ciencia2007/

L'exposició Apropa´t a la Ciència. De la recerca a la innovació ha estat dissenyada per donar a conèixer els plans de recerca i desenvolupament impulsats per la Generalitat de Catalunya per tal de fer veure la relació entre els coneixements científics i la innovació tecnològica i promoure vocacions científiques. Se'ns ofereix en l'exposició amb exemples ben concrets i molt suggerents. En ella no se'ns parla de res que recordi el contingut dels llibres de text (potser perquè es reconeix de manera implícita que, si ho fes, seria difícil l'apropament del públic que es vol aconseguir), la seva finalitat no és fer comprendre conceptes teòrics, fórmules ni equacions. Se'ns presenta, en canvi, una àmplia panoràmica de l'activitat científica real, la que es produeix en diversos contextos i impregna la vida de totes les persones. Per això, per la novetat que representa i per les noves possibilitats educatives que ofereix, és molt important donar aquest nou significat, d'empresa col·lectiva, a la paraula 'ciència'.

Anem del passat més remot, de quan encara no hi havia humans sobre la Terra fins a l'avenir incert dels viatges espacials i dels robots. En Pau d'Hostalets de Pierola ens proporciona una ocasió per a pensar en el lent procés que ha donat lloc a l'emergència de l'espècie humana (una evolució afortunada de la clavícula que proporciona noves possibilitats de manipulació) i la cadira Mares ens fa veure les dificultats d'adaptar un cos que ha de moure's a la inactivitat forçosa en l'interior d'una nau espacial. Els robots ens fan pensar en quines tasques faran en lloc nostre i com fer-les o no fer-les podrà afectar les nostres pròpies capacitats. S'han ampliat les comunicacions, que connecten els satèl·lits artificials, tan llunyans, amb els mòbils, tan propers. Les intervencions humanes en el món són ara d'abast planetari i transformen la Natura, perquè en són part. Les tecnologies per a l'aprofitament de l'energia (els molins de vent), per a la conservació dels aliments, per a prevenir malalties o superar-ne d'incurables (les vacunes i els trasplantaments), han de poder arribar a tot arreu.. La mostra ocupa físicament la sala 3 del Palau Robert dividida en vuit àmbits, a més dels escenaris un d'entrada i un de sortida, en el primer

dels quals es fa una ràpida pinzellada a les aportacions realitzades pels grans científics de la història com ara Newton o Curie, entre molts d'altres. Aquesta introducció inicial remet al darrer dels apartats de tancament de l'exposició, tot invitant les generacions futures a prendre el relleu científic.

9.6 La robòtica i les seves aplicacions socials

Un futur de persones i robots. Tot i que per a la majoria de persones els robots són una realitat confinada a les fàbriques i a la producció, el futur de la robòtica no passa per les naus industrials. Si l'any 2000 els robots industrials representaven el 95mercat global de la robòtica, es calcula que durant el 2025 no passaran de ser el 20% d'aquest mercat (situat als 65.000 milions de dòlars), perdent el seu lideratge en favor dels robots personals i de serveis. Els seus usos es distribuiran en tres grans apartats: - En el sector professional, des de l'agricultura a la cirurgia passant pel transport i la construcció. L'aplicació de la robòtica en aquest sector, en el que Europa manté un cert lideratge, constitueix l'evolució natural de la robòtica tradicional i ha de permetre automatitzar processos fins ara exclusius dels humans durant la seva vida professional. - En el sector domèstic, des de la neteja de la casa fins a la cura de la gent gran o dels malalts. Aquestes aplicacions, liderades actualment pels EUA i Corea, faran canviar la imatge del robot amb forma de "braç mecànic"que munta el vidre d'un cotxe a la línia de producció per un robot, en alguns casos antropomòrfic, que conviu i actua a l'entorn domèstic de les persones fent tasques de suport a les persones. - El sector de l'oci i l'entreteniment, des de les joguines robotitzades fins als entrenadors personals d'algun esport. Aquest sector, liderat pel Japó, és ja una realitat amb una gran capacitat de creixement que pot fins i tot superar als jocs d'ordinador.

A l'exposició veurem el robot AIBO i les possibilitats que un programari especial té per a les persones amb discapacitats, persones grans, etc.

Figura 36. Fira