Universitat d'Alacant
Universidad de Alicante

Instituto Universitario de Investigación Informática

Educational framework using robots with vision for constructivist teaching Robotics to pre-university students

Entorno educativo usando robots con visión para la enseñanza constructivista de Robótica a estudiantes preuniversitarios

Julio Vega Pérez

TESIS PRESENTADA PARA ASPIRAR AL GRADO DE DOCTOR POR LA UNIVERSIDAD DE ALICANTE

MENCIÓN DE DOCTOR INTERNACIONAL

PROGRAMA DE DOCTORADO EN INFORMÁTICA

Dirigida por:

Dr. José María Cañas Plaza

Dr. Miguel Ángel Cazorla Quevedo

2018

The thesis presented in this document has been
reviewed and approved for the

INTERNATIONAL PHILOSOPHY DOCTOR (Ph.D.)
HONOURABLE MENTION

I would like to thank the advises and contributions for this thesis of the
external reviewers:

Dr. Antonio J. Rodríguez-Sánchez
(University of Innsbruck)

Dr. José Carlos Rangel Ortiz
(Universidad Tecnológica de Panamá)

A mi familia
(biológica y
política).

A person who does not know
the history of the last 3,000 years
wanders in the darkness of ignorance,
unable to make sense of the reality around him.

El que no sabe llevar su contabilidad
por espacio de tres mil años
se queda como un ignorante en la oscuridad
y sólo vive al día.

Göethe

Agradecimientos

Desde que comenzara esta travesía han pasado ya diez años, durante los cuales son muchas las personas que han aportado su granito de arena para que finalmente pueda hacer cumbre.

Es por ello que dedique, en primer lugar, mis más sinceros agradecimientos a José María, director de esta tesis, por su paciencia en los altibajos sufridos por el camino, su admirable dedicación y —en definitiva— por todo el esfuerzo y tiempo que ha dedicado para que lleguemos a presentar el presente trabajo. También he de agradecer a mi otro director, Miguel, su inestimable ayuda en esta última etapa del trayecto, con todo el papeleo que ello conlleva, sus consejos y realimentaciones respecto al presente documento, y su capacidad para enfocar de forma panorámica todos los esfuerzos, organizarlos y canalizarlos en esta disertación.

Por otro lado, el apoyo incondicional y el ánimo constante de la familia y amigos ha sido vital para superar los momentos más difíciles que se han ido sucediendo durante estos años. Y, sobretodo, agradecer a Águeda su comprensión y paciencia, pues ha sido quien lo ha *sufrido* más de cerca. También merecen su hueco aquí quienes han hecho las veces de familiares cuando *uno* está alejado de los suyos. Antonio y Margarita, que me acogieron como a un hijo en el lejano Toronto; y John, mi director de allí, que fue todo un gran anfitrión. De igual forma, Sampo, que me dedicó todo su tiempo durante mi estancia en Joensuu para que conociera profundamente el sistema educativo finlandés.

Parte de este trabajo también se lo debo al equipo de desarrollo de `JdeRobot`. Sus contribuciones, apoyo y respaldo en las situaciones que parecen no tener salida han supuesto una fuente de oxígeno cuando los

desafíos técnicos se ponían más cuesta arriba. Especialmente Edu, que fue mi compañero de aventuras durante gran parte del camino; así como Aitor y Fran, quienes me han ido allanando los terrenos más abruptos que se han ido sucediendo en las últimas etapas.

Y ya, para acabar, sería injusto no mencionar a algunos de los compositores musicales que me han acompañado durante las incalculables horas pasadas frente al ordenador. Por citar algunos: S. Rós, L. Einaudi, M. Richter, O. Arnalds y J. Jóhannsson. Sus creaciones artísticas han amenizado toda la caminata y la han hecho menos solitaria.

A todos, gracias.

Alicante, 24 de julio de 2018
Julio

Preámbulo

El presente trabajo es el resultado de los estudios doctorales llevados a cabo bajo el Programa de Doctorado en Informática entre los años 2008 y 2012 en el Grupo de Robótica del Departamento de Sistemas Telemáticos y Computación de la Universidad Rey Juan Carlos de Madrid, así como fruto del trabajo compaginado con la experiencia en docencia preuniversitaria adquirida entre los años 2012 y 2018 en distintos centros de Educación Secundaria, para desarrollar finalmente la disertación en el Programa de Doctorado en Informática de la Universidad de Alicante.

La tesis se ha llevado a cabo bajo el marco de diversos proyectos y la financiación de distintas ayudas y becas nacionales y europeas.

- *JdeRobot-Kids: Middleware para la programación de robots y sistemas de visión enfocado a la labor docente preuniversitaria.* Grupo de Robótica de la Universidad Rey Juan Carlos (2017-2018).

- *JdeRobot: Middleware para la programación de robots y sistemas de visión enfocado a la labor docente universitaria.* Grupo de Robótica de la Universidad Rey Juan Carlos. Financiado parcialmente por Google a través del programa *Google Summer of Code* en las ediciones de 2015, 2017 y 2018.

- *RETOGAR (Retorno al hogar): Sistema de mejora de la autonomía de personas con daño cerebral adquirido y dependientes en su integración en la sociedad.* Programa estatal de investigación, desarrollo e innovación orientada a los retos de la sociedad, Ministerio de Economía y Competitividad. Ref: TIN2016-76515-R (2017-2020).

- *RoboCity2030-III: Robots de servicio para la mejora de la calidad de vida de los ciudadanos en áreas metropolitanas.* Programa de Actividades de I+D de la Comunidad de Madrid. Ref: S2013/MIT-2748 (2014-2018).

- *Beca de estancia Erasmus+ en Finlandia: De la tiza al robot.* Proyecto de investigación sobre el modelo educativo de Finlandia durante 2015-2017. Una parte del mismo suponía la estancia en el Centro de Investigación Robótica Joensuu Science Society de la Universidad of Eastern Finland, en colaboración con centros educativos de Joensuu y Helsinki. Financiado por la Comisión Europea bajo el programa Erasmus+, soportado con fondos europeos. Ref: 2015-1-ES01-KA101-014274.

- *RoboCity2030-II: Robots de servicio para la mejora de la calidad de vida de los ciudadanos en áreas metropolitanas.* Programa de Actividades de I+D de la Comunidad de Madrid. Ref: S2009/DPI-1559 (2010-2013).

- *Roboterapia en Demencia.* Ministerio de Ciencia e Innovación. Acción estratégica de Salud. Ref: 10/02567 (2010-2013).

- *Beca de estancia predoctoral en Canadá.* Laboratory for Active and Attentive Vision (LAAV), Dept. of Computer Science & Engineering y en el Centre for Vision Research, York University, Toronto, Canada, bajo la dirección de Dr. John Tsotsos. Programa de Formación PDI de la Universidad Rey Juan Carlos (2009).

- *Beca de formación predoctoral.* Programa de Formación PDI de la Universidad Rey Juan Carlos (2008-2012).

La motivación, guía y respaldo de esta disertación surge por la participación y colaboración en éstos y otros proyectos previos relacionados con la robótica móvil y los algoritmos de visión. Asimismo, el enfoque práctico en docencia está motivado por la vocación personal y la experiencia adquirida durante estos años.

Abstract

Robotics will be a dominant area in society throughout future generations. Nowadays its presence is increasing in the majority of contexts of daily life, with devices and mechanisms which facilitate the accomplishment of diverse daily tasks; as well as at labor level, where machines occupy more and more jobs.

This increase in the presence of autonomous robotic systems in society is due to the great efficiency and security they offer compared to human capacity, thanks mainly to the enormous precision of their sensor and actuator systems. Among these, vision sensors are of utmost importance. Humans and many animals enjoy powerful perception systems in a natural way, but which in Robotics constitutes a constant line of research. The main problem lies in the correct interpretation of visual data and the extraction of relevant information from camera images.

Thus, Robotics becomes something beyond an scientific are, but also a social and cultural topic. Therefore, it is essential to raise an early awareness and train younger students to acquire the skills which will be most demanded in the short and mid-term future. In doing so, we will be ensuring their integration into a labor market dominated by intelligent robotic systems. In addition to having a high capacity for reasoning and decision-making, these robots incorporate important advances in their perceptual systems, allowing them to interact effectively in the working environments of this new industrial revolution.

Since a few years ago, there are different Educational Robotics kits available in the market which are designed to be used in pre-university education. To use them as a learning tool, a correct teacher training is

necessary, as well as a change in the teaching-learning methodology and in the educational environment in general. In addition, taking into account that young people live immersed in a constant environment of technological learning, most of these kits usually have a short period of interest for students, who demand motivating intellectual challenges.

This thesis aims to provide several solutions to some classic problems inherent to Robotics, such as navigation and localization, but using a camera as the main sensor. In addition, a learning framework for teaching of Robotics with Vision as a subject is presented. Using it the students at pre-university curricular level learn the principles of Science and Engineering and the computer programming skills demanded in today's society. The use of Python language and its exercises about robots with vision makes this learning framework unique and more powerful than other existing frameworks.

This teaching framework has been successfully used in several secondary education schools during the last two academic years (2016/2017 and 2017/2018), which includes: its software infrastructure, its hardware platform, an academic curriculum with theoretical and practical content, as well as a constructivist pedagogical methodology. The performance and satisfaction of more than $2,000$ students and teachers using it, in curricular subjects such as *Programming, Robotics and Technology* and *ICTs* of Secondary Education (CSO) and extracurricular activities, have been evaluated.

Resumen

La Robótica será un área dominante en la sociedad de las próximas generaciones. Actualmente su presencia es cada vez mayor a nivel doméstico, con dispositivos y mecanismos que facilitan la realización de diversas tareas cotidianas; así como a nivel laboral, donde las máquinas van ocupando cada vez más puestos de trabajo.

Este aumento en la presencia de sistemas robóticos autónomos en la sociedad es debido a la gran eficiencia y seguridad que ofrecen frente a la capacidad humana, gracias fundamentalmente a la enorme precisión de sus sistemas de sensores y actuadores. Entre estos sensores adquiere especial importancia el de visión; algo que los humanos y los animales tienen muy desarrollado, pero que en Robótica constituye un frente de investigación en constante desarrollo. El principal problema reside en la correcta interpretación de los datos visuales y la extracción de información relevante a partir de las imágenes.

La Robótica se convierte así en algo más que un área de la ciencia, sino también en una cuestión social y cultural. Por ello es fundamental crear una conciencia temprana y formar a los estudiantes más jóvenes en adquirir las habilidades que serán más demandadas en el futuro a corto y medio plazo, garantizando así su integración en un mercado laboral dominado por sistemas robóticos inteligentes. Además de tener una elevada capacidad de razonamiento y de toma de decisiones, estos robots incorporan importantes avances en sus sistemas perceptivos, lo que les permite interactuar inteligentemente en los entornos laborales de esta nueva revolución industrial.

Desde hace unos años existen en el mercado diferentes kits de Robótica Educativa para su uso en enseñanza preuniversitaria. Para usarlos como herramienta de aprendizaje es necesaria también una correcta formación del profesorado, un cambio en la metodología de enseñanza-aprendizaje y en el entorno educativo en general. Además, teniendo en cuenta que los más jóvenes viven inmersos en un constante entorno de aprendizaje tecnológico, la mayoría de estos kits suelen tener un corto periodo de interés para los estudiantes, que demandan desafíos intelectuales que les resulten motivadores.

En esta tesis se proponen varias soluciones a problemas clásicos de la Robótica, como la navegación y la localización, pero empleando una cámara como sensor principal. Asimismo, se presenta un entorno para la docencia de Robótica con Visión como materia a través de la cual los alumnos de niveles académicos preuniversitarios aprenden los principios de la ciencia y la ingeniería y las habilidades de programación de ordenadores tan demandadas por la sociedad actual. El lenguaje Python elegido y los ejercicios de robots con visión, suponen un aporte novedoso único y ventajoso, más potente que otros entornos docentes existentes.

Este entorno docente se ha implantado con éxito en varios centros educativos durante los dos últimos cursos académicos (2016/2017 y 2017/2018), incluyendo: infraestructura software, plataforma hardware, currículum académico con contenido teórico y práctico, así como la metodología pedagógica constructivista. Se ha evaluado el rendimiento y la satisfacción de más de 2,000 estudiantes y profesores usando este entorno docente en las asignaturas curriculares de *Tecnología, Programación y Robótica* y *TICs* de Educación Secundaria (ESO), así como en actividades extraescolares.

Acronyms

List of acronyms used in this document shown in alphabetical order:

AERO *Autonomous Exploration Rover*

AI *Artificial Intelligence*

ANN *Artificial Neural Network*

API *Application Programming Interface*

AR *Assistive Robot*

ATM *Automated Teller Machine*

CSO *Compulsory Secondary Education*

EKF *Extended Kalman Filter*

FLL *First Lego League*

FOA *Focus of Attention*

GA *Genetic Algorithm*

GPIO *General Purpose Input/Output*

GPS *Global Positioning System*

HCI *Human-Computer Interaction*

HRI *Human-Robot Interaction*

ICT *Information and Communication Technology*

IDE *Integrated Development Environment*

IOM *Instant Obstacle Map*

ISS *International Space Station*

KF *Kalman Filter*

MCL *Monte Carlo Localization*

MER *Mars Exploration Rover*

NASA *National Aeronautics and Space Administration*

NXT *LEGO®Mindstorms®NXT*

NXT-G *LEGO®Mindstorms®NXT Graphical development environment*

PBL *Problem-Based Learning*

PCL *Point Cloud Library*

PID *Proportional Integral Derivative*

PISA *Programme for International Student Assessment*

RCJ *RoboCup Junior*

RIE *Robotics in Education*

ROS *Robot Operating System*

RVW *Robot Virtual Worlds*

SAR *Socially Assistive Robot*

SIR *Socially Interactive Robot*

SLAM *Simultaneous Localization And Mapping*

STEM *Science, Technology, Engineering and Mathematics*

TRROS *Teaching Robotics with ROS*

T-L *Teaching-Learning process*

Contents

List of Figures

List of Algorithms

List of Equations

List of Tables

Chapter 1

Introduction

This first chapter introduces the main topic of this thesis. It is organized according to the following sections. Section 1.1 shows a general description of the situation of the Robotics area, its role in the current society and the near future and how the vision of robots is fundamental in this new era of smart robots. Section 1.3 provides a brief analysis of the worldwide boom in robotics and educational programming. In Section 1.4 the educational proposal developed in this dissertation is presented. Section 1.5 describes the main objectives of this work. And, finally, in Section 1.6 a tour is made of the structure according to which this thesis is vertebrated.

1.1 Technology and Robotics

In the last decade, technology has become increasingly common in the majority of contexts of daily and industrial life. In homes, there has been a major increase in the presence of technological devices, such as computers, *tablets*, *smartphones*, domotic systems, etc. They are all interconnected through Internet. At industrial level, rather than autonomous machines, factories are increasingly incorporating in their production chains intelligent robots with sophisticated sensory systems, with vision as the main mechanism of perception.

1.1.1 Robotics in the domestic field

A home usually has numerous technological elements; the appearance of robotic devices in the mass market such as robotic vacuum cleaners and mops (Figures 1.1 left and middle), as well as numerous applications and existing domotic services (Figure 1.1 right) have made this technology increasingly present in the daily routine of society, not to mention other frequently automated tasks: withdrawing money at the ATM, automatic payment in supermarkets, or the massive use of Internet, shopping, banking, and much more.

Figure 1.1: iRobot Roomba And Jet Braava, and domotic application Wattio

Furthermore, autonomous cars or drones make the use of this technology more visible and reinforce its appeal. The large automotive manufacturers are behind these new advances; they have advanced prototypes of autonomous cars. Also, large software companies such as Google (Figure 1.2) or Apple, or new companies like Tesla are well positioned in this sector.

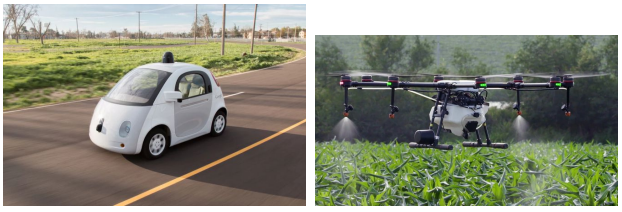


Figure 1.2: Google autonomous car and a drone devoted to agricultural tasks

However, the use of drones for leisure activities has remained in the background, giving way to their increasingly widespread use at professional level (Figure 1.2) for works in the fields of emergency, events, people searches, fiscal control, border surveillance, agriculture, traffic surveillance,

etc. In Spain there are already almost 3,000 companies working in this sector and the Ministry for Public Works has recently promoted the Strategic Plan for the use of drones 2018-2021 ([Fomento, 2018]) for the development of the civil drone sector, which establishes the roadmap to be followed to promote the development of this incipient and potentially high growth sector.

Because of this tendency towards the automation of almost all daily tasks as well as an increase in the presence of robotic devices in the day to day, knowledge of the use of the technologies is becoming increasingly key.

1.1.2 Robotics and Industrialization

Robotics at industrial level has its origins in the Industrial Revolution of 1800 when, for the first time, products and services were developed by machines. The flagship innovation was the steam engine, which was used to replace a large number of jobs.

The second industrial revolution began with electricity, at the end of the 19th century. The main new concept here was the assembly line, used for the first time in the car industry (Figure 1.3 left).

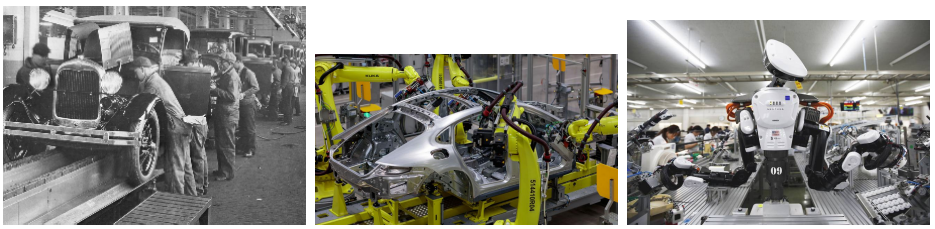


Figure 1.3: Industrial revolutions: (a) Ford assembly line, (b) Porsche serial production, (c) Intelligent robots at Glory Ltd.

The third industrial revolution began in 1970s and was characterized by greater automation through electronics. The first personal computers, Internet and global access to information were incorporated into society. At labor level, human work is replaced by machines —which are programmed to mass manufacture products— (Figure 1.3 middle), where speed, precision and reliability are paramount.

The so-called *Industrialization 4.0* involves the integration of complex

robotic systems in factories (Figure 1.3 right), logistics and what is known as the *Internet of things*, where sophisticated automatons handle an immense quantity of data to take strategic decisions for companies.

The short and mid-term future is/will be marked by industrial production dominated by intelligent machines. The presence of humans in these *intelligent factories* tends to be increasingly reduced and will eventually be symbolic and sporadic. There is no doubt that a machine's capacity for taking optimum decisions in real time and simultaneously handling an enormous quantity of data, is far greater than that of a human being.

1.1.3 Intelligent robots in complex surroundings

These mobile and intelligent robots need, in addition to a large computational capacity, a complex sensory system to *act* intelligently not only in factories but in robot-human interaction at general level ([Vega and Cañas, 2009]). The fixed automation of structured production chains is giving way to an unpredictable world and a totally unstructured *reality* which makes evident the need for a wide complementary range of sensors and actuators to attain complete autonomy ([Arbel and Ferrie, 2001]).

This struggle for autonomy in complex, unstructured and unpredictable surroundings, cohabited by humans, constitutes at present a deep field of investigation in Robotics, *Intelligent Robotics*, where visual perception, reasoning and performance are intimately entwined to perform useful tasks with little human intervention. At this point, it is worth mentioning Duplex, the intelligent assistant that Google presented in May of this year, which is an accurate reflection of how machines are evolving.

1.2 Vision in Robotics

Computational vision is the branch of Artificial Intelligence (AI) that includes the techniques and methods applied to a camera-like sensor. Although this sensory modality has not been the most used for some years in mobile robotics (sonar and/or laser have been more used as sensors, Figures 1.4 left and middle), at present it has become the most widely used sensor and will definitely be the most commonly used in the long-term

future, because of the possibilities it offers and the power of calculation of current computers. They are low-cost devices which are potentially very computationally rich, since they provide a lot of information.

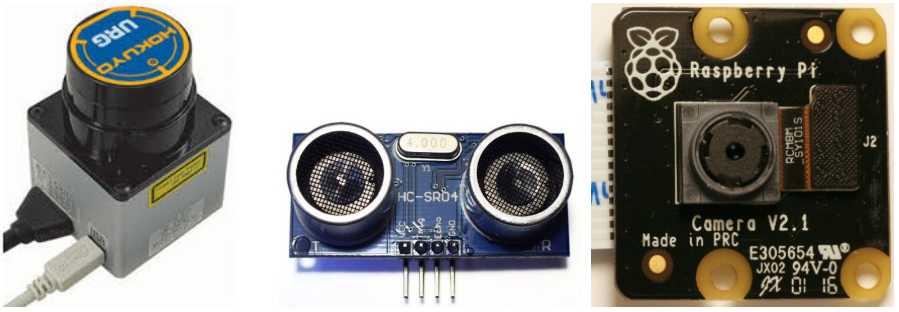


Figure 1.4: Hokuyo laser sensor, HC-SR04 ultrasonic sensor and PiCam camera

However, visual capacity in robots, in contrast to that of living beings, is not an easy technique ([Ramachandran, 1990]). The main difficulty lies in extracting useful information from the large amount of data that a camera provides (Figure 1.4 right), for which good algorithms are needed. Although we have to be wary when comparing a robot with a biological organism ([Nehmzow, 1993]), what is clear is that the sight is the main sense on which animals base their movement through the surroundings ([Tinbergen, 1951]).

Figure 1.5: Advanced systems of Visual perception: Bumblebee, Mobile Ranger and Kinect

Although in the market there are various devices of perception that combine vision and infrared to obtain a greater amount of useful information of the surroundings without the need for large data processing (Figure 1.5), in this work we use simple cameras, focusing the efforts on developing complex algorithms able to extract information of interest from the data they provide.

In the literature, there are several open lines of investigation regarding robot vision. Below, some are described.

Detection of stimuli through neural networks

In recent decades, new models of artificial neural networks (ANN) have emerged, which are posited on diverse theories of the operation of biological neural networks. Numerous works have been proposed, applying ANNs in different areas of engineering. An area where they have been widely used is that of processing images, where a large number of works using ANNs exist ([Schmidhuber, 2015]).

For example, autonomous cars apply neural networks techniques to manage the paths and the controls of position and orientation ([Caceres et al., 2017]).

Visual control

Another widespread use of the vision of robots is visual control, detection of faces ([Vega and Cañas, 2009]), video surveillance systems ([Srinivasan et al., 2006]), traffic control (cameras with seat belt detection systems, etc.), and in autonomous cars to detect lane change ([Cho et al., 2014]). There is also an investigation area in Biometrics, for recognition of signatures, characters and the study of writing traces([Pinto et al., 2015]).

3D Reconstruction

The acquisition of three-dimensional models has for some years been one of the research challenges with the greatest activity. The use of these 3D stages is very wide-ranging, from developing surroundings of virtual reality ([Barrera et al., 2005]) to video games ([Richter et al., 2016]), as well as in the reconstruction of ancient ([Murgul, 2015]) or modern buildings ([Fathi et al., 2015]) for their analysis.

Visual self-localization

Visual, or image-based, self-localization refers to the recovery of the position and orientation of a camera in the world according to recorded

images. This is an interesting application in environments where GPS-based systems are not available or are imprecise, such as indoors or in densely built cities ([Antequera et al., 2017]).

Applications that conduct their tasks in indoor places, such as *Roomba* by *iRobot*, use these techniques to know at all times their position in the surroundings in which they are performing their tasks and to avoid, for example, unintelligent behaviors like repeating in zones in which they have already been, etc.

Furthermore, autonomous cars include visual self-localization techniques ([Wolcott and Eustice, 2014]), which allows them to navigate securely in zones or moments where the reception of GPS satellites is poor, such as cities with high buildings, tunnels or at moments when bad weather renders it impossible to visualize these satellites.

Visual attention

When processing and selecting useful details from among the vast amount of information perceived by a camera, it is crucial to look at how the visual systems of organisms in nature work ([Zaharescu et al., 2005]). Humans have a precise system of active vision ([Bajcsy, 2009, Sawides et al., 2018]). This means that we can concentrate on specific regions of interest of the scene that surrounds us ([Marocco and Floreano, 2002]) thanks to the movement of our eyes ([Murray et al., 2003]) and/or of our head ([Vega and Cañas, 2009]), or simply by extending our gaze ([Arbel and Ferrie, 2001]) to several zones within the current image being perceived ([Itti and Koch., 2005]).

Visual attention is a key task in autonomous Robotics, since a robot with vision that interacts in a real environment has to be reactive. It has to include rapid vision systems that extract useful information from camera data in real time.

Visual memory

If a robot with vision, in addition to treating the information it receives in each moment from its visual system, can store this in a memory, it will

be able to take more intelligent decisions when it has to navigate through the environment. Hence, in recent years, several works have focused on this research challenge.

Autonomous cars, domestic applications like *Roomba*, or autonomous aerial navigation systems incorporate this memory.

As regards the subject of the thesis, some methods have been developed on what and where the objects are that a robot finds in its path ([Zaharescu et al., 2005]), which will allow the robot to navigate the surrounding environment intelligently and be located in it.

1.3 Educational robotics

As described in Section 1.1.2, the advance of Artificial Intelligence (AI), Robotics and automation in society, the future of work and industry in particular ([Mies and Zentay, 2017]) converge in what is already known as the fourth industrial revolution ([Schwab, 2016]).

According to the analysis of the University of Oxford ([Frey and Osborne, 2013]) and the professional services of Deloitte ([Deloitte, 2015]), almost half of all jobs will be occupied by robots in the next 25 years. Furthermore, as the Mckinsey institute[1] shows in its last report on the global economy ([Institute, 2017]), robots will perform the work of about 800 million jobs in 2030.

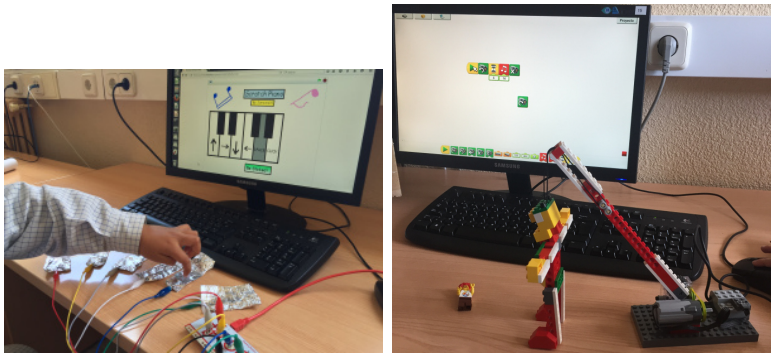


Figure 1.6: Different robotic prototypes to work in different educational areas

[1] `https://www.mckinsey.com`

It is therefore of vital importance to incorporate technology, and specifically Robotics, in the pre-university educational system since todays' youngest students will be those who, within a decade, have to confront a labour market that will demand profiles related to automation of systems ([UK-RAS, 2016]). From the educational point of view, Robotics is a field where many areas converge: electronics, physical (Figure 1.6 left), mechanical (Figure 1.6 right), computer sciences, telecommunications, mathematics, etc.

Figure 1.7: Students working in a robotic prototype

Moreover, the use of technologies in public and private schools will also help to reduce the *digital divide* that currently exists across the world. The digital divide is the term used to describe the large differences in the use of the technology between different ethnic and economic-social groups ([Schiller, 1996], [Wresch, 1996]).

In addition, the increasing strength robotic technology requires professionals to be trained in this sector, extending boundaries even further and helping to create new robotic applications that serve people (Figures 1.8 left and middle) and the progress of humanity (Figure 1.8 right). At present the training in Robotics has some presence in secondary education but is fundamentally conducted in universities, with specific undergraduate and master's degrees.

Nevertheless, it is a fact that Robotics is growing in importance in pre-university education as much in Spain as in other western countries, either as a field of knowledge in itself, or as a tool to present technology and other subjects to young students in an attractive way. Furthermore, Robotics has the power to motivate students and this allows us to bring technology closer

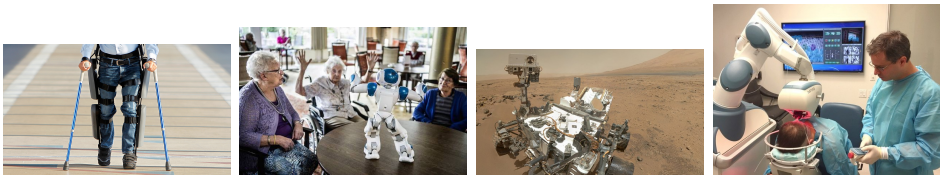


Figure 1.8: Robotics as a profession for society

to boys and girls ([Rodger and Walker, 1996]) using robotics as a tool to present basic concepts of science ([Altin and Pedaste, 2013]), technology, engineering and mathematics (STEM) ([Mubin et al., 2013]). Students learn, almost through playing (Figure 1.7), notions which are difficult and complex to explain or to assimilate through the classic masterclass ([Cerezo and Sastrón, 2015, Jiménez et al., 2010]).

For example, at official level, the Autonomous Community of Madrid (Decree 48/2015, of 14 May, of the Council of Government) included the subject *Technology, programming and robotics* in the official curriculum of Compulsory Secondary Education in the 2015-16 academic year. There is also an increasing demand for robotics for extracurricular activities, which is creating an ecosystem of companies to fulfill this demand. Outside Spain, the implantation of Robotics in education is a fact. Six states in the U.S. (Iowa, Nevada, Wisconsin, Washington, Idaho and Utah) have announced plans and investments with this aim in the last five months. Likewise, four countries —Canada, Ireland, New Zealand and Romania— have recently announced similar plans, with a total investment of 300 million dollars. Japan, in its `New Robot Strategy Report` ([Japan-Economic, 2015]) makes clear that investing in Robotics is fundamental for the growth of the country.

In this educational field, there is a convergence of the teaching of Robotics itself and other disciplines (e.g. programming) using Robotics as a teaching tool ([Magnenat et al., 2014, Merkouris et al., 2017, Kubilinskiene et al., 2017]).

Another example of the increasing importance of Robotics in education are the robotic championships for teenagers, which encourage interest in this technology. For example, the robotic championship RoboCampeones (Figure 1.9) in the Autonomous Community of Madrid, which brought

Figure 1.9: RoboCampeones robotic championship

together no fewer than 2,000 students in its last edition, with challenges like the follow-lines, sumo between robots, etc. Similarly, at international level, numerous championships are organized, which bring together students from all over the world to learn, share experiences and enjoy the development of robotic prototypes. It is worth highlighting the RoboCup Junior[2] ([Eguchi, 2016, Navarrete et al., 2016, Kandlhofer and Steinbauer, 2014]), with tests such as the rescue or robotic soccer. There is also the First Lego League (FLL) and the *VEX Robotics Competitions*[3]. In Finland, the quintessential championship, which attracts students from all over Europe ([Jormanainen and Korhonen, 2010]) and has agreements with centers of South Africa ([Graven and Stott, 2011]), is the *SciFest*[4].

Also, in the academic community a group of congresses and conferences have emerged that emphasize the role of Robotics in Education, including the *Conference on Robotics in Education* (RIE), and the *Workshop on Teaching Robotics with ROS* (TRROS) within the *European Robotics Forum*[5]. Special editions on education in robotics have also appeared in several scientific journals. Finally, it is worth noting the presence of Robotics and Education in high-impact journals: *Computers and Education*, *British Journal of Educational Technology*, *International Journal of Robotics Research*, *Journal of the Learning Sciences* or *Journal of Research in Science*

[2] `http://rcj.robocup.org`
[3] `https://www.vexrobotics.com/vexedr/competition`
[4] `http://www.scifest.fi`
[5] `http://www.eu-robotics.net/robotics_forum`

Teaching.

To support this increasing presence of educational robotics, there exist in the market few educational kits that provide useful and motivating educational frameworks for pre-university students. The existing kits focus on their use in class time, with a low level of complexity, which results in *a short useful* life and a low motivation of students, who achieve the challenges posed in a few sessions of work. There is no system, either, and even less a guided one, that maintains a constant level of motivation and challenge, especially where vision plays an important role.

1.4 Proposal

After having described in Section 1.1 the role of Robotics in society today and in coming years and after describing in Section 1.3 the consistent importance played by education as a means for new generations to prepare for the short and mid-term future, an enormous gap has been identified between the level of the academic training at university level in scientific and technological degrees and the official curriculum implemented at pre-university levels, specifically in science subjects at Secondary Education level.

Thus, this work proposes to acquire an in-depth background, for subsequent educational application of the main problems inherent to Robotics, such as navigation and location, using vision as the main sensor. The development of robot techniques with vision, like visual control, visual memory or a visual attention system is also proposed.

Furthermore, to mitigate this academic gap detected between pre-university training and that offered by universities, the author proposes to develop a complete teaching framework for Robotics with vision, which today is non-existent, integrating: (a) a suitable teaching methodology in accordance with the new profile of *New digital age students*, where each student is more than ever an individual with their own interests ([Rose and Meyer, 2002, Selber, 2004]) within the enormous universe of possibilities offered by this Era of Information ([Solove, 2004]); (b) a software infrastructure that is simple and intuitive for young students to manage

but that at the same time is powerful and versatile, incorporating enough resource libraries to provide practical exercises that are sufficient in both, number and complexity, on programming robots with vision, so as to continuously motivate students ([Benitti, 2012]), as well as diverse examples; (c) a proven academic program that can be followed during a complete academic year and that includes sufficient and properly staggered sessions for correct assimilation by the students ([Ainley et al., 2008]); (d) a versatile hardware platform, economically suitable for secondary education centers to satisfy the needs of a complete class, but at the same time standardized and powerful, which allows the execution of algorithms of Robotics with vision.

At present, only educational kits mainly designed to be sporadically used in class are available. This follows an approach that is considered obsolete, and that in the short run it is not very motivating for students. In fact, the majority of these kits existing in the market are designed to arouse the interest of the youngest students in Robotics, but not so that students in pre-university courses acquire correct and complete training in programming, something which is in great demand and so widespread in almost any degree. Although it is true that other kits exist which are more specialized in specific scientific fields ([Schweikardt and Gross, 2006]), the proposed framework goes further and provides all the necessary tools for both students and teachers ([Bers et al., 2002]) required to develop a complete academic year in a versatile way by putting at their disposal numerous and sophisticated algorithms, including vision, with a pleasant and intuitive interface.

1.5 Goals

The main objective of this work is the development, implementation and evaluation of a complete framework using robots with vision for teaching of Robotics. This environment will mitigate the great gap detected during years of experience by this author between the teaching of scientific-technological subjects at pre-university academic levels and the *curricula* of technical degrees. In this way, new generations may be better trained

in the skills demanded by today's society and in the coming years.

The first objective is to develop the algorithms that provide a solution to the main problems existing within the framework of Robots with vision and that will address issues such as navigation and localization using vision as the main sensor. This in turn will determine and implement the inherent problems in the treatment and selection of useful information of an image, as well as the generation of a complex visual memory that incorporates the elements detected in the scene surrounding the robot, resulting in more intelligent behavior by the robots, which is what will be demanded in Robotics over the coming years.

The second objective is to integrate the above into a complete educational environment that will include an entire academic program to follow during the teaching of subjects such as Technology and ICTs in pre-university courses. For this, we will develop a software infrastructure that facilitates the programming of robots with vision to these students, so that it is simple and intuitive to develop solutions to the classic problems of robots: navigation, vision, etc.

Furthermore, the physical robotic platforms to be considered will be those with a low cost, so that their acquisition by Secondary Education institutions is viable for a complete class. That is why the developments will be based on existing standardized and low cost platforms in the market. In addition, and to meet the above objectives of putting robotics and robot vision algorithms into practice, a physical robotic platform will be developed to computationally allow the execution of these as well as the use of a camera as the main sensor.

This educational environment is intended to be implemented in different schools in the Autonomous Community of Madrid, where Robotics is officially integrated into the Secondary Education curriculum for more than one academic year. The different teaching methodologies existing in the literature will be investigated to analyze which may be the best for the implementation of the Teaching-Learning process of Robotics and which, therefore, will be the one chosen to conduct out the sessions that make up the educational program we develop.

Finally, the proposed teaching environment with real students will be

validated, the academic results and satisfaction will be evaluated by the students and teachers who use the educational environment of this thesis in comparison to the groups that follow the traditional programs of Technology subjects and ICTs. We will analyze the results of this evaluation, which will be obtained through daily interviews, academic qualifications obtained by the students, as well as surveys on the degree of satisfaction across both parties: students and teachers.

1.6 Structure of the thesis

This thesis document has the following structure:

It begins with this Chapter 1 in which the context and motivation of the thesis is presented. Chapter 2 describes the state of the art of Robotics with vision with its different main research fronts. The state of the art of educational Robotics is also studied.

Chapter 3 describes the problems inherent to the navigation and location of robots using vision, as well as the different software developments that we have contributed to solve them with selective attention mechanisms and a short-term visual memory . In addition, this chapter details the mathematical procedures we have used to integrate a conventional camera as a powerful visual sensor.

Chapter 4 is dedicated to describing the design and development of the hardware and software teaching infrastructure created in this thesis and carried out over a number of years, following the chosen academic curriculum. There is also a tour of the different teaching approaches and a description of focused on in this thesis. The validation results of the environment are also described, following its implementation with more than $2,000$ students.

Finally, Chapter 5 presents the conclusions drawn from this work. This is all accompanied by the publications derived from this dissertation. The chapter closes with possible future lines of research.

Chapter 2

State of the art

This chapter describes the state of the art regarding the fundamental pillars on which this research work is based, such as the navigation and location of robots using the vision system as a sensory element, as well as an in-depth analysis of the different methodologies pedagogical that helps us when putting into practice successfully the environment of teaching robotics in the classroom.
Thus, Chapter is structured in five sections. Section 2.1 begins with a brief summary of the state of the art that is detailed in the following sections. Section 2.2 describes the state of the art in visual attention systems, whose main uses shown in the literature are detailed in the sections corresponding to navigation (2.3) and location (2.4). Finally, Section 2.5 shows the state of the art in educational Robotics, making a review of the existing environments in the market.

Efforts are focused on these four pillars already listed in order to prepare new generations of pre-university students in their correct training to confront a short-term future in which intelligent robots will occupy a large part of the labor market. These are so named because they have to be able to reason and interact in a real environment; considering as such one in which there are changing conditions and mobile elements that require machines skilled enough to quickly adapt to what surrounds them

at any time to navigate safely, locate themselves in the environment and, ultimately, interact intelligently with the elements that integrate it.

2.1 Introduction

Vision systems are currently one of the most used sensory elements in autonomous robotics (Section 1.2). Their main difficulty lies in extracting useful information from the captured images, as well as the small visual field of conventional cameras.

However, with *active cameras* it is possible to revisit characteristics of a previously visited area, even if such an area is out of immediate visual range. In order to have accurate information about the areas of interest that surround the robot, a detailed memory map of the environment is necessary. Since the computational cost of maintaining such an amount of information is high, only a few references can be maintained.

Humans already naturally have a precise active vision system ([Robinson, 1964, Clark and Stark, 1975, Bajcsy, 2009]), which means that we can concentrate on certain regions of interest in the scene around us, thanks to the movement of the eyes and/or of the head, or simply distributing the gaze in different zones within the current image that we are perceiving ([Biswas, 2016]).

But, in addition, the aim of integrating a robot as much as possible in a real environment requires that it understands the natural forms of communication of humans. The most basic, primitive form of interaction between humans is to look at the face ([Farroni et al., 2002]). It is therefore interesting to develop techniques that allow the robot to know the position of the people around him and to follow them at all times ([Sidner et al., 2004]). Hence, for an acceptable robot-human interaction, a face detection and tracking mechanism ([Lang et al., 2003, Parks et al., 2014, Menéndez et al., 2013]) is indispensable. This allows a more fluid interaction with people, because they perceive the robotic interlocutor as more natural ([Rautaray and Agrawal, 2015]).

This visual attention (Section 2.2) can be used for robots to detect and avoid obstacles, because it is generally required to navigate autonomously

through dynamic environments (Section 2.3). When using cameras, obstacles can be detected through 3D reconstruction. Thus, the recovery of three-dimensional information is the main focus of the computer vision community for decades ([Goldberg et al., 2002]). Although many other works have presented solutions that do not require 3D reconstruction ([Remazeilles et al., 2006]).

Other relevant information that can be extracted from the images is the location of the robot (2.4). Robots need to know their location within the environment in order to develop the appropriate behavior. Using vision and a map, the robot can estimate its own position and orientation within a known environment. The auto-location of robots has proven to be complex, especially in dynamic environments and those with high degree of symmetry, where the values of the sensors can be similar in different positions.

Far from these real problems of Robotics are numerous educational Robotics kits (Section 2.5), which usually include the components needed to build the robot and a programming environment specific to that robot and that typically consists of a simple and intuitive graphic interface for students, but with limited potential. The objective of these kits is to enrich the education of students and introduce them to Science and Technology. They are based on the imitation of real robots but very far removed in functionality with respect to employees in a real and/or industrialized environment.

2.2 Visual attention system

Visual attention has two clearly marked stages: the first, considered prior processing, is one in which objects are extracted —that fulfill certain characteristics— within the visual field; and the second, called focused attention, consists of the identification of those objects.

Within autonomous robotics it is important to perform a visual attention control ([Nobre, 2015]). The cameras of the robots provide a wide flow of data from which you have to select what is interesting and ignore what is not; this is the selective visual attention. There are two aspects of vi-

sual attention, the global (*overt attention*) and the local (*covert attention*). Local attention ([Tsotsos et al., 1995, Itti and Koch, 2001, Marocco and Floreano, 2002]) consists of selecting within an image those data that interest us. Global attention consists of selecting from the environment that surrounds the robot, beyond the current visual field, those objects that interest, and directing the look towards them ([Cañas et al., 2008, Borji et al., 2013]).

The visual representation of interesting objects in the vicinity of the robot can improve the quality of robot behavior, as well as the ability to handle more information when making decisions. This poses a problem when the objects are not in the immediate field of vision. To solve this problem, omnidirectional vision is used in some works ([Gaspar et al., 2000]); in others, a normal camera and a global attention mechanism are used ([Itti and Koch, 2001, Zaharescu et al., 2005]), which allows samples to quickly be taken from a very wide area of interest. The use of camera movement to facilitate the recognition of objects was proposed by [Ballard, 1991] and is used, for example, to distinguish between different shapes in the images ([Marocco and Floreano, 2002]).

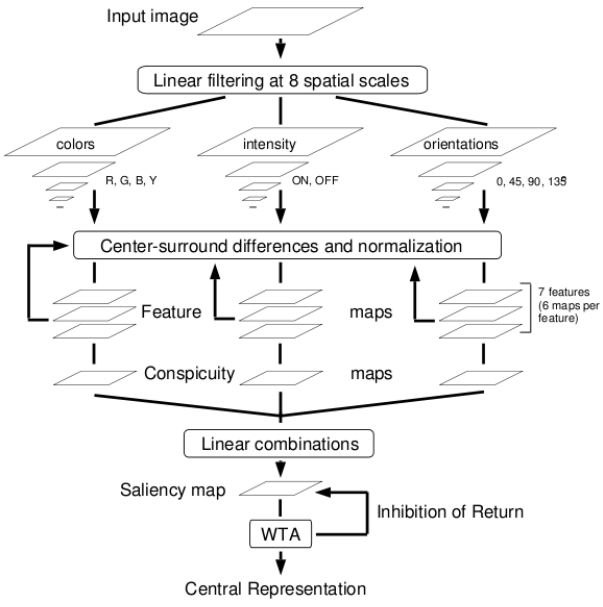


Figure 2.1: Diagram of salience map formation

One of the concepts widely accepted in the works in this area is the map of salience. We can find it in [Itti and Koch, 2001, Borji et al., 2014] as a local visual attention mechanism, independent of the particular task to be performed, and formed by the set of visual stimuli that draw the attention of the scene. This work considers a bottom-up model, where —as we can see in Figure 2.1— in each iteration the different descriptive maps of the scene (according to colors, intensities or orientations) then merge into what they call maps of conspicuity (one for each characteristic feature), which will finally make up a unique and representative map of salience.

There are two fundamental advantages offered by this natural solution compared to a passive attention mechanism where the sensors are equally centered in all areas of the image.

1. Areas of the scene that can not be accessed by fixed sensors, but by mobile sensors.

2. By directing attention to specific areas of the image that are interesting, we can avoid costs in visiting areas that we do not concern us. For example, in the task of catching, humans concentrate only on the moving object.

Thus, we can consider active vision as a supervisor of a broad repertoire of tasks, notably greater than that contemplated by passive vision.

A good overview of these approaches to model visual attention is found in [Bauer, 2015], where the author assumes that his evaluation is not trivial, since the applications are diverse and the biological processes that take place in our brain, the model for many of them, are still mysterious and are only being slowly understood despite the large number of psychophysical studies and experiments.

Another solution was proposed in [Hulse et al., 2009], where an active robotic vision system was presented based on the biological phenomenon of feedback inhibition, which is used to modulate the action selection process for the saccades of the camera. In this article it was argued that visual information has to be processed later by a series of cortical and subcortical

structures that: (1) place it in a context of attentive bias within egocentric salience maps; (2) implement the aforementioned return inhibition entries of other modalities; (3) cancel the voluntary saccades and (4) influence the selection of the action of the basal ganglia. Therefore, there is a specific and sophisticated method within a biological context to facilitate the most appropriate saccadic movement as a way of selecting attention.

Arbel and Ferrie presented/displayed in [Arbel and Ferrie, 2001] a strategy of planning of the glance that moves the camera to another point of view around an object to recognize it. The recognition itself is based on the optical flow paths that result from the movement of the camera. The new measurements, accumulated over time, are used in a one step ahead Bayesian approach that solves the ambiguity of object recognition, while navigating with an entropy map.

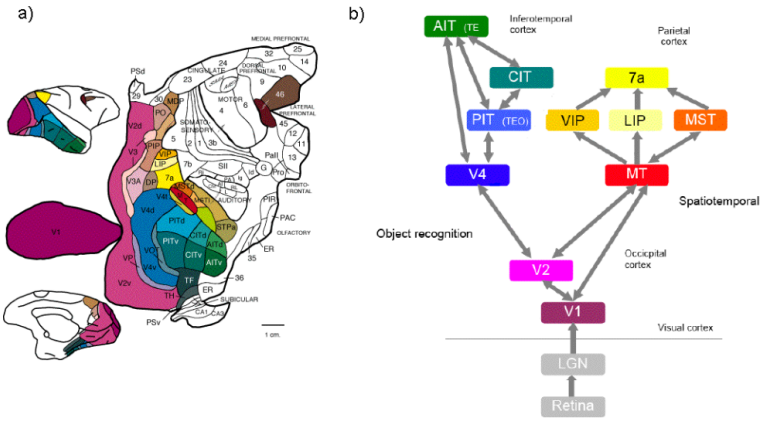


Figure 2.2: (a) Visual cortex of primates, (b) Simplified version of the visual cortex

In his thesis, Rodríguez-Sánchez defined ([Rodríguez-Sánchez, 2010]) a model inspired by Biology for the representation of forms that included intermediate layers of visual representation corresponding to the layers that exist naturally in the natural visual cortex (Figure 2.2), not previously explored, and which showed that they had a direct impact on the selection of color and 2D shapes. Furthermore, their combination led to the formation of selective neurons.

Two profound comparative studies are found in [Borji and Itti, 2013]

and in [Bylinskii et al., 2015], where different taxonomies of up to 65 models are presented, which provide a critical comparison of approaches, their capacities and deficiencies. In particular, more than a dozen criteria derived from behavioral and computational studies are formulated for the qualitative comparison of care models. In addition, they address several challenging problems with the models, including the biological plausibility of the calculations, the correlation with eye movement data-sets, the ascending and descending dissociation, and the construction of meaningful performance measures.

2.3 Vision-based navigation

Regarding vision-based navigation, Badal ([Badal et al., 1994]) presented a system to extract gradual information and perform detection and avoidance of obstacles in outdoor environments, based on the calculation of the disparity of the two images of a pair stereo of calibrated cameras. The system assumes that the objects are on the ground plane located at the bottom. Each point on the ground is configured as a potential object and is projected onto the ground plane in a local occupation grid called the Instant Obstacle Map (IOM). The commands to direct the robot are generated according to the position of the obstacles in the IOM.

Goldberg ([Goldberg et al., 2002]) introduced a navigation algorithm based on stereo vision for the mobile exploration robot on Mars, Rover (MER), whose objective was to be able to explore and map dangerous terrains locally. The algorithm calculates the epipolar lines between the two images of the stereo pair and thus verifies the presence of an object, calculates the Laplacian of both images and correlates the filtered images so that the pixels of the left image coincide with their corresponding pixels in the image of the right. The work also includes a description of the GESTALT navigation module, which packages a set of routines capable of calculating action and direction commands from the sensor information.

Remazeilles ([Remazeilles et al., 2006]) presented the design of a control system for the autonomous navigation of robots using vision as a perceptual system. The particularity of this system is that it does not require any

reconstruction of the environment and does not force the robot to converge towards each intermediate position of the route.

Srinivasan ([Srinivasan et al., 2006]) introduced a new system to increase the accuracy of optical flow estimation for insect-based flight control systems. A special mirror surface is mounted in front of the camera, which points forward instead of pointing to the ground. The mirror surface slows down movement and eliminates the distortion caused by perspective. Theoretically, the image must have a slow and constant speed at all times, thus simplifying the calculation of the optical flow and increasing its accuracy. As a result, the system increases the speed range and the number of situations under which the aircraft can fly safely.

In [Chen and Birchfield, 2009] a simple approach for tracking a vision-based route for a mobile robot is presented. Following the concept of lane reduction or *funnel lane*, the coordinates of the characteristic points during the navigation phase are compared with those obtained during the learning phase to determine the direction of rotation. Greater robustness is achieved by coupling the coordinates of the functions with the odometry information. The system requires a single camera facing forward, without external or internal calibration. The algorithm is qualitative in nature, and does not require a map of the environment, nor a Jacobian image, nor a homography, nor a fundamental matrix, nor any assumption on a plane. The experimental results demonstrate the capacity of autonomous navigation in real time in indoor and outdoor environments and in flat, inclined and rugged terrain with objects of dynamic occlusion for distances of hundreds of meters. It also works with wide angular and omnidirectional cameras, with only minor modifications.

In [Hornung et al., 2010], the authors develop a novel approach to learning efficient navigation policies for mobile robots that use visual characteristics for localization. To avoid the blur caused by the inherently fast movements of a robot while navigating, they introduce a reinforcement learning approach to determine the appropriate navigation policy in each situation, balancing between speed and accuracy in the location, and taking into account the impact of motion blur in observations. In addition, they develop a method to compress the policy learned through a cluster

approach, which is especially desirable in the context of systems with limited memory. It is shown that this learned policy significantly exceeds policies that use a constant speed and more advanced heuristics. In addition, it is generally applicable to different indoor and outdoor scenarios with different densities of reference points and navigation tasks of different complexity.

In [Dimitrov et al., 2015], the improvements are presented in the autonomous navigation of the Rover of Autonomous Exploration of NASA (AERO) in terms of robustness and reliability of sample collection. The AERO is required to navigate a large outdoor area, find and collect several geological samples, and return to the home platform autonomously and using only technologies compatible with the space.

In the study [Faessler et al., 2016], Matthias and his team describe how an aerial microrobotic device can autonomously and visually describe a specific trajectory and at the same time build a three-dimensional map of the area over which it flies. The processor it uses is the same as that of a mobile phone, which due to its low weight is passively safe and can be deployed close to humans, for rescue tasks.

Another use of a navigation system based on vision, internationally recognized by researchers, can be seen in [Tweddle et al., 2016]. This study describes the VERTIGO vision system, a hardware update to the SPHERES satellites that allows the investigation of navigation based on vision in the environment of 6 degrees of freedom and microgravity of the International Space Station (ISS). This vision system includes stereo cameras, designed to be used by researchers in numerous experiments.

The weaknesses of a vision-based global navigation system are shown in [Smith et al., 2018]. This evaluation also compares it with the introduction into the system of *Deep Learning* techniques, and concludes that it is very efficient to apply this novel system to provide samples of the environment to the global planning system, so that it makes smarter decisions. In particular, simulations and tests on a real mobile robot show that the number of samples obtained by Deep Learning can be reduced by an order of magnitude and preserve navigation performance.

2.4 Self-localization using vision

The probabilistic location algorithms based on grids with laser or sonar information were successfully applied in small known environments ([Burgard and Fox, 1997]). They use discretized probability distributions and update them from sensor data and movement orders, accumulating information over time and providing a robust position estimate. Particle filters use sampled probability functions and extend the techniques to larger environments, using them even with visual data such as input ([Dellaert et al., 1999]). At the beginning, the maps were provided in advance ([Royer et al., 2007]), but in recent years the SLAM techniques address the location simultaneously with the construction of the map ([Blosch et al., 2010]).

These SLAM (Simultaneous Localization and Mapping) techniques try to locate the robot accurately in unknown environments where no map is available; and to create this it is necessary to be perfectly localized ([Fuentes-Pacheco et al., 2015]). That is, SLAM solves the most complex cases, in which the environment is unknown and there is no information about the exact location of the robot. The way to proceed in these situations is to generate a map while maintaining the location simultaneously. For this, the robot only has the information provided by the measurements obtained by its sensors, as well as the information that can be extracted from the robot's own movement. In this context there are therefore several points of uncertainty that must be taken into account ([Atanasov et al., 2015]): sensor noise, inaccurate robot displacement, environmental symmetries, partial observability, dynamic environment, etc.

There are many SLAM techniques based on particle filters. In addition, perhaps the most successful approach in recent years is the MonoSLAM of A. Davison ([Gerardo Carrera and Davison, 2011]). The detection of relevant points in the image and an extended Kalman filter allows the system to continuously estimate the position and orientation of the 3D camera and the 3D position of those points. The location results are impressive. The quality of the maps, mainly as a collection of 3D points, was not as good at first, but they have improved it even with dense real-time maps ([Newcombe and Davison, 2010, Weiss et al., 2011]).

Genetic Algorithms (GA) have also been proposed for robot localization tasks. For example, in [Tong and Meng, 2007] the location of a robot with legs is modeled as the optimization of an objective function, and the GAs are used for that optimization using the vision data as input. In [Duckett, 2003], Duckett combines an Extended Kalman Filter and a genetic algorithm to locate a robot within a known environment using ultrasonic sensors; this technique uses the EKF to obtain a seed and then searches within the seed neighborhood for the most accurate solution with a GA. SLAM approaches also include genetic algorithms, such as [Moreno et al., 2002], where the best trajectory of a robot is calculated with a GA according to the odometry of the robot and laser measurements.

In [Jensfelt and Kristensen, 2001], Jensfelt and Kristensen present an active global localization strategy that uses a Kalman filter (KF) to track multiple robot posture hypotheses. Its approach can be used even with incomplete maps and the computational complexity is independent of the size of the environment.

Gartshore ([Gartshore et al., 2002]) develops a map construction framework and a characteristic position detector algorithm that processes images online from a single camera. The system does not use coincidence approaches. Instead, calculate the probabilities of finding objects in each location. The algorithm begins to detect the boundaries of the objects for the current frame using Harris edge and corner detectors. The detected characteristics are projected again from the 2D image plane taking into account all possible locations at any depth. The positioning module of the system calculates the position of the robot using odometry data combined with the extraction of image characteristics. The color or gradient of the edges and the characteristics of previous images help to increase the confidence of the presence of the object in a certain location. The experimental results tested in indoor environments establish the size of the grid cells at $25mmx25mm$. The robot moves $100mm$ between consecutive images.

In [Mariottini and Roumeliotis, 2011], Mariottini and Roumeliotis present a strategy for active navigation and location based on the vision of a mobile robot with visual memory where previously visited areas are represented as a large collection of images. Clarifies the location by taking into account

the sequence of distinctive images, while simultaneously navigating to the original image.

2.5 Educational Robotics

Educational robotics encompasses a wide range of robotic platforms designed for use in the Teaching-Learning process (T-L) in an educational context. The literature on the subject is increasing ([Sklar et al., 2003, Goldman et al., 2004, O'Hara and Kay, 2003, Eguchi, 2012]), as are the congresses on this subject and the realization of different international projects, as already commented in Section 1.3.

In the basic definition of Robotics, in educational terms, two fundamental branches have to be considered: *(1)* hardware, which represents the physical body of the robot and includes sensors, actuators and the central microcontroller; and *(2)* software or programming of the robot, whose language may or may not depend on the platform used. The actuators are the mechanisms that allow the robot to interact with the environment that surrounds it and which are, essentially, motors and servos. The sensors are the mechanisms that report information on the outside world to the robot, for example: laser, infrared, ultrasound, contact and camera. The central microcontroller is the chip that performs the processing of the data received by the sensors to conveniently command the actuators.

There are many teaching environments used to teach robotics to children, from those focused on primary education to more powerful ones oriented to secondary education and high school. They are usually composed of a concrete *robotic platform*, that is to say a robot, which is programmed in a certain *language* using *software tools*. Different exercises, challenges or projects are then proposed to the students (*practice activities*). They teach the basic operation of sensors, actuators and the rudiments of programming. These teaching environments are used as a tool within an specific way of teaching robotics classes, that is, of a particular methodology.

Four elements are identified that characterize the numerous ways of teaching robotics to adolescents and the most used environments: hardware platform, software language and infrastructure, concrete practices

and methodology. Several illustrative examples will be described in this section and the proposed teaching proposal will be framed in Chapter 4.

2.5.1 Hardware platforms

The robots used in pre-university education usually incorporate a limited processor, sensors and simple actuators. There are frequent infrared sensors (photodetectors), ultrasound, contact, sound, light sensors, etc. The usual actuators include LEDs, screens, small loudspeakers and fundamentally motors. These motors can be of several types: DC motors, stepper motors or servomotors. They are usually connected to the robot's processor using direct cables or simple connectors (such as RJ25).

Some platforms have a closed mechanical design, others allow some flexibility from pre-built blocks that can be connected in multiple ways, or parts with sensors or actuators where which to mount in each case and in what position can be decided by the students. Other platforms do not have any *a priori* mechanical design. They are open-ended, and provide the learning material with the students.

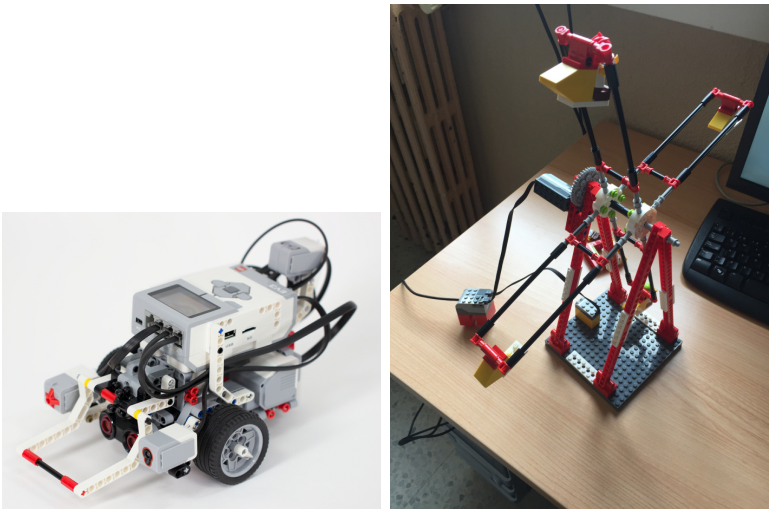


Figure 2.3: LEGO robots: Mindstorm Ev3 and WeDo

Some widely used platforms are the LEGO one in their different models: MindStorms RCX, NXT, EV3 and WeDo (Figures 2.3 left and right) ([Jiménez et al., 2010, Navarrete et al., 2016]).

Figure 2.4: Arduino free hardware microcontroller board

Another option widely used, both in secondary and high school, are plates with Arduino processors (Figure 2.4) to which low-cost sensors and loose servos are connected ([Araujo et al., 2015, Jamieson, 2012, Navarrete et al., 2016, Chaudhary et al., 2016, Filippov et al., 2017]). It allows students to interact with a real robot, sensors and real actuators at an affordable cost. It also offers many didactic possibilities, such as those described in ([Junior et al., 2013, Plaza et al., 2016, Balogh, 2010, Afari and Khine, 2017, Beyers and van der Merwe, 2017]).

Figure 2.5: Robots Thymio, VEX IQ and VEX CORTEX

Another prominent platform is that of the Thymio robot ([Mondada et al., 2017, Magnenat et al., 2014]), open hardware, and the Thymio-II (IniRobot [Roy et al., 2015], Figure 2.5 left). In addition, VEX robots robots and robotic kits are used with certain frequency in education ([Demetriou, 2011]): IQ and CORTEX models (Figure 2.3 middle and right).

It is worth mentioning the Spanish manufacturer robots BQ Zowi (Figure 2.6 left), based on Arduino, and PrintBot evolution (Figure 2.6 right),

Figure 2.6: BQ robots: Zowi and PrintBot

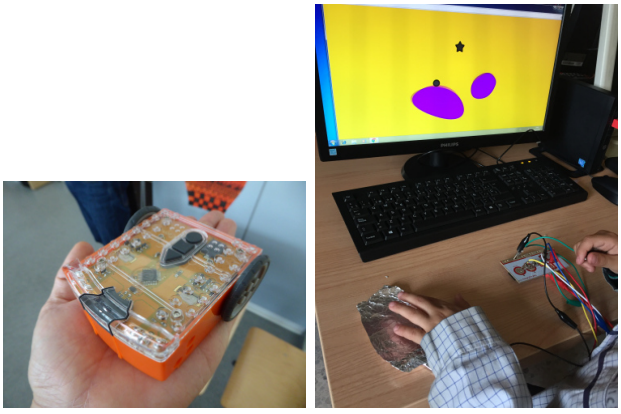


Figure 2.7: Meet Edison robot and Makey-Makey board

based on the ATmega328P microcontroller. Also worth noting are Meet Edison's robots[1] (Figure 2.7 left); these are small robots created by an Australian company that allow younger children to start managing and programming a robot. Finally, we have the Makey-Makey[2] (Figure 2.7 right) plates, which allow transforming any electrical current, however weak, into a signal that is interpreted and used to simulate for example a joystick or the keys of a piano; it is usually used in a simulated physics environment called Flabby Physics[3].

In addition to real robots, simulated robots are also used in pre-university education. For example, the TRIK-Studio environment includes a simple 2D simulator for the TRIK robot ([Filippov et al., 2017, Stone and Farkhat-

[1] `https://meetedison.com`
[2] `https://makeymakey.com`
[3] `http://flabbyphysics.com`

dinov, 2017]). Another important example is the 3D simulator used in Robot Virtual Worlds (RVW)[4] ([Witherspoon et al., 2017]), which simulates robots from different manufacturers (VEX, LEGO and TETRIX).

2.5.2 Languages and software environments

Typically, each robot has a software environment that allows programming in a certain language. The environment usually includes code editors, utilities to download in real robot and even simulators in some occasions. As for languages, simple languages are used to facilitate programming by children and include instructions for ordering commands to actuators, to read sensor measurements, loops, conditional and sequencing instructions.

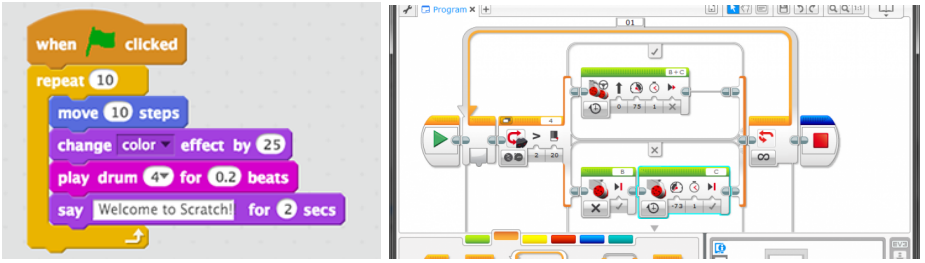


Figure 2.8: Programs in graphic language Scratch and in EV3-software

A successful option is the graphic languages of LEGO, specific to their robots, for example the old RCX-Code, RoboLab (built within LabVIEW), NXT-G and the latest EV3-software (Figure 2.8 right). All contain blocks of action, sensors, flow control, operations with data, etc.

Another visual alternative is the Scratch language[5] (Figure 2.8 left) ([Beyers and van der Merwe, 2017, Olabe et al., 2011, Plaza et al., 2017]) or variants such as Blockly (for example with the robot RoBOBO ([Naya et al., 2017])), as Bitbloq (for BQ-Zowi) or as VPL (for Thymio). All these also have graphic blocks that typically connect in sequence in a graphic editor.

Another option is simple text languages, such as the Arduino language with the Arduino IDE editor (Figure 2.9). JavaScript is also used, as

[4] `http://www.robotvirtualworlds.com/`
[5] `https://scratch.mit.edu/`

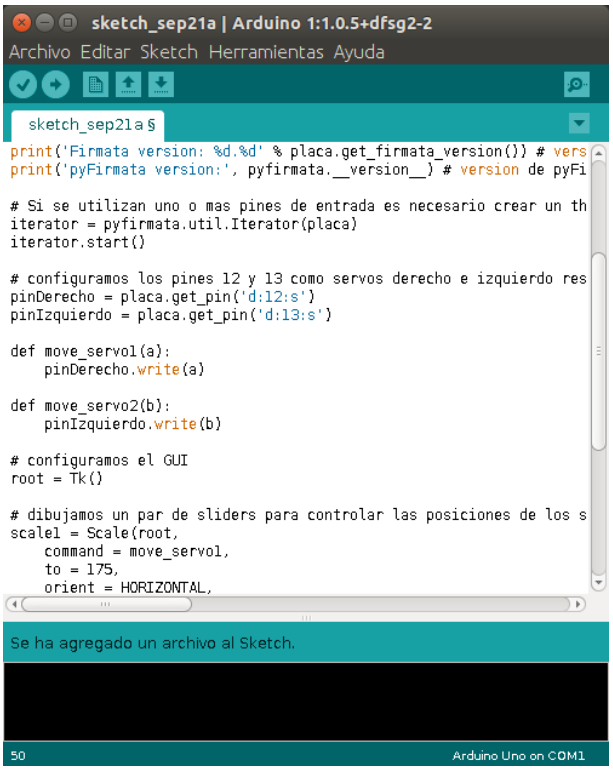


Figure 2.9: Program in Arduino IDE environment

with the educational robot TRIK ([Stone and Farkhatdinov, 2017]) and its TRIK Studio environment, which also has support for its own visual language.

Languages such as C++, which are used successfully at university level, are not recommended for adolescents due to their complexity. However, similar languages to C are used, without object orientation. For example NXC for LEGO robots ([Navarrete et al., 2016]).

In this line is the ROBOTC[6] environment, which uses the C language and a graphical variant of it (ROBOTC-graphical) to program robots from different manufacturers (VEX IQ, VEX CORTEX, LEGO EV3, LEGO NXT and Arduino) and simulated robots in RVW. In particular it is used in the Carnegie Mellon Robotics Academy ([Witherspoon et al., 2017]) with different exercises and competitions.

[6] `http://www.robotc.net`

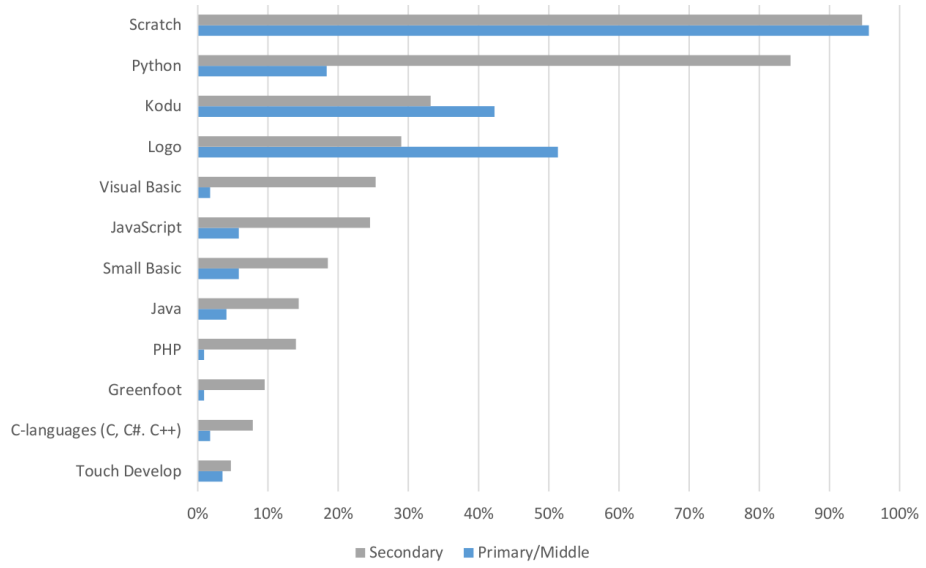


Figure 2.10: Languages used in primary and secondary education in the UK, 2015

Sentance ([Sentance, 2015]) analyzed the use of programming languages in UK schools through a survey of 1,159 teachers of Technology (Figure 2.10). The most widely used language is Scratch (95 % in primary and secondary), followed by Python (18 % in primary, 84 % in secondary).

2.5.3 Exercises

Robotic teaching has a marked practical character. By its very nature, it lends itself to *learning by doing*. Thus, in addition to the theoretical content, emphasis is usually placed on certain projects or exercises that students have to tackle and solve using the appropriate robot and its software environment. Performing these projects means that students encounter specific problems and that to solve them, they learn a range of robotic knowledge.

Exploring the existing literature we have found a set of exercises that are frequently used in different teaching environments and academic proposals, in an almost cross-curricular way. One of the classic projects is behavior follows-lines ([Filippov et al., 2017, Stone and Farkhatdinov, 2017, Navarrete et al., 2016, Jiménez et al., 2010]). In this, the robot

has IR sensors pointing to the ground, which is white but has a thick black line. Another is the avoidance of obstacles ([Stone and Farkhatdinov, 2017, Navarrete et al., 2016]), where the robot has an ultrasound sensor that allows it to detect objects that interfere with the movement of the robot. The student's program must then order the motors to stop and turn until they find a free space for the new advance.

There are also several exercises aligned with tests within some championships, such as the game of sumo between two robots ([Filippov et al., 2017]), and several related to robotic football ([Filippov et al., 2017]). These exercises allow a competitive playful approach that increases the motivation of the students.

Other interesting examples are that of the robot that follows a wall ([Stone and Farkhatdinov, 2017]) or that escapes from a labyrinth ([Filippov et al., 2017]).

2.5.4 Methodologies

Teaching methodologies underpin the cognitive processes that take place in students when they learn. They are different ways of motivating students. All of them seek to reach out to the students, capture their attention and/or awaken interest in the subject ([Bain, 2007, Vega, 2011]). Several stand out: (a) the traditional approach, (b) constructivism, (c) project-oriented learning, (d) cooperative learning, (e) problem solving, and other derived methodologies. Rather than mutually exclusive, they are complementary. One usually makes use of one or the other according to the objectives to be achieved in class. The nature of teaching robotics means it is usual to emphasize the practical approach.

The traditional approach is based on master classes, whose teaching-learning process is radically marked: the teacher teaches, the student receives ([Dewey, 2007]). They usually include teaching material for theory and practical exercises, with instructions that students follow.

Constructivism considers that by providing students with the necessary tools, they can build their own procedures to solve a problematic situation, which implies that their ideas can be modified and learning can be continued.

Project-oriented learning also considers the student more responsible for their own learning. In this case, it is focused on applying to real projects the skills and knowledge that they have acquired in the master class theory sessions ([Morón, 2015]). A frequent project is the participation in robotic competitions where the students' robots (by groups or by centers) have to compete with those of other students passing tests. This participation helps motivate them.

Cooperative learning ([Schul, 2011]) focuses on assessing the educational potential derived from the interpersonal relationships of any group, so that the work is carried out in common, balancing and taking advantage of the skills of its components.

Problem Based Learning ([Acosta-Nassar, 2014]) is mainly based on constructivist theory, so it follows its fundamental principles. The importance of this methodology lies in its influence when reorganizing the information stored in the student's cognitive structure, which is modified. In this very modification, learning takes place.

Within the European Erasmus+ project, the most commonly used methodologies in the field in Finland were studied ([Opoku and Lehtinen, 2015]). The constructivist approach and problem-based learning are mostly used in the Finnish pre-university education system. There is a great advance in the results of the PISA report ([OECD, 2015]) the paradigm shift in the Teaching-Learning process ([Tirri, 2014]), where the teacher is not the only transmitter of knowledge, but also becomes a guide in the self-learning of the adolescents, who can give free rein to their creativity thanks to robotics ([Jormanainen and Erkki, 2013]).

They clearly differentiate between formal sessions, in which the teachers deal with content that they consider necessary for the student, and non-formal ones, in which the students themselves perform the learning process by their own means ([Tirri and Kuusisto, 2013]) .

In addition to specific teaching, Robotics is also used in Finland as a vehicular subject. In this 2017-18 school year, included in the secondary curriculum is that all students must know how to program robots at least in a simple way, with LEGO MindStorms. Thus, for example, they introduce a series of specific skills to be developed from 1st to 9th grade in Maths.

Chapter 3

Robotic vision systems

This chapter shows the developments carried out on a dynamic visual memory to store the information gathered from a moving camera on board a robot, an attention system to choose where to look at with this mobile camera, and a visual localization algorithm which uses this visual memory. All the experiments which are shown in this chapter, and much more, are publicly available[a]. The organization of this chapter is as follows. The first Section (3.1) introduces the design of the visual perceptive system developed, its components and connections. The next three sections describe its three building blocks: the visual memory component (Section 3.2), the visual attention module (Section 3.3) and the localization component (3.4). Finally, several experiments are shown in the last Section 3.5, which were carried out with simulated and real Pioneer and Nao robots to validate the system and each of its components.

[a] `http://jderobot.org/JulioVega_PhD`

Cameras are one of the most relevant sensors in autonomous robots. Two challenges with them are to extract useful information from captured images and to manage the small field of view of regular cameras.

In this chapter, a novel visual perceptive system is shown, which was developed for an autonomous robot composed of three modules. First, a

short term dynamic visual memory of robot surroundings. It gets images from a mobile camera, extracts edge information and offers a wider field of view and more robustness to occlusions than instantaneous images. This memory stores 3D segments representing the robot surroundings and objects. The memory contents are updated in a continuous coupling with the current image flow. Second, a gaze control algorithm was developed to select where the camera should look at every time. It manages the movement of the camera to periodically reobserve objects already stored in the visual memory, to explore the scene, and to test tentative object positions in a time sharing fashion.

These two modules working in conjunction build and update an attentive visual memory of the objects around the robot. Third, a visual localization algorithm was developed which uses the current image or the contents of the memory to continuously estimate the robot position. It provides a robust localization estimation and was specifically designed to bear with symmetries in the environment.

3.1 Design

The perceptive system developed is designed for autonomous robots that use a single mobile camera, like that on the head of humanoids or in robots with pan-tilt units ([Vega et al., 2013]). The block diagram of the robot control architecture is showed in Figure 3.1. The three building blocks (visual memory, gaze control and localization algorithm) were grouped into two main software components: `active_visual_memory` and `localization`. They receive data from robot sensors, like camera and encoders, and extract refined information like description of the objects around the robot or the robot position. This information is provided to other actuation components like the navigation algorithm or other control units.

First, the `active_visual_memory` component builds a short term visual memory of objects in the robot's surroundings. The memory is built analyzing each camera image looking for relevant objects (like human faces, segments, parallelograms, arrows, etc.) and updating the object features

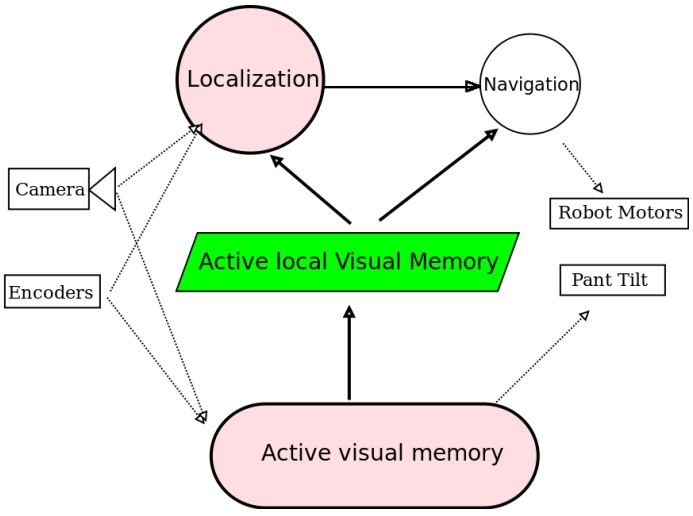


Figure 3.1: Block diagram of the proposed visual system

already stored in the memory, like their 3D position. The memory is dynamic and is continuously coupled with camera images. The new frames confirm or correct the object features stored in memory, like their 3D relative position to the robot, length, etc. New objects are introduced in memory when they appear in images and do not match any known object.

This memory has a broader scope than the camera field of view and objects in memory have more persistence than the current image. Regular cameras typically have 60 degrees of scope. This would be good enough for visual control but a broader scope may improve robot responses in tasks like navigation, where the presence of obstacles in the robot's surroundings should be taken into account even if they lie outside the current field of view.

This memory is intended as local and short-term. Relative object positions are estimated using robot's odometry. Being only short term and continuously correcting with new image data there is no much time to accumulate error in the object estimated relative position. Currently, the system only deals with objects on the floor plane and uses a single camera. It can be extended to any 3D object position and two cameras.

Second, in order to keep this short term visual memory consistent with

reality, the system has mechanisms to properly refresh and update it ([Vega and Cañas, 2011]). The camera is assumed to be mobile, typically mounted over a pan-tilt unit. Its orientation may be controlled and changed during robot behavior at will, and so, the camera may look towards different locations even if the robot remains static. In order to feed the visual memory, an overt attention algorithm was designed to continuously guide camera movements, choosing where to look at every time ([Vega et al., 2012a]). It was inserted inside the `active_visual_memory` component and associates two dynamic values to each object in memory: *salience* and *life* (quality). Objects with low life are discarded and objects with high salience are good candidates to look at.

The position of objects already in memory are themselves foci of attention in order to refresh their perceived features. Random locations are also considered to let the robot explore new areas of its surroundings. In addition, new foci of attention may also be introduced to check the presence of some hypothesized objects. For instance, once the robot has seen three vertices of a parallelogram, the position of the fourth one is computed from the visual memory and ordered as a tentative focus of attention for the camera.

Third, a vision based localization algorithm was developed in the localization component. It uses a population of particles and an evolutionary algorithm to manage them and find the robot position ([Vega et al., 2012c]). The health of each particle is computed based on the current image or based on the current contents of the visual memory. The local visual memory provides information about robot's surroundings, typically more than the current instantaneous sensor readings. In this way, the visual memory may be used as a virtual sensor and its information may be used as observations for the localization algorithm. Because of its broader scope it may help to improve localization, especially in environments with symmetries and places that look like similar according to sensor readings ([Vega et al., 2012b]).

These two software components and the three building blocks will be described in detail in the following sections.

3.2 Visual memory

The goal of the visual memory is to do a visual tracking of the various basic objects in the scene surrounding the robot. It must detect new objects, track them, update their relative positions to the robot and remove them from the memory once they have disappeared.

Figure 3.2 shows the main modules of the visual memory building block. The first stage of the visual memory is a 2D analysis, which detects 2D segments present in the current image. These 2D segments are compared with those predicted from the current visual memory 3D contents. The 3D object reconstruction module places relevant 2D segments in 3D space according to the *ground-hypothesis* which, as a simplifying hypothesis, let is assumed that all objects are flat on the floor (it will be showed in detail in Section 4.2.2). Finally, the 3D memory module stores their position in the 3D space, update or merge them with existing 3D segments, calculates perceptual hypotheses and generates new predictions of these objects in the current image perceived by the robot.

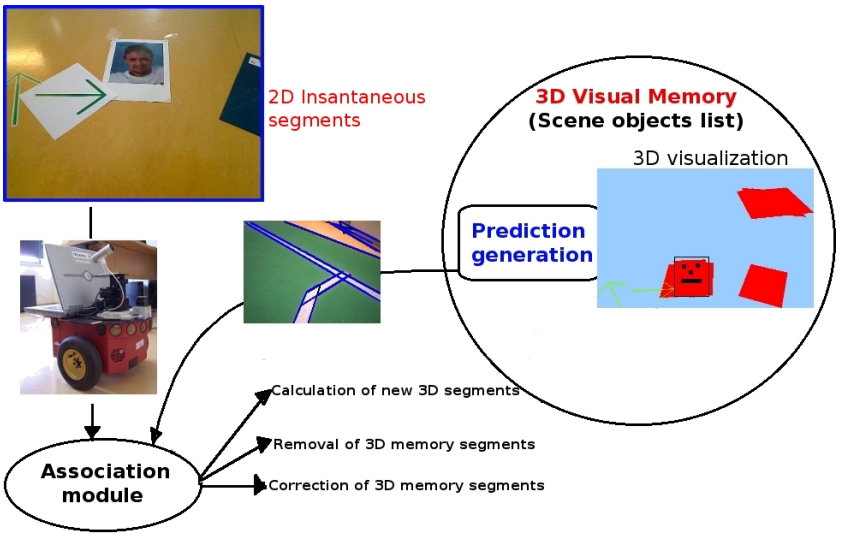


Figure 3.2: Modules of the Visual Memory

The visual memory also creates perceptual hypothesis with the stored items, allowing the system to abstract complex objects ([Vega et al., 2010]).

For instance, it groups a set of 3D segments into the parallelogram concept if some geometric properties are hold (Code 3.1).

The visual memory was coded inside the `active_visual_memory` software component, running iteratively at a frequency of $5Hz$. In each iteration a motion prediction is made for objects in the visual memory (Code 3.2) according to robot encoders, images are acquired and displayed, image processing occurs and the memory contents are updated. To save computing power, robot odometry can be used to trigger the image processing and the system only analyzes frames when the robot has moved away a certain distance or angle from the position where the last image processing was done.

3.2.1 2D Image Processing

The main goal of this module is to extract 2D straight segments as a basic primitive to get object shapes. In prior implementations it is used the classic Canny edge filter and Hough transform to extract lines, but it was not accurate and robust enough. Usually its outcome was not fully effective, line segments were often disconnected as can be seen in Figure 3.3. In last releases this was replaced with a Laplace edge detection and the Solis' algorithm [Solis et al., 2009] for 2D extraction, improving the results. Solis' algorithm uses a compilation of different image processing steps such as normalization, Gaussian smoothing, thresholding, and Laplace edge detection to extract edge contours from input images (Code 3.3). This solution is surprisingly more accurate, robust, faster and with less parameters for detecting line segments at any orientation and location than the widely used Hough Transform algorithm.

The OpenCV library is used to implement these techniques. A comparison can be seen in Figure 3.3, where the Solis algorithm extracts many more 2D segments. The detected segments are shown as blue lines. While the Hough approach is able to recognize just a really small set of segments, the Solis one gets most of them. The floor used in this image is a textured surface and so some false positives appear. The steps and parameters used in Hough algorithm is shown in Code 3.4.

The 2D analysis system is connected directly to the 3D visual memory

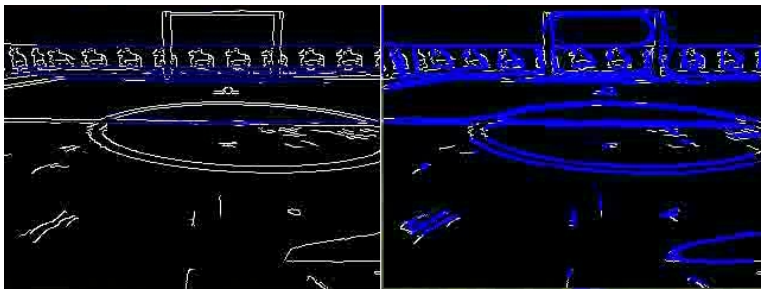


Figure 3.3: Differences between Canny+Hough (left) and Solis algorithm (right)

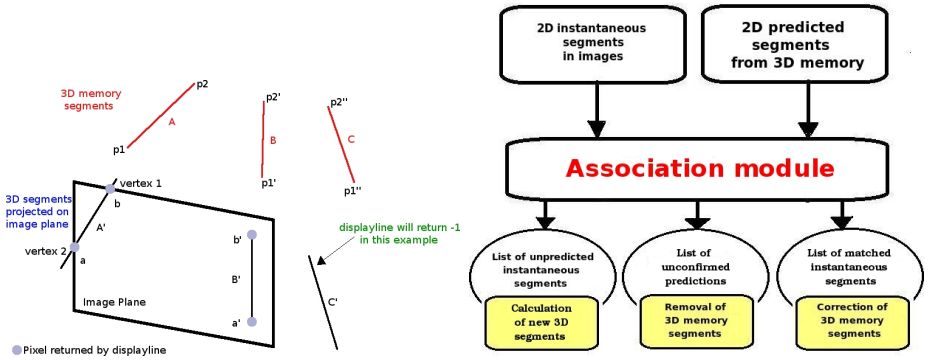


Figure 3.4: 3D projection on the image plane (left) and matching between predicted and observed segments (right)

contents to alleviate the computational cost of image analysis. Before extracting features of the current image, the system predicts inside the 2D image the appearance of those objects already stored in the 3D memory which are visible from the current position. An own *projective geometry* library, relied on the book [Richard and Zisserman, 2003], is used to do this. Each stored 3D visible object is projected on the image plane as shown in Figure 3.4 (left). The system refutes/corroborates such predicted segments, comparing them with those coming from the 2D analysis on observed images (Code 3.5). This comparison provides three sets of segments, as seen in Figure 3.4: those that match with observations, those that do not match, and observed segments that are unpredicted, that is, without an homologous in the 3D memory. Matched segments will be used to update the information of their homologous in 3D memory. Unpredicted

observed segments will be located in 3D and inserted in the visual memory as new 3D segments.

3.2.2 Reconstruction with 3D segments

This module is responsible of obtaining 3D instantaneous information from 2D segments and objects in the current image. To do this we rely on the idea of *ground-hypothesis*, assuming that all the objects are flat on the floor. This simplifying assumption makes it easier to estimate the third dimension from a single camera. This module can be replaced with other 3D techniques and full 3D estimation in case of using a stereo pair.

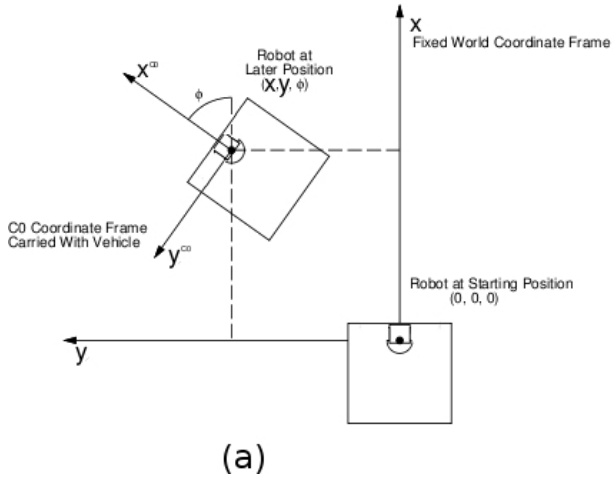
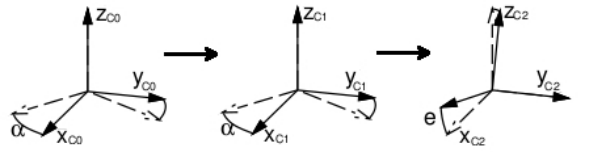


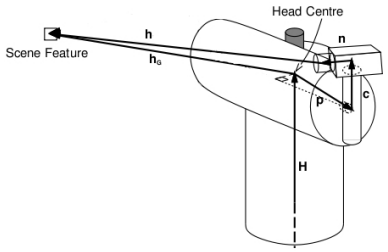
Figure 3.5: C0 coordinate system

There are four relevant 3D coordinate systems in this approach. First, the absolute coordinate system, its origin lies somewhere in the world where the robot is moving. Second, the system located at the base of the robot (Figure 3.5). The robot odometry gives its position and orientation with respect to the absolute system, with some noise. Third, the system relative to the base of the pan-tilt unit to which the camera is attached to (Figure 3.6). It has its own encoders for its position inside the robot at any given time, with pan and tilt movements with respect to the base of the robot. And fourth, the camera relative coordinate system (Figure 3.7), displaced and oriented in a particular mechanical axis from the pan-tilt unit. All



(b)

Figure 3.6: C1 and C2 coordinate system



(c)

Figure 3.7: C3 coordinate system

this mathematical devolpments are described in detail in Section 4.2.3.

The visual memory is intended to be local and contains accurate relative position of objects around the robot, despite their global world coordinates are wrong. For visual memory purposes the robot position is taken from the encoders, so it accumulates error as the robot moves along and deviates from the real robot location in the absolute coordinate system. The segment and object positions computed from images take into account such robot location. This way, the segment and object positions in visual memory are absolute, accumulate error and deviate from their real coordinates in the world, but subtracting the robot position measured from the encoders their relative position are pretty accurate. There is no problem with this as long as the visual memory does not intend to be a global map of the robot scenario. The visual memory is going to be used from the robot point of view, extracting from it relative coordinates of ob-

jects: for local navigation and as a virtual observation for the localization algorithms.

Once the 3D segments are got, and before including them on the 3D memory, some post-processing is needed to avoid duplicates in memory due to noise in the images (Code 3.6). This post-processing compares the relative position between segments, as well as its orientation and proximity, maybe merging some of them. The output is a set of observed 3D segments situated on the robot coordinate system. Figure 3.8 shows the segments detected in the current image, the predicted segments from the current position, and all the objects (parallelograms and arrows) formed by these segments and recognized by the system in the 3D observed scene.

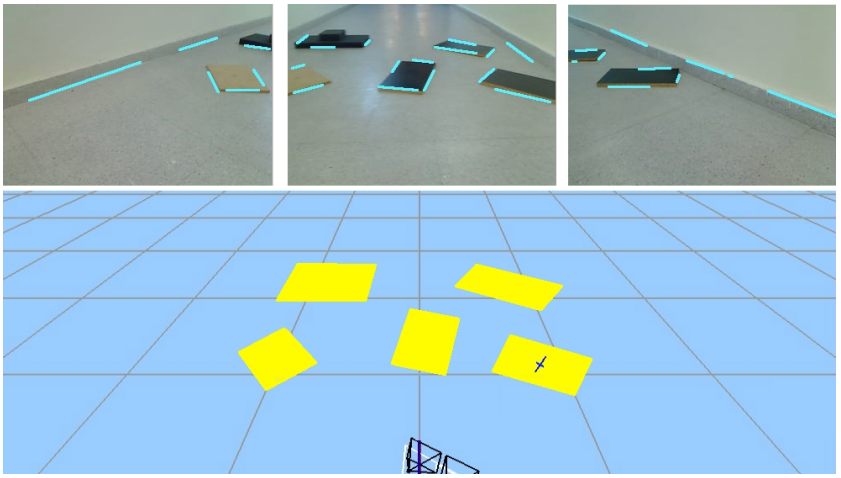


Figure 3.8: Scene situation with three instantaneous images and 3D scene reconstruction

3.2.3 Inserting segments into the 3D visual memory

3D visual memory comprises a dynamic set of lists which stores information about the different types of elements present in the scene (position, type or color). The most basic element is the 3D segment. The visual memory also can establish relationships between them to make up more complex elements such as human faces, arrows, parallelograms, triangles, circles or other objects (Code 3.7).

As mentioned before, the 2D analysis returns different subsets of segments, as the result of comparison between observed and predicted segments from 3D memory. If a segment is identified in the current image and it does not match the predictions, the system creates a new one in 3D. For matched segments they are located in 3D merged with the 3D segments already stored in the visual memory. They will be nearby segments with similar orientation and so the system combines these segments into a new one taking the longest length of its predecessors, and the orientation of the more recent, as probably it is more consistent with reality (the older ones tend to have more noise due to errors in robot odometry). The 3D segments have an attribute named *uncertainty* which increases whilst the segment remains in memory and is also taken into account and updated (Code 3.8). To make this fusion process computationally lighter, the system has a 3D segment cache of the full 3D segment collection with only the segments close to the robot (in a radius of 4m).

As it will be described later (Section 3.3.4), the 3D segments have an attribute named *life* which decreases whilst the segment remains in memory and is not matched with any observation. Every time there is a matching, the *life* of the corresponding 3D segment is increased. If the uncertainty on a 3D segment falls below a given threshold it is deleted from visual memory.

3.2.4 Complex primitives in visual memory

Visual memory manages human faces, simple 3D segments and other primitives like parallelograms and arrows.

3.2.4.1 Human faces

Human faces are detected using the *Haar Classifier* offered by OpenCV (Code 3.9). This classifier is a machine learning based approach, an algorithm created by Paul Viola and Michael Jones ([Viola and Jones., 2001]) which is trained from many positive images (with faces) and negatives images (without faces). Figure 3.9 shows an example of tracking paying attention to a single face. Figure 3.10 shows an example doing the same

but with several faces in the surrounding scene.

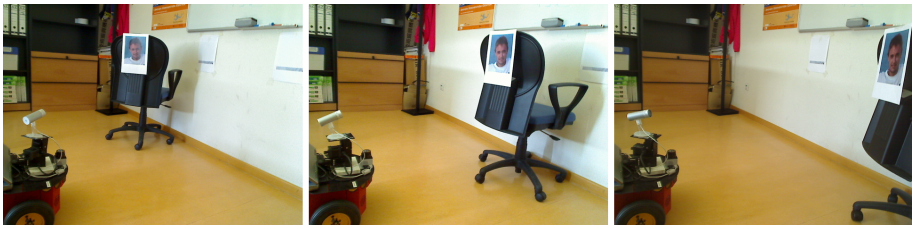


Figure 3.9: Tracking and paying attention to a single face

Figure 3.10: Tracking and paying attention to several faces

3.2.4.2 Segments and parallelograms

The segments and their corresponding vertices are used to detect parallelograms checking the connection between them and the parallelism conditions. The analysis of the angles formed by each segment provides information about how the segments are connected to each other. The visual memory can estimate the position of a possible fourth vertex using the information about edges and the other three vertices. In addition, the parallelogram primitive can be used to merge incomplete or intermittent segments (Code 3.6). This capability makes the algorithm robust against occlusions, which occur frequently in the real world.

Figure 3.11 illustrates an example of occlusion that is successfully solved by the algorithm. Figure 3.11-(b) shows the observation of incomplete parallelograms. The results of reconstruction of parallelograms can be seen in Figure 3.11-(a), with the extracted four parallelograms spread on the floor. The robot, after several snapshots with incomplete parallelograms, captures the real ones in 3D avoiding the noise in the observations. In this example, one threshold of Solis detection algorithm was increased to demonstrate the robustness of the algorithm against noise and incomplete detections of complex objects in the scene. From the input image in Figure 3.11-b, there are many noisy 2D instantaneous segments extracted (Figure 3.11-c) and therefore in the corresponding 3D segments (Figure 3.11-a), but parallelograms are perfectly detected in the visual memory (red objects in Figure 3.11-a).

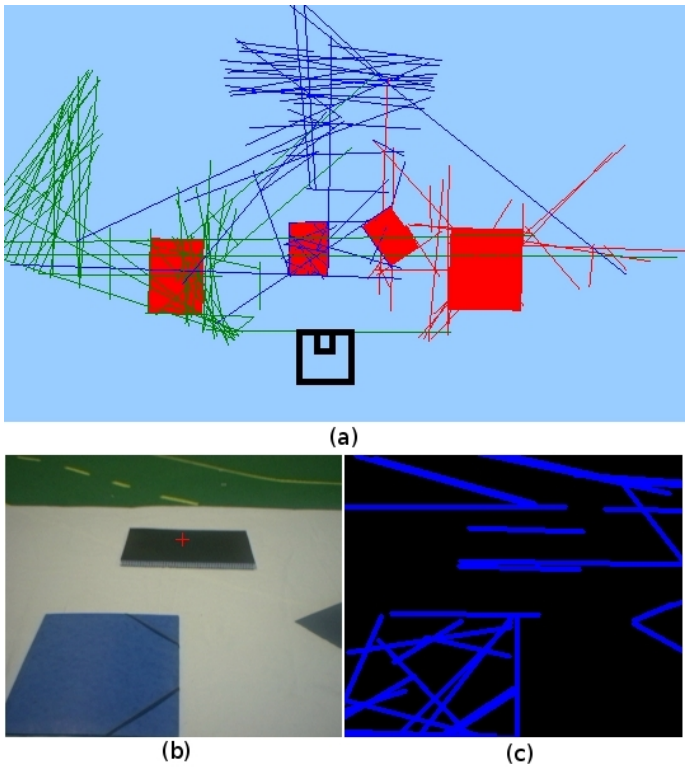


Figure 3.11: Complex primitives in visual memory: parallelograms with occlusion

3.2.4.3 Arrows

Similarly, other objects such as arrows can be abstracted (Code 3.10). These objects are crucial to the system, because they help the robot to navigate in the direction indicated by the arrow (Figure 3.8).

3.3 Visual attention

The second building block of the visual perception system is the visual attention. It uses two main object attributes: *salience* and *life* to decide where to look at every moment and to forget objects when they disappear from the scene (all the visual attention system objects initial attributes are shown in Code 3.11). In addition, this block includes a mechanism to control the camera movements for object tracking and exploring of new unknown areas from the scene.

3.3.1 Gaze control: salience dynamics

Each object in the visual memory has its own 3D coordinates in the scene. It is desirable to control the movement of the pan-tilt unit towards that position periodically in order to reobserve that object. To do so, the dynamics of salience and attention points is introduced. The position of different 3D segments and objects in the visual memory are attention candidate points. Each one has a salience that grows over time and vanishes every time that element is visited, providing so the mechanism of inhibition of return described in Section 2.2, following the equation 3.1 (Code 3.8).

$$Salience(t) = \begin{cases} 0 & \text{if object attended} \\ Salience(t-1) + 1 & \text{otherwise} \end{cases} \tag{3.1}$$

Equation 3.1: Objects saliency

The system continuously computes the salience of all attention points and chooses the most salient one to control the gaze. The pan and tilt orders that make the camera look at it are then computed and commanded

to the pan-tilt unit. When a point is visited, its salience is set to 0. An element that the system has not visited recently gets more salience than one which has just been attended. If the salience is low, it will not be visited now. The system is thus similar to the behavior of a human eye, as pointed by biology studies [Itti and Koch., 2005]: when the eye responds to a stimulus that appears in a position that was previously treated, the reaction time is usually higher than when the stimulus appears in a new position.

After a while the most salient element is recomputed again and so the focus of attention is changed, implementing a kind of time sharing gaze control. The designed algorithm allows so to alternate the focus of the camera between the different objects in the scene according to their salience. It is assumed that an object will be found near the location where it was previously observed the last time. If the object motion were too fast then the reobservations would not match with the current object location.

In the system all objects have the same slope in the saliency dynamics, the same preference of attention and so all of them are observed with the same frequency. Objects could have different priorities whether rates of growth of salience is assigned, causing the pan-tilt unit to look more times at the objects which salience grows faster.

3.3.2 Tracking of a focused object

When the gaze control chooses the attention point of a given object, the system will look at it for a certain time (3 seconds), tracking it if it moves spatially. For this tracking it is used two proportional controllers to command the pan and tilt speeds and thus continually keep that object in the center of the image, previously translated from Cartesian coordinates to Polar coordinates (Code 3.12).

The controller follows the equations 3.2 and 3.3, where: K_p is the P control gain, T_t is the Tilt of the target, T is the current Tilt, P_t is the Pan of the target, P is the current Pan, M_t is the maximum Tilt acceptable error, M_p is the maximum Pan acceptable error, ϵ_p is $(P_t - P)$ and ϵ_t is $(T_t - T)$.

$$v(Pan) = \begin{cases} 0 & \text{if} \quad \epsilon_p < 0.3 \\ K_p \cdot (P_t - P) & \text{if} \quad 0.3 \leq \epsilon_p < M_p \\ K_p \cdot M_p & \text{if} \quad M_p < \epsilon_p \end{cases} \tag{3.2}$$

Equation 3.2: Pan speed controller

$$v(Tilt) = \begin{cases} 0 & \text{if} \quad \epsilon_t < 0.1 \\ K_p \cdot (T_t - T) & \text{if} \quad 0.1 \leq \epsilon_t < M_t \\ K_p \cdot M_t & \text{if} \quad M_t < \epsilon_t \end{cases} \tag{3.3}$$

Equation 3.3: Tilt speed controller

3.3.3 Exploring new areas of interest

The robot capability to look for new objects in the scene is interesting. This search is especially convenient at the beginning of operation, when there are many unknown areas of the scene around the robot with objects of interest. For that search the system periodically inserts (every *forcedSearchTime*) attention points with high salience in the visual memory. Due to its high salience they will be quickly visited with the camera and so that location checked whether any object of interest is found around it. In such a case that object will enter into the visual memory and into the regular gaze sharing.

The scanning points can be of two types: random and systematic ones. Random points are distributed uniformly within the pan-tilt range. Systematic scanning points follow a regular pattern to finally cover the whole scene around the robot. With them, the system ensures that eventually all areas of the scene will be visited.

There will be a proliferation of points of exploration in the beginning, when there are few objects in memory to reobserve. As the robot discovers objects, the desire to explore new areas will decrease in proportion to the number of already detected objects.

3.3.4 Representation of the environment: life dynamics

As already mentioned in previous sections, the objects may eventually disappear from the scene, and then they should be removed from the mem-

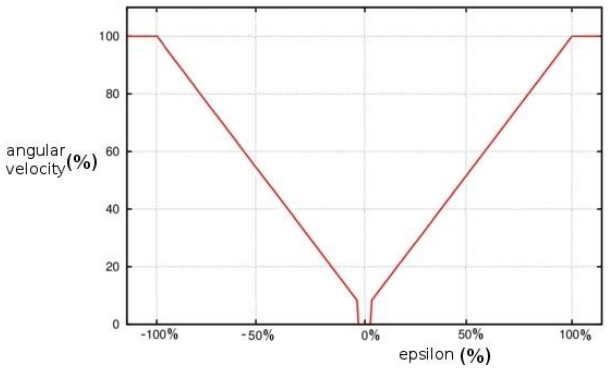


Figure 3.12: P-controller mechanism

ory in order to maintain coherence between the representation of the scene and the reality. To forget such old elements, it was implemented the *life* dynamics that follows the equation 3.4.

$$Life(t) = \begin{cases} min(MAX_{LIFE}, Life(t-1) + \Delta) \\ \qquad \text{if object observed} \\ Life(t-1) - 1 \\ \qquad \text{otherwise} \end{cases} \tag{3.4}$$

Equation 3.4: Life dynamics

Where Δ is a bonus factor used when the element is a human face, because this kind of element it is considered more relevant for the visual attention system.

Life of unobserved 3D segments or objects decreases over time. Every time an object or 3D segment is observed in the images (just because the gaze control visits it or visits one near object), its *life* increases, with a maximum saturation limit. This way when the *life* of an object exceeds a certain threshold, that means it is still on the scene, whereas when it is below it that means it has gone and so, is deleted from visual memory.

3.3.5 Attention module operation

The visual attention module is fully *bottom-up*. The objects surrounding the robot guide the movements of the camera, just to reobserve them,

to track them or to explore the environment looking for them. Periodically the system updates the salience and life attributes of the objects that are already stored in memory following previous equations. It checks whether any of them is already outdated, because its life is below a certain threshold. If not, it increases its salience and reduces its life.

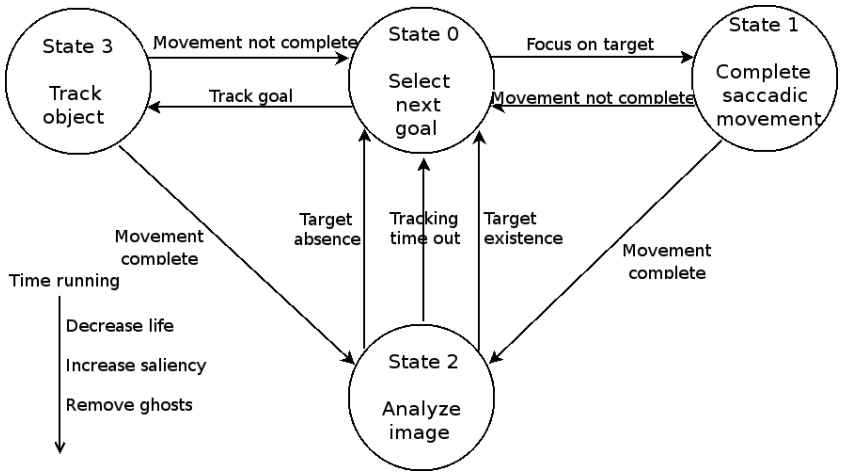


Figure 3.13: Finite state machine attention system

It was implemented following a *finite-state-machine* (Figure 3.13) that determines when to execute the different steps of the algorithm: select next goal (state 0), complete the saccadic movement (state 1), analyze image (state 2) and track the object (state 3). In the initial state the system asks whether there is any attention point to look at (in case it has an object previously stored in memory) or not. If so, it goes to state 1. If not, it inserts a new scanning attention point into memory and goes back to state 0. In state 1 the task is to complete the movement towards the target position. Once there, the automaton goes to stage 2 and it analyzes whether there are relevant objects in the images or not. After a while it returns to state 0 and starts again.

3.4 Localization

A new approach to solve robot self-localization was specifically designed to deal with symmetries. It is an evolutionary algorithm, a type of meta-heuristic optimization algorithm that is inspired by the biological evolution.

In this kind of algorithms, candidate solutions are so-called "individuals", which belong to a population that evolves over time using genetic operators, such as mutation or crossover. Each individual is a tentative robot position (X, Y, θ) and it is evaluated with a quality function which calculates its "health", that is, a measure to know how good its localization is with regard to the optimal solution. It was defined two different health functions, one based on instantaneous measurements of robot sensors and another one based on the visual memory contents.

Races are sets of individuals around a given location, they perform a fine-grain search around it. The algorithm has R races which compete among each other to be the race containing the best pose estimation. Each race has several associated parameters like the number of iterations without being deleted, the number of iterations containing the best pose estimation, etc.

The main idea of the algorithm consists of keeping several races competing among each other in several likely positions. In case of symmetries from observations, the algorithm will create new races on various positions where the robot might be located. After some iterations, predictably, new observations will provide information to reject most of the races and the algorithm will obtain the real robot pose from the best race. On each iteration of the algorithm it performs several steps to estimate the current robot pose (Figure 3.14). They are described in detail in the following sections.

- Health race calculation using the information obtained after analyzing images.
- Explorer creation: New individuals are randomly spread; explorers, with the aim to find new candidate positions where new races could

be created.

- Race management: To create, merge or delete races depending on their current state.
- Race evolution: To evolve each race by using genetic operators. Besides, if the robot is moving, all races are up-to-date taking into account this movement.
- After calculating each race health, one of them is selected to set the current robot pose estimation.

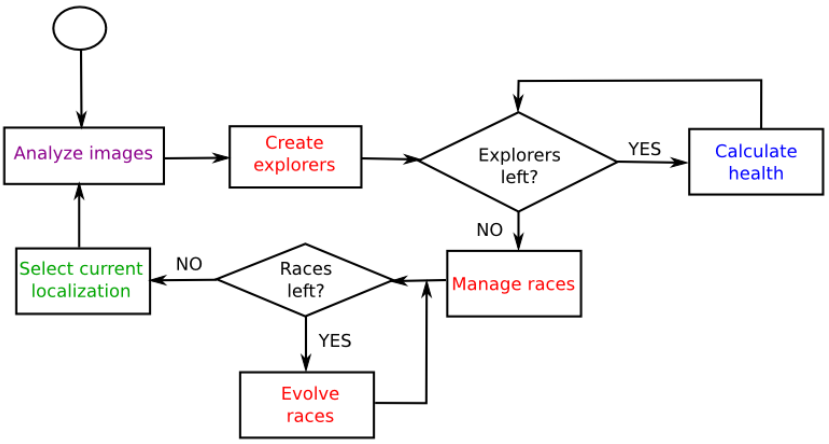


Figure 3.14: Basic diagram of evolutionary algorithm

We have created two branches of the proposed evolutive localization algorithm. The first one takes data directly from the current camera image. Its health function is described in Section 3.4.2. The second one takes data from the local visual memory, its 3D segments. Its health function is described in Section 3.4.3. Current implementation of the visual memory component only creates 3D segments lying on the floor, but the localization algorithm is ready for accepting 3D segments in any position and orientation.

3.4.1 Analyzing images

When the localization algorithm uses the instantaneous camera images, it first detects lines inside the images following the same technique described in Section 3.2.1. This time the algorithm does not discard segments over the horizon (Code 3.13), as it does not require that objects lie on the floor. Also, some post-processing in the image is performed to clean and refine the segments. The post-processing associates a label to each segment depending on the main colors at both sides of the segment. Segments with unknown type are rejected. After labelling each segment, the algorithm tries to merge segments with the same type if their extremes are close to each other (Code 3.6), joining consecutive segments together (Figure 3.15). Too small segments are discarded.

Figure 3.15: Image analysis before merging (left) and after merging (right)

Instead of using these lines directly as input data, the algorithm divides them into sampling points to make the comparison between lines easier, as it will be explained in Section 3.4.2. It is created a grid with different cell sizes and a single new point is saved where the detected lines intersect with this grid (Figure 3.16). The size of this grid cells changes because it is required to analyze the upper part of the image more deeply than the lower one, since further objects will be at the top of the image and its resolution will be smaller. All these selected sampling points will be the input data to calculate the health of each individual.

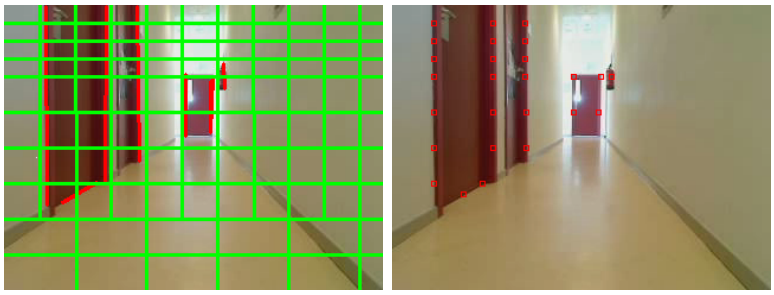


Figure 3.16: Image grid (left) and points selected (right)

3.4.2 Health calculation from real-time images

The health of an individual placed at certain location is computed comparing the theoretical set of visible objects and segments from that location (theoretical observation) with objects and segments currently observed (real observation). The more similar the predicted segments and the observed ones are, the more likely such location is the correct one.

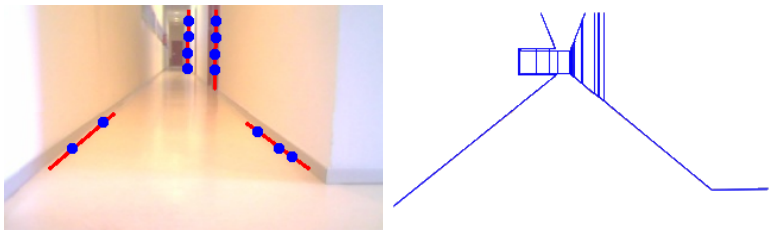


Figure 3.17: Detected lines in current image and theoretical image

The theoretical observations are generated ad-hoc for each particular location, projecting lines from the environment map into the camera placed at that location (Figure 3.17). It contains the lines the robot would see if it were placed at that location (Figure 3.17 (right)). It is assumed that the map of the environment is known.

For each sampling point in the observed lines (Figure 3.17 (left)) the Euclidean distance d_i in pixels to the closest theoretical line with the same label is computed (Code 3.14). After calculating d_i for all points, the individual's health is computed as the average distance, following the equation

3.5, where N is the number of points and M is the maximum distance allowed in pixels (set to 50 pixels for a 320x240 image size).

$$H = 1 - \frac{\sum_{i=0}^{N} \frac{d_i}{N}}{M}$$

Equation 3.5: Points health

We will show in Section 3.5.2.4 several experiments to analyze the health function behavior in different situations.

3.4.3 Health calculation with Visual Memory

It is not necessary to analyze each image, in case of using the visual memory; just the current visual memory contents from the `active_visual_memory` component, that is, the set of 3D segments inside, relative to the robot. So it is not able to compare lines in image as it was done previously. Besides, it has to be taken into account that lines may not be detected completely or they may be divided into several small lines. Thus, the equation 3.5 is followed to calculate the health of an individual, where all lines belonging to the visual memory are covered. For each line its extremes are obtained and calculated the Euclidean distances d_{j_s} and d_{j_e} to the closest theoretical line with the same label. This is similar to health function with instantaneous images but in 3D. After calculating d_{j_s} and d_{j_e} for each line, the health can be calculated as follows, where N is the number of lines and M is the maximum distance allowed in meters (set to 0.5 meters).

$$H = 1 - \frac{\sum_{i=0}^{N} \frac{(\frac{d_{j_s} + d_{j_e}}{2})}{N}}{M}$$

Equation 3.6: Lines health

3.4.4 Explorer creation

Explorers are individuals that do not belong to any race and that try to find likely positions with good health. There are two ways to spread

explorers around the environment: randomly or following a predesigned pattern of search positions. In order to be truly general and avoid the over-fitting of the algorithm the random approach was chosen.

When explorers creation is performed, the algorithm creates E explorer individuals, calculates their health and arranges them according to their health. Then, the best M explorers are promoted to become a candidate to create a new race. The number of explorers created changes dynamically depending on the current algorithm status: if the robot is lost, E is increased, and decreased it if its location is reliable. Besides, since explorers creation is time consuming, when current location is reliable the explorer creation is not executed at each algorithm iteration, but once every certain iterations.

3.4.5 Race management

In the long term the algorithm manages several races and needs a policy to decide when to create a new race, delete it, or merge two races. The algorithm uses two parameters of each race for that: "victories" and "age". A race increases its "victory" counter in an iteration when its health is higher than that of the rest of races. At the race creation this parameter is set to 0. This number can also decrease if the winner in a given iteration is another race. This parameter is useful to select the race that finally sets the current robot pose estimation in each iteration. The age parameter shows the number of algorithm iterations since its creation. It was created with two objectives. First, it preserves new races from dying too soon to avoid creating races that will be deleted in the next iteration. When a race is created this race can not be deleted or replaced (although it may be merged) until its age reaches $RACELIFE$, whose value that works best is being 3. Second, it avoids deleting a race because of wrong sensor information. If a race has had the highest health in an iteration, the algorithm will not delete it at least until 9 iterations after that. This provides some stability to races and avoids the continuous creation and deletion of races.

The followed approach has a maximum number or races R that avoids the exponential increasing of computation time related to the number of

races. Whenever the explorers creation is executed and there are new candidates to become a race, we have to decide when to create a new race and when to replace an existing one. If the maximum number of races R has not been reached, for each candidate we find out if an existing race is located in the same position (X, Y, θ). In such case, we do not try to create a new race, but we assume that the existing race already represents this candidate, and its age is increased. If candidates are innovative enough, we create a new race. If the maximum number of races is reached, the candidate will replace a victim race if the health of the candidate is greater than that of the victim and the victim's victories parameter falls below 0. If no race can be replaced, candidates are ignored.

In case two races evolving towards the same location, we consider that they have led to the same solution, so we merge them. This merging consists of deleting the race with lower victories. If both have the same victories, we keep the best race according to its health.

A race will be deleted when its victories are 0 and its health is below 0.6. In such a case the race is not at the real robot pose anymore, the algorithm considers it wrong and deletes it.

3.4.6 Race evolution

When a race is created from an explorer, all its individuals are created applying a random thermal noise to the explorer who created the race. From then on, in the next iterations, its individuals evolve through three genetic operators: elitism, crossover and mutation. With elitism, the algorithm selects the best individuals of each race, arranging them according to their health. They are saved in the next iteration without any change. With crossover, the algorithm randomly selects several pairs of individuals and calculates their average with their values (X, Y, θ) for the next iteration. With mutation, the algorithm selects an individual randomly and applies a thermal noise to its position and orientation.

Besides, in case the robot has moved since the last iteration, we apply a motion operator to all races and individuals at the beginning of each iteration using robot odometry. Once all the individuals of each race have evolved, the algorithm calculates the final pose of the race as the average

of its elitist individuals, to avoid abrupt changes.

3.4.7 Selecting the robot pose estimation

After evaluating all the existing races, the algorithm chooses one of them in each iteration to be the current pose of the robot. The selected race will be the one with more victories and its pose will determine the robot pose calculated by the algorithm.

The first step is selecting the race with greatest health in the current iteration (R_i). If the race selected was the same in the previous iteration (R_p), we increase the victories of R_i and we decrease the victories of the rest. However, if R_i and R_p are different, we only change the races victories if the difference between R_i health and R_p health is greater enough. This distinction is made because we want race changing to be difficult if R_p was the selected race during a lot of iterations. With this behavior, we try to help the races which were selected in the previous iterations and we only change the winner race when the health difference is big enough (what would mean that the current localization is wrong).

3.5 Experiments

To verify the different approaches of visual memory, visual attention and visual localization, several experiments were conducted. The experimental real platforms were an ActivMedia Pioneer 2DX robot equipped with a Logitech Autofocus camera (2 megapixels) and a Nao Robot from Aldebaran Robotics (v3 model). Besides, we have used Gazebo as robot simulator. All the experiments are implemented on C++ with Jderobot robotics software platform, which uses ICE as communication middle-ware.

The visual system is primarily intended as underlying technology for service robot applications at real people homes. The service robot using vision will need to perceive obstacles around it, even beyond the current field of view, and to know its position in the house to lunch proper actions. For the real tests we have used the attentive visual memory and localization algorithm in an office scenario with doors, corridors, lights, etc. without any robot-specific landmark. We have preferred this real life environment,

as is, over simplified ones like, for instance, those in the RoboCup. Moreover, the features of RoboCup competition, with highly dynamic situations around the robot, are different from those at real homes ([Perdices et al., 2010]). The proposed algorithms were tested so far on a moving robot in static or low dynamic scenarios, with few moving objects around or slow movements.

3.5.1 Attentive visual memory experiments

3.5.1.1 Robot in the middle of a room

For the first experiment, the robot is in the middle of a room (see Figure 3.18-a). Then the robot turns around itself. Figure 3.18-b shows a instantaneous view from the room, where robot is able to detect a simple line on the floor. After a few seconds, the robot has turned a full circle, having stored all the information about its surrounding environment. Thus, we can see in Figure 3.18-c, how the short-term memory provides more information than an instantaneous image.

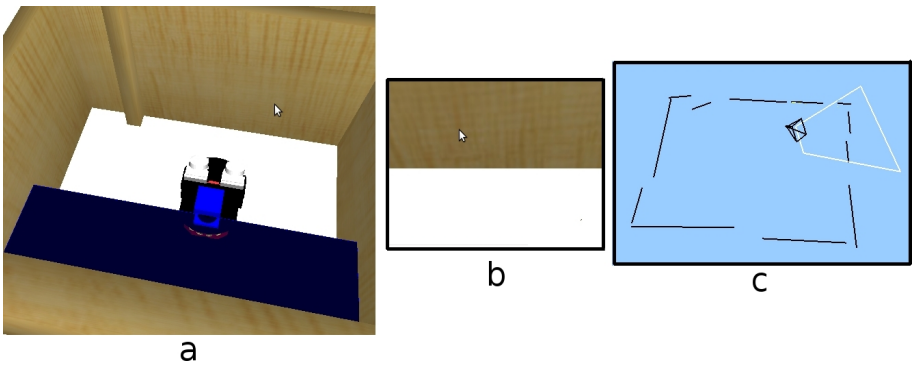


Figure 3.18: (a) Situation; (b) Instantaneous image; (c) Short-term memory

3.5.1.2 Robot navigating a curve

This experiment shows how the robot is unable to navigate using only the instantaneous information received from the camera. The situation is shown in Figure 3.19-a, the robot approaches to a curved area, while navigating through a corridor. If the robot used only instantaneous images

(Figure 3.19-b), it would be able to see only just some lines in front of itself (Figure 3.20-a), but with short-term memory it can observe that the path in front of itself is a curve (Figure 3.20-b). In addition, robot can quickly explore its surrounding environment thanks to the visual attention mechanism, which forces the system to explore unknown areas of the environment.

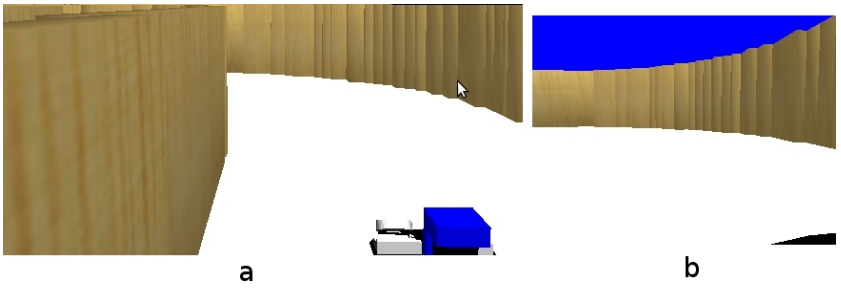


Figure 3.19: (a) Situation; (b) Current on-board image

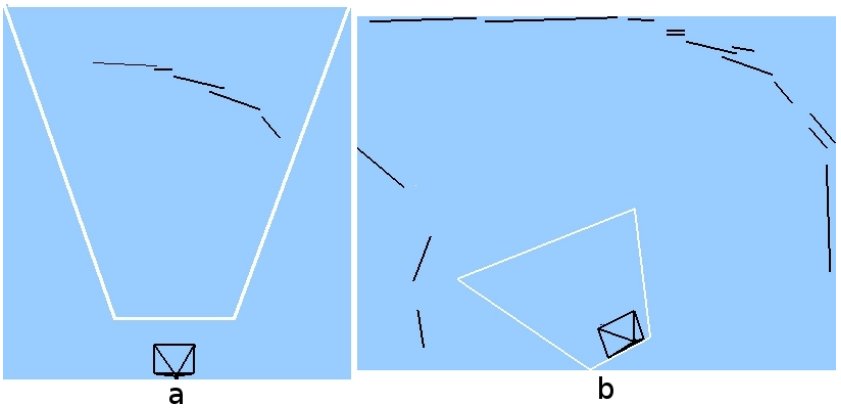


Figure 3.20: (a) Information in current field of view; (b) Short-term memory

3.5.1.3 Robot occlusions

Here, the situation is presented to solve a temporary occlusion. This happens very often in real environments where there are dynamic objects which can obstruct the robot field of view.

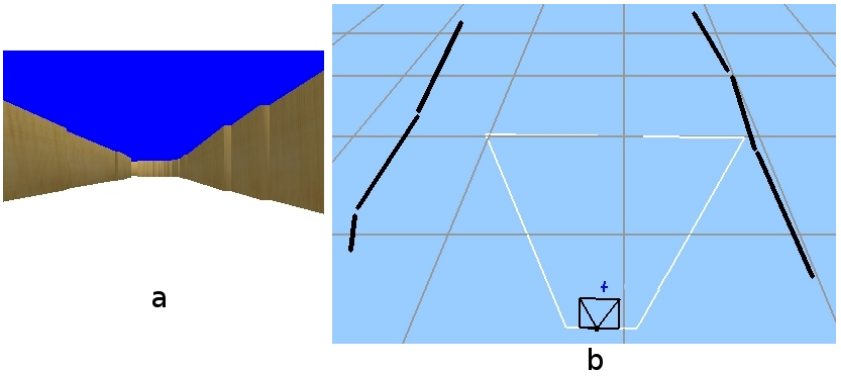


Figure 3.21: (a) Situation; (b) Short-term memory got after a while

The initial situation is shown in Figure 3.21-a. After a few seconds, robot has recovered environment information thanks to the short-term memory and the visual attention system (as displayed in Figure 3.21-b).

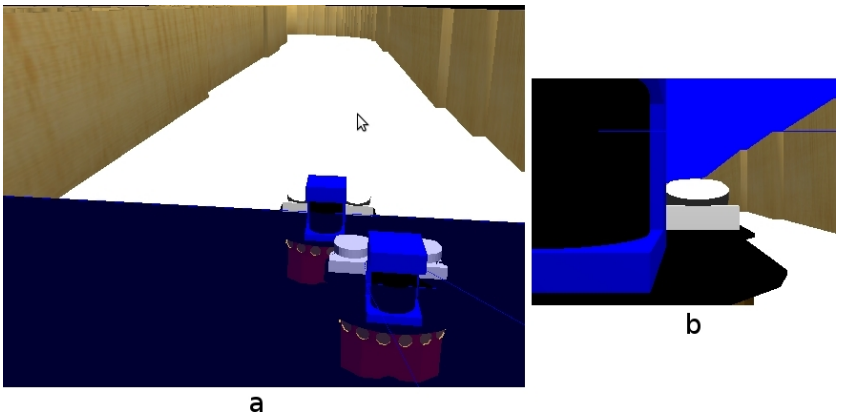


Figure 3.22: (a) Situation; (b) Field of view

Then another robot appears, as shown in Figure 3.22-a, temporarily occluding the field of view of the robot (Figure 3.22-b), so the robot is unable to see anything. This situation continues for some time until the second robot moves away from our robot (Figure 3.23-a,b).

This situation is solved by the system with the persistence of the short-term memory, and so, the robot can make control decisions taking into account information of areas beyond the robot that is occluding its camera.

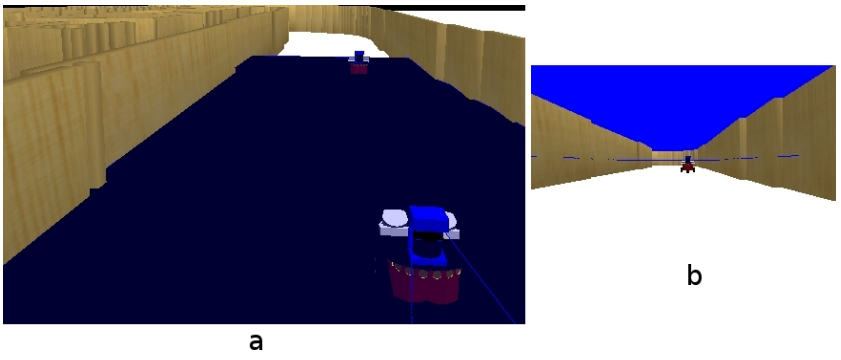


Figure 3.23: (a) Situation; (b) On board current image

As we presented before, the memory is continuously refreshed and updated over time. If it is inconsistent, that is, what the robot sees does not match the information stored in memory, the system has some persistence before changing the memory contents.

3.5.1.4 Attentive visual memory on a humanoid robot

In this experiment we have tested the whole system including visual memory, visual attention algorithm and visual localization on a real robotic platform such as Nao humanoid robot. The robot is in the middle of the department corridor gathering information about its environment. It is able to move autonomously around the corridor; furthermore, it is moving its neck in order to detect all segments in a few seconds. We can see in Figure 3.24 some snapshots and the short-term memory built with them.

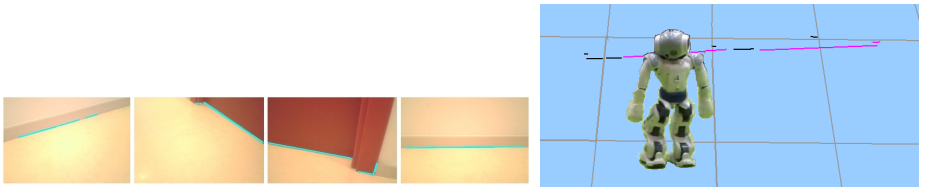


Figure 3.24: Visual memory with 3D segments coming from four images of robot surroundings

The local visual memory updating and the attention module run in it-

erations. The average iteration time including all computations (2D image processing, matching with predictions, 3D reconstruction, segment insertion, complex primitives management, etc.) was 80 ms.

3.5.2 Visual localization experiments

We have performed several experiments to validate the evolutionary algorithm as well, especially with real robots. The localization algorithm is map-based. The map in these experiments was introduced in the robot as a collection of 3D segments at certain positions, stored in a file.

3.5.2.1 Testing MCL algorithm behavior

We have implemented a Monte Carlo localization algorithm [Fox et al., 1999] with the same health calculation (observation model) and motion operator (motion model) that we explained in Section 3.4. We performed a theoretical experiment to analyze MCL behavior within symmetric environments. We placed two particle populations with the same number of particles (125) in front of two identical doors of the environment, placing each populations at the same distance from each door and updating MCL particles with the same observation (Figure 3.25):

Figure 3.25: Observation taken to update MCL particles

Since both populations are located at the same distance from each door, and observations are identical, MCL should keep both populations until new information is obtained. However we show in Figure 3.26 the real

behavior of the algorithm, where one of the populations is always ruled out after a while. In vertical axis the number of particles of each race is displayed (different colors for different runs). The horizontal axis displays the iteration number. In the experiments, MCL was able to keep both populations with enough particles only once for more 250 iterations (about 30 seconds). In most of the runs one of the two populations eventually gained all the particles of the algorithm.

This bias towards one of the populations happens because MCL picks up the particles of the next population randomly in the "roulette" step. By chance, this randomness may choose more particles from one population than from the other, and then this last population has even fewer probability of providing samples at the next iteration, decreasing gradually the particles coming from it, until such population is finally ruled out.

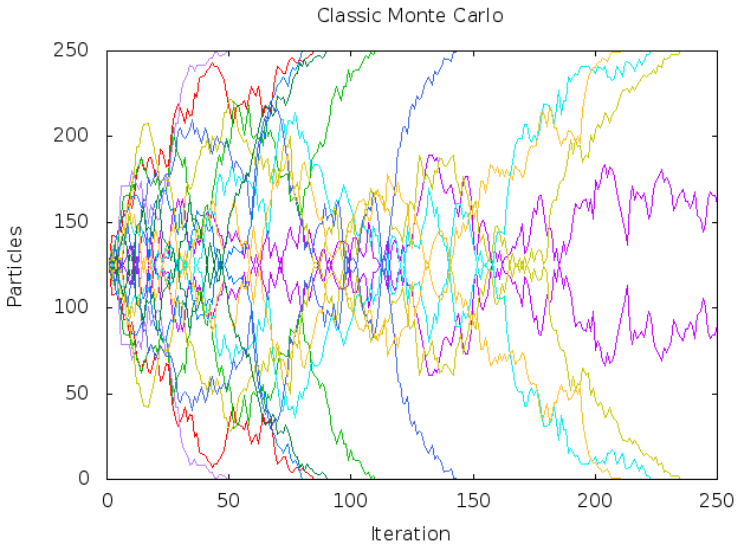


Figure 3.26: Monte Carlo particles evolution

When we performed this experiment with the evolutive approach, it created a race in front of each door and kept the races separated from each other all the time. This way, both races are really kept until new information is provided.

There are other solutions to solve this bias in the MCL, for instance

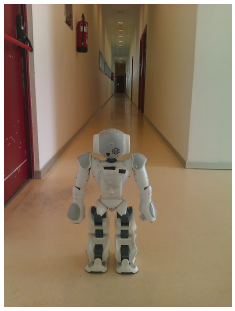
using omnidirectional cameras. We have preferred to improve the localization algorithm and still use regular monocular cameras as they appear more frequently in the robot equipment and other perceptive algorithms work with and require these regular cameras.

3.5.2.2 Typical execution in humanoid robot

The evolutionary algorithm has several parameters to be configured, such as the maximum number of races (10), the number of explorers (30), and the percentages of the genetic operators previously explained (20% elitism, 20% crossover and 60% mutation). The maximum number of races was a crucial factor. A high number of races improves the accuracy of the calculated localization, but it also increases the algorithm execution time. We have selected a value, 10, that offers a good balance between efficiency and accuracy.

The first experiment was performed with a real Nao robot travelling through a corridor (Figure 3.27). It shows how the algorithm is able to follow the real movement of the robot starting on a known position. At first, the robot is located in a known position, afterwards we move the robot around the environment and measure its localization error. The red line in Figure 3.27 shows the calculated positions, the green line the real robot path, and the brown area is the measured error. The average error was 11,8 cm and 2,1 degrees. The algorithm is able to follow the robot movement even when its instantaneous observations do not provide enough information due to robot odometry. Besides, we can emphasize that the trajectory followed by the robot is very stable and is always close to the real location of the robot. The ground truth location of the robot was calculated manually measuring the distance from the robot to known objects, such as doors or corridor lines. These distances were calculated each 5 seconds and interpolated in between.

The evolutionary visual localization algorithm runs in iterations. The average iteration time including all processing (health computation, race management, explorer creations, etc.) was 110 ms.



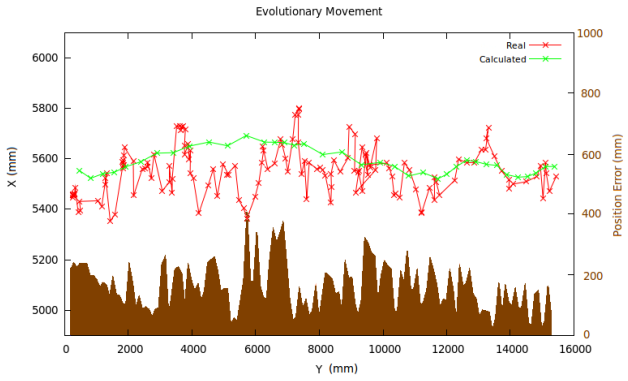


Figure 3.27: Nao robot travelling a corridor for experiments (left) and estimated localization and position error over time (right)

3.5.2.3 Dealing with symmetries and kidnappings

The second experiment (Figure 3.28) shows how the algorithm works with symmetries and kidnappings. At first instant (1.a), we locate the robot in front of a door, so the algorithm creates several races where the robot may be located. The algorithm selects one of them (a wrong one) but keeps another one on the right location. Then the robot moves and obtains more information from the world and finally rules the wrong location out and selects the correct one (at 1.b instant). Afterwards, we kidnap the robot to another location (2.a) and it takes a while until the robot changes its estimation to the new right location. This happens because the location's reliability changes gradually to avoid changing with false positives, but after a while, it changes to the new position (2.b). A second kidnapping is performed (3.a), this case is similar to the first one: first, it selects a wrong localization (3.b), but after some iterations it changes to the correct one (3.c). The average error after selecting the correct race was 22,5 cm and 5,8 degrees. We also measured the time spent until the algorithm calculates a new plausible pose after a kidnapping (recovery time). In this experiment it took 21 secs.

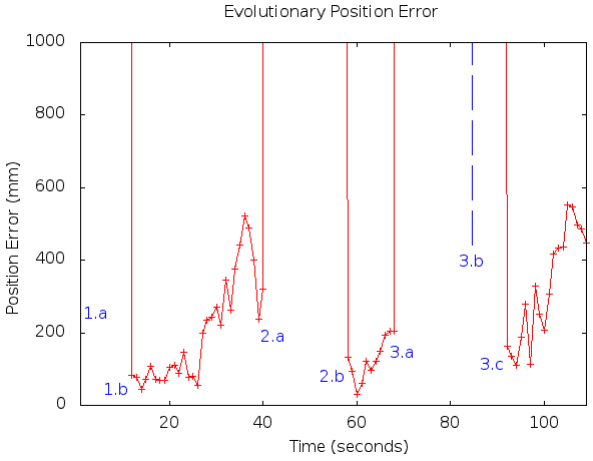


Figure 3.28: Position error over time

3.5.2.4 Health function based on instantaneous images

To validate the health function, we have implemented a debugging mechanism to show graphically the value returned by the health function in different positions. In Figure 3.29 we show the value returned in all positions (X, Y), where red areas are the ones with highest probability and white areas with the lowest. As we can see, the position with highest probability is likely with the input image.

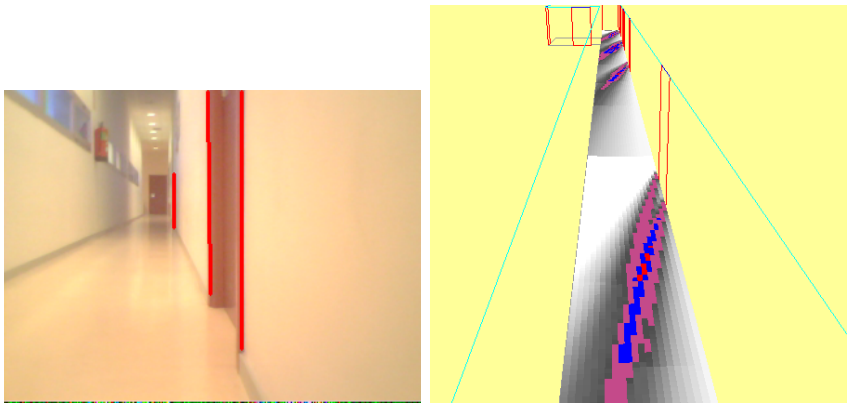


Figure 3.29: Observed image (left) and probabilities calculated with *theta* equals to 0 radians(right)

In case of symmetries we obtain high probabilities in several positions. In Figure 3.30 we show what would happen if we obtained an image in front of a door, then we would obtain high probabilities in front of each door of the environment:

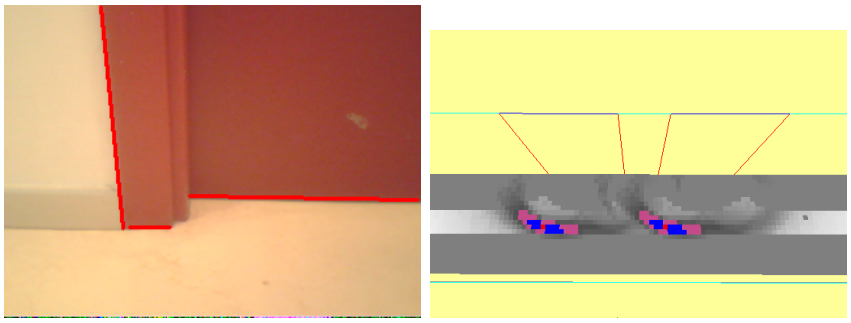


Figure 3.30: Observed image in front of a door (left) and probabilities calculated for any *theta* (right)

The localization algorithm was tested in more complex scenarios, with occlusions and false positives. In case of occlusions the algorithm keeps a good behavior, since the followed approach does not penalize the probability if an object is not detected in the image when it should be. The light negative impact of occlusions is a higher probability in some more field areas (compare Figure 3.31 and Figure 3.32). However, false positives affect very negatively to health function and the calculated position can be totally wrong (compare Figure 3.31 and Figure 3.33).

3.5.2.5 Health function based on visual memory

In case of using visual memory instead of instantaneous images for the health function, the calculated values will be similar to previous health function, but we will get two benefits: there will be less symmetries, because we will get more information about the environment, and temporal occlusions will affect even less to the health function.

We have performed an experiment to compare the localization algorithm with and without visual memory. We placed the robot in front of two consecutive doors but without observing both at the same time. At first, the robot only detects the first door, and afterwards we move the

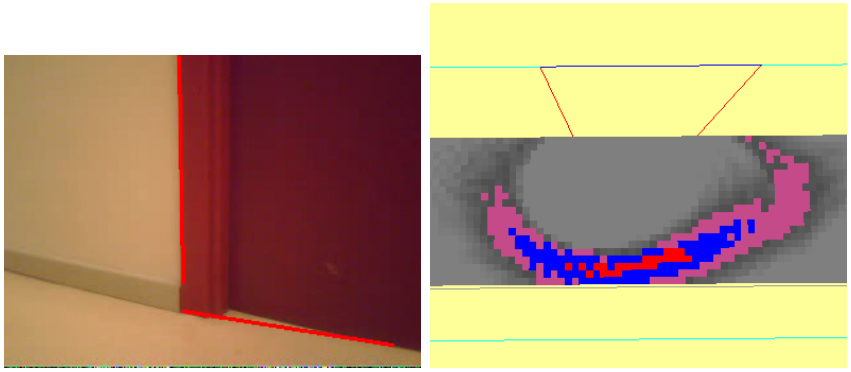


Figure 3.31: Observed image (left) and calculated probabilities without occlusions or false positives for any *theta* (right)

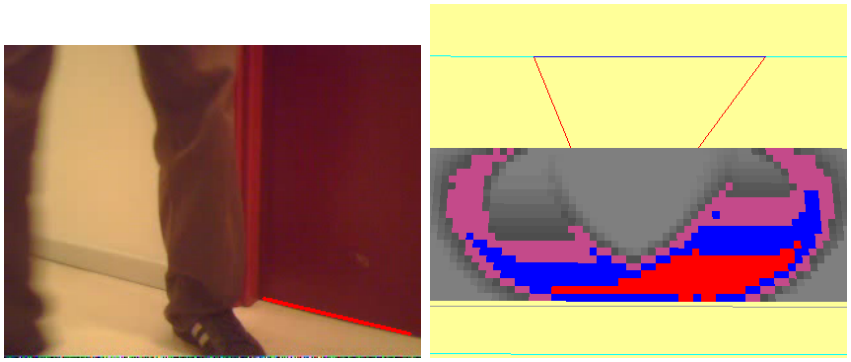


Figure 3.32: Observed image (left) and calculated probabilities with occlusions for any *theta* (right)

robot sideways so that it does not detect the first door any more and it starts observing the second one.

The localization algorithm without visual memory is not able to locate the robot always in the proper place. It creates a race in front of several doors once the first door is detected, but not in front of all of them. When the second door is observed, there may be two possibilities:

- If a race was created in front of two consecutive doors when the first door was observed, the algorithm would choose this race because the second door would fit with that position.

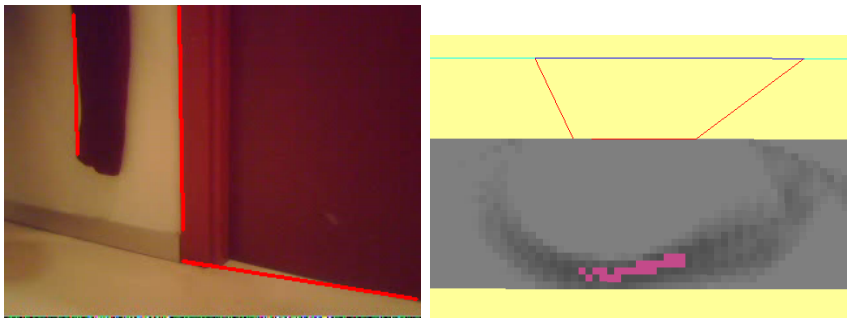


Figure 3.33: Observed image (left) and calculated probabilities with false positives for any *theta* (right)

- If it was not the case, the algorithm would select a race in front of any door. Since it can not remember that there was another door before, then, most of cases it would pick up a wrong position.

However, when the second door is detected using the visual memory, the algorithm would be able to select the correct race even in the second case, since it wouldn't forget the first door. We show in Figure 3.34 the health map and position estimation calculated with the visual memory.

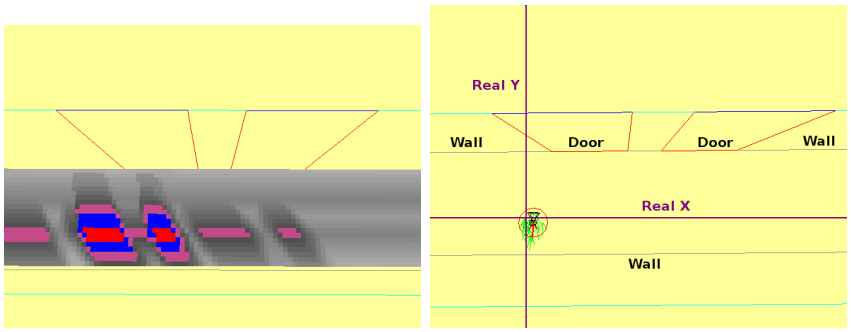


Figure 3.34: Health values for any *theta* (left) and estimated position (right) calculated with visual memory

3.6 Conclusions

In this chapter a visual perception system for autonomous robots was presented. It processes the images from a mobile camera and builds a short term local memory with information about the objects around the robot, even if they lie outside the current field of view of the camera. This visual memory stores 3D segments and simple objects like parallelograms with their associated properties like position, uncertainty (inverse of *life*), color, etc. It allows to make better navigation and even localization decisions since it includes more information than the current image, which can even be temporary occluded.

On the one hand, an overt visual attention mechanism was created to continuously select where the mobile camera should look at. Using a *salience* dynamics and choosing the most salient focus of attention the system shares the gaze control between the need to reobserve objects on the visual memory and the need to explore new areas, providing also inhibition of return.

On the other hand, we have developed a visual self-localization technique that uses an evolutionary algorithm. It keeps a population of particles to represent tentative robot positions and the particle set evolves as new visual information is gathered or with robot movements. It was especially designed to deal with symmetries, grouping particles into races. There is one race for each likely position and inside it individuals do the fine grain search. It can work both with just the current image or the contents of the visual memory.

This visual perception system was validated both on real robots and in simulation. The memory nicely represents the robot surroundings using the images from the mobile camera, which movement is controlled by our attention mechanism. The memory is dynamic but has some persistence to deal with temporary occlusions. The localization works in real time, provides position errors below 15cm and 5 degrees and is robust enough to recover from kidnappings or estimation errors in symmetric environments.

We are working in extending the visual memory to manage stereo pairs and RGB-D sensors as inputs and to deal with objects at any 3D posi-

tion, not just the floor. The localization algorithm is designed and ready for that, but the current visual memory implementation works with one regular camera and does not provide full 3D segments at any position, it assumes all the objects lie on the floor. This limitation also causes that painted lines or color changes on the floor, which are traversable, are wrongly considered as obstacles to avoid. More experiments are also required in more challenging scenarios, with more dynamic obstacles or people around the robot and with faster movements. We are also studying how to deal with more abstract objects like tables and chairs into the visual memory. Regarding localization we are working on introducing a mono-SLAM EKF for each race of the evolutionary algorithm and on improving it to extract localization information from abstract objects, not only 3D points.

```
void Memory::hypothesizeParallelograms () {
  for(it1 = this->controller->segmentMemory.begin(); it1 !=
      this->controller->segmentMemory.end(); it1++) {
    squareFound = false; it2 = it1; it2++;
    while ((it2 != this->controller->segmentMemory.end()) &&
        (!squareFound)) {
      if (geometry::haveACommonVertex((*it1),(*it2),&square)) {
        dist1 = geometry::distanceBetweenPoints3D ((*it1).start,
            (*it1).end);
        dist2 = geometry::distanceBetweenPoints3D ((*it2).start,
            (*it2).end);
        ang = geometry::angleSquare (square);
        if(((dist1<MAX_TAM_SEG)&&(dist2<MAX_TAM_SEG))&&
        ((dist1>MIN_TAM_SEG)&&(dist2>MIN_TAM_SEG))&&
        ((ang > (DEGTORAD*80))&&(ang < (DEGTORAD*100)))) {
          // it is a parallelogram
          geometry::GetLastPointSquare (&square);
          squareFound = true;
          it3 = this->controller->parallelograms.begin ();
          coincidencia = false;
          while ((it3 != this->controller->parallelograms.end ())
              && (!coincidencia)) {
            igualQue = geometry::areTheSameParallelogram (square,
                (*it3), COMP_PARAL_THRES); // threshold to compare
            contieneOContenido = this->contieneOContenido (square,
                (*it3));
            coincidencia = (igualQue || contieneOContenido);
            if (!coincidencia) it3++; }
          square.life = MAX_LIFE;
          square.saliency = MIN_SALIENCY;
          square.timestamp = actualInstant;
          if (coincidencia) {
            int whichIsTheMax = geometry::getMaximizedParallelogram
                (square, (*it3));
            if (whichIsTheMax == 1) {
                this->controller->parallelograms.erase (it3);
                this->controller->parallelograms.push_back
                    (square); }
            this->controller->fittedParallelograms1_3D.push_back
                (square); } else {
            this->controller->parallelograms.push_back (square);
            this->controller->newParallelograms1_3D.push_back
                (square); } } }
      it2++; } } }
```

Code 3.1: Parallelograms abstraction function

```
inline void makePrediction (IplImage *img) {
  while (p != NULL) { // p is the list of segments in memory
      if (p->segment->isValid == 1) {
      // projection means to translate a point from 3D to 2D
      project(p->segment->start.position, &p1, robotCamera);
      project(p->segment->end.position, &p2, robotCamera);

        if(displayline(p1,p2,&gooda,&goodb,robotCamera)==1) {
          // it's projectable, inside the visual space
          // optical coordinates are translated to pixels
          // All of them are bad predicted at the beginning
          p->segment->isWellPredicted = FALSE;
          pt1.x=(int)gooda.y;
          pt1.y=SIFNTSC_ROWS-1-(int)gooda.x;

          pt2.x=(int)goodb.y;
          pt2.y=SIFNTSC_ROWS-1-(int)goodb.x;

          if (showPredictions == TRUE)
              // All of them are red in the beginning
              cvLine(img, pt1, pt2, CV_RGB(0,0,255), 3, 8, 0);

          my2y3DSeg = NULL;
          my2y3DSeg = (struct Segment2y3DList*)
              malloc(sizeof(struct Segment2y3DList));
          my2y3DSeg->segment2D.start.x = pt1.x;
          my2y3DSeg->segment2D.start.y = pt1.y;
          my2y3DSeg->segment2D.start.h = 1.;
          my2y3DSeg->segment2D.end.x = pt2.x;
          my2y3DSeg->segment2D.end.y = pt2.y;
          my2y3DSeg->segment2D.end.h = 1.;
          my2y3DSeg->segment3D = p->segment;
          my2y3DSeg->next = NULL;

          if (last2y3DSegment != NULL) {
              last2y3DSegment->next = my2y3DSeg;
              last2y3DSegment = last2y3DSegment->next;
          } else { // list = NULL, so it's initialized
              seg2y3Dlist = my2y3DSeg;
              last2y3DSegment = seg2y3Dlist;
      } } }
      p = p->next;
} }
```

Code 3.2: Motion prediction for objects in visual memory

```
/*Convert to Gray Image*/
cvCvtColor (&image, IplTmp1, CV_RGB2GRAY);

/*Normalize image*/
cvNormalize(IplTmp1, IplTmp1, 0, 255, CV_MINMAX);

// Make a average filtering
cvSmooth(IplTmp1,IplTmp2,CV_BLUR,3,3);

//Laplace
cvLaplace(IplTmp2, IplLaplace, 3);
cvConvertScale(IplLaplace,IplTmp1);

/*Perform a binary threshold*/
cvThreshold(IplTmp1,IplTmp2,ThressValue,255,CV_THRESH_BINARY);

/*Find contours*/
cvFindContours(IplTmp2, storage, &contour,sizeof(CvContour),
    CV_RETR_LIST, CV_CHAIN_APPROX_NONE);

/*Run through found coutours*/
while (contour != NULL) {
  /*Check length*/
  if (contour->total >= min_size_contour) {
      /*Convert to array*/
      WholePointArray = (CvPoint *)malloc(contour->total *
          sizeof(CvPoint));
      cvCvtSeqToArray(contour, WholePointArray, CV_WHOLE_SEQ);
// [...]
```

Code 3.3: Solis' algorithm steps

```
cvCvtColor(src, gray, CV_RGB2GRAY);
cvCanny(gray, edge, sliderThreshold, sliderThreshold*3, 3);
// canny filter is used to extract borders
lines = cvHoughLines2 (edge, storage, CV_HOUGH_PROBABILISTIC,
    distanceResolution, angleResolution, HOUGH_LINE_THRESHOLD,
    HOUGH_MIN_DIST_SEG, MAX_GAP_BETWEEN_SEGMENTS);
// probabilistic Hough transformation is more efficient when
    picture contains a few long linear segments
```

Code 3.4: Steps and parameteres used in Hough function

```
// [...]
if (areTheSameSegment (seg1b, seg2b)) {
   found = 1;
} else {
   // Check every segment in memory, with segment. Calculate
       intersection point between segment i and segment.start
       perpendicular
   ubi1 = distancePointLine (seg1b.start, seg2b, &proy1, &dist1);
   ubi2 = distancePointLine (seg1b.end, seg2b, &proy2, &dist2);

   found = overlapping (seg2b, proy1, proy2, ubi1, ubi2, dist1,
       dist2, 1);
  // If they're parallel lines we finish checking
}
// if "found" is True means that actual segment was well predicted
```

Code 3.5: Refutation of predicted segments in memory

```
inline int mergeSegment (Segment3D segment, int* areTheSame) {
   // p is the list of segments in memory
   while ((p != NULL) && (!found)) {
      if (p->segment->isValid == 1) {
         if (areTheSameSegment (*(p->segment), segment)) {
            found = 1;
            (*areTheSame) = 1;
         } else {
            // Check every segment in memory, with segment.
                Calculate intersection point between segment i and
                segment.start perpendicular
            ubi1 = distancePointLine (segment.start, *(p->segment),
                &proy1, &dist1);
            ubi2 = distancePointLine (segment.end, *(p->segment),
                &proy2, &dist2);

            found = overlapping (*(p->segment), proy1, proy2, ubi1,
                ubi2, dist1, dist2, 0);
          // If they're parallel lines we finish checking
          }
          if (found) { // refresh timestamp and travel time
            p->segment->timestamp = segment.timestamp;
            p->segment->traveled = segment.traveled;
          }
      }
      p = p->next;
   }
   return found;
}
```

Code 3.6: Process to avoid duplicated segments in memory

```
typedef struct {
  HPoint3D p1;
  HPoint3D p2;
  HPoint3D p3;
  HPoint3D p4;
  HPoint3D centroid;
  bool isValid;
} Parallelogram3D;

typedef struct {
  HPoint3D center;
  int isValid;
} Face3D;

typedef struct {
  HPoint3D start;
  HPoint3D end;
  colorRGB color;
  int isAttainable;
} Arrow3D;

struct elementStruct {
  double lastInstant;
  double firstInstant;
  float latitude;
  float longitude;
  int scenePos;
  float saliency;
  float liveliness;
  int type;
  // 0 = virtual element; 1 = rectangle; 2 = face; 3 = arrow
  int isVisited;
  Parallelogram3D parallelogram;
  Face3D face;
  Arrow3D arrow;
  struct elementStruct* next;
};
```

Code 3.7: Main visual memory elements

```
void Attention::updateElements () {
  std::vector<attentionElement>::iterator it;
  float maxSaliency = -9999.;

  for(it = this->elements.begin(); it != this->elements.end();
      it++) {
    (*it).liveliness -= LIFE_DECREMENT;
    (*it).saliency += SALIENCY_INCREMENT;

    if (((*it).saliency > maxSaliency)) { // attainable element
      maxSaliency = (*it).saliency;
      maxSaliencyElement = &(*it);
    }

    if ((*it).liveliness < LIFE_TO_DEAD) {
      this->elements.erase (it);
      if (depuracion::DEBUG) printf ("updateElements: Deleting old
          element of memory...\n");
    } else if ((*it).saliency > MAX_SALIENCY) {
      (*it).saliency = MAX_SALIENCY; // to avoid saturation
    }
  }

  if (maxSaliencyElement != NULL) {
    maxSaliencyElement->saliency = MIN_SALIENCY;
    maxSaliencyElement->liveliness = MAX_LIFE;
    maxSaliencyElement->lastInstant = actualInstant;
  }
}
```

Code 3.8: Update visual memory elements function

```
cvCvtColor (&img, imgLocal, CV_BGR2GRAY);
cvClearMemStorage (storage);
faces = cvHaarDetectObjects (imgLocal, cascade, storage,
    SCALE_FACTOR, MIN_NEIGHBORS, OPERATION_MODE, cvSize
    (MIN_WINDOW_WIDTH, MIN_WINDOW_HEIGHT));
```

Code 3.9: Haar Cascade face detection OpenCV function

```
inline void hypothesizeArrows () {
  while (p != NULL) { // p is the list of segments in memory
    if (p->segment->isValid == 1) {
      dist1 = distanceBetweenPoints (p->segment->start.position,
          p->segment->end.position);
      if ((dist1<MAX_TAM_ARROW) && (dist1>MIN_TAM_ARROW)) {
      q = p->next;
      found = FALSE;
      while ((q != NULL) && (!found)) {
          if (q->segment->isValid == 1) {
              dist2 = distanceBetweenPoints
                  (q->segment->start.position,
                  q->segment->end.position);
              if ((dist2<MAX_TAM_ARROW) && (dist2>MIN_TAM_ARROW)) {
                if (((dist1<MAX_TAM_ARROW) && (dist1>MIN_TAM_BASE)
                    && (dist2>MIN_TAM_ARROW)) ||
                    ((dist2<MAX_TAM_ARROW) && (dist2>MIN_TAM_BASE)
                    && (dist1>MIN_TAM_ARROW))) {
                  if (haveACommonVertex(*(p->segment),
                    *(q->segment),&square)) {
                    ang = angleBetweenSegments(square);
                    if(((ang > (DEGTORAD*25))&&(ang <
                        (DEGTORAD*65))) || ((ang >
                        (DEGTORAD*115))&&(ang < (DEGTORAD*155)))) {
                      found = TRUE;
                      // square.p1 is the intersection point between
                          base and arrow. It gives the direction
                      buildSemiArrow (*(p->segment), *(q->segment),
                          square.p1.position, dist1, dist2,
                          &mySemiArrow);
                      buildArrow (mySemiArrow.base, &myArrow);
                      insertArrowOnAttentionSystem (myArrow);
                      if (DEBUG) printf ("Arrow Found!\n");
                      if ((!checkedArrow) && (myMaxSaliency != NULL)
                          && (myMaxSaliency->type == 3) &&
                          (areTheSameArrow
                          (myMaxSaliency->arrow,myArrow))) {
                      // review arrow and it is there
                        checkedArrow = TRUE;
            } } } } } }
          q = q->next;
    } } }
    p = p->next;
} }
```

Code 3.10: Arrows abstraction function

```
inline void initSaliencyParams () {
  myElements = NULL;
  myMaxSaliency = NULL;
  myPrevMaxSaliency = NULL;
  myActualElement = NULL;
  seg2y3Dlist = NULL;
  segPointList = NULL;
  elementsStructCounter = 0;
  completedSearch = FALSE;
  analizedImage = FALSE;
  checkedParallelogram = FALSE;
  checkedFace = FALSE;
  checkedArrow = FALSE;
  completedTrack = FALSE;
  myActualState = think;
  timeToForcedSearch = TIME_TO_FORCED_SEARCH;
  isForcedSearch = FALSE;
  randomPosition = FALSE;
  nextLatitude = 0;
  nextLongitude = 0;
  completedMovement = FALSE;
  actualInstant = ((double) cvGetTickCount() /
      ((double)cvGetTickFrequency()))/1000000;
  timeForced = actualInstant;
  stopInstant = actualInstant;
  timeToUpdateCache = actualInstant;
  timeToMaintenance = actualInstant;
  timeToSaveImage = actualInstant;
  comeFromCenter = TRUE;
  last_full_movement = right;
  navigationTime = 0.;
  imageNumber = 0;
}
```

Code 3.11: Visual attention system objects initial attributes

```
void Attention::point2angle (HPoint3D p, HPoint3D cameraPosition,
    double *longitude, double *latitude) {
  float pan, tilt;

  p.X = p.X - cameraPosition.X;
  p.Y = p.Y - cameraPosition.Y;
  p.Z = 0. - cameraPosition.Z;
  p.H = 1.;

  if (p.X == 0.) pan = 0.;
  else pan = atan(p.Y/p.X);

  if (p.X == 0.) tilt = 0.;
  else // Z is always < 0 because is upper than point
    tilt = atan (p.Z/p.X);

  *longitude = pan * 180.0/M_PI;
  *latitude = tilt * 180.0/M_PI;
}
```

Code 3.12: Translating from Cartesian coordinates to Polar coordinates

```
void SolisDetector::process_horizon(cv::Mat &src) {
   /*Calculate horizon line using kinematics*/
   this->getHorizonLine(A, B, C);

   for(col=marginx;col<ImageInput::IMG_WIDTH;
      col=col+ImageInput::IMG_STEP) {
      /*Starts from the horizon line*/
      row = this->getHorizonPos(A, B, C, col);
      while(row <= ImageInput::IMG_HEIGHT-1-marginy) {
         rowend = ImageInput::IMG_HEIGHT-1-marginy;
         ccolor = this->imagehandler->getColor(col, row);
         if(ccolor != ImageInput::CGREEN) {
            row+=step;
            continue;
         }
         /*Get the size of the ground color*/
         size = this->imagehandler->getSegmentSizeDown(col, row,
             rowend, ccolor);
         if(size >= num_green_pixels)
            break;
         row+=size;
      }
      borders[counter].x = (float) col;
      borders[counter].y = (float) row;
      borders[counter].h = 1.0;
      counter++;
   }
   /*Calculate convex hull*/
   this->calcConvexHull();
}
```

Code 3.13: Solis' extra step to preprocess horizon

```
double geometry::distanceBetweenPoints2D(int x1, int y1, int x2,
    int y2) {
   return sqrt(G_SQUARE(x2-x1) + G_SQUARE(y2-y1));
}
```

Code 3.14: Getting Euclidean distance between segments extrems

Chapter 4

Educational framework

This chapter presents the teaching environment (`JdeRobot-Kids`) that has been developed and that is aimed at improving the teaching of Robotics to secondary students. It is based on the Python programming language and on different robotic platforms. It consists of a software infrastructure and a set of practical exercises that complete an academic course. The software infrastructure offers support for both real and simulated robots, one of which (`PiBot`) has been specifically developed for this teaching environment. In addition, the constructivist methodology that has been followed to improve the current technological education in Robotics is also described. The teaching proposal has been validated experimentally with more than 2,000 real students during the last two years. All the experiments shown here and many more are publicly available[a].

[a] `http://jderobot.org/JulioVega_PhD`

The developed teaching environment[1] includes a hardware platform (Section 4.1), a software infrastructure (Section 4.2), as well as an educational program (Section 4.3) for a full academic year, and a suggested specific pedagogical methodology (Section 4.4). All this was put into practice and was appropriately evaluated (Section 4.5).

[1] `http://jderobot.org/JdeRobot-kids`

The central pillars of the design of this environment are: robots with free hardware processors (Arduino and Raspberry Pi), the Python programming language and a collection of practice activities of progressive complexity.

4.1 Hardware platform

An Arduino-based robot, the MakeBlock `mBot`[2] (Figure 4.2 left) was chosen as the main reference hardware platform. Another more advanced and powerful robot, based on Raspberry Pi, was developed specifically for this framework. It was named `PiBot`.

4.1.1 Robot based on Arduino

The `mBot`, with its Arduino Uno processor, can be connected to the sensors and motors that are commonly used in educational robotics. It has different models depending on its connectivity: USB, 2.4G, Bluetooth. It can be connected through the USB cable to the computer to download programs. It is affordable, mechanically very compact and extensible. Kits of mechanical parts, such as sensors or actuators, can be bought also at low cost.

In addition, it has good support for programming in the mBlock graphic language (Figure 4.1), which is based on Scratch 2.0, and in the Arduino language, which has an extensive community of users all over the world and proven software tools.

In addition to the real robot, the counterpart for the Gazebo simulator (Figure 4.2 right) was also programmed in `JdeRobot-Kids`. Gazebo is a free software 3D simulator that incorporates several physical engines for realistic simulations and is a *de facto* standard in the robotics research community, with more powerful robot models ([Cañas et al., 2014]). Specifically, the graphic, mechanical model and a C ++ plugin that runs within the simulator and that is able to communicate with external programs were developed for `mBot`. This plugin allows the students' programs to collect readings from the virtual IR and ultrasound sensors, as well as to

[2] `https://makeblock.es`

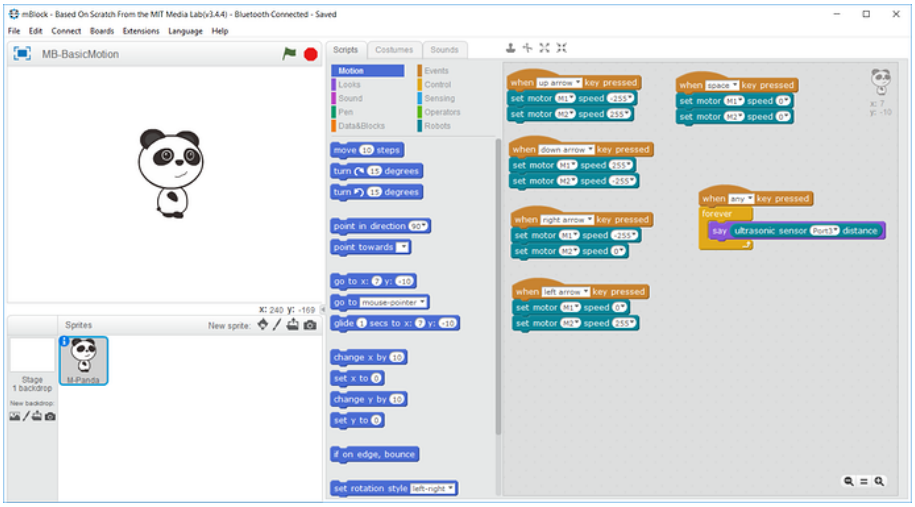


Figure 4.1: Program in mBlock graphic language

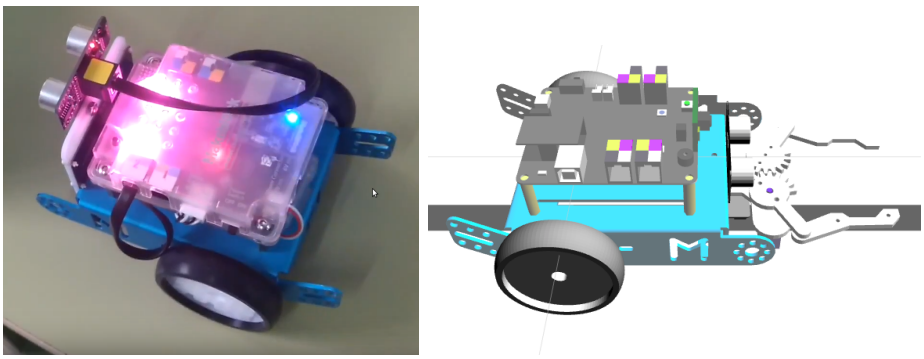


Figure 4.2: Robot mBot real and simulated in Gazebo

send movement commands to the emulated motors. That is, it allows the behavior of the robot in the simulated world to be controlled.

The initial motivation of giving support to the simulated robot is that students and educational centers that do not have the physical robot can nevertheless practice and learn or teach robotics with `JdeRobot-Kids` ([Vega et al., 2018]). In addition, thanks to this support, the problem that always arises when introducing robotic artifacts in a classroom is mitigated: economic costs and hardware maintenance.

Likewise, a homemade robot was also built connecting sensors and actuators to a protoboard mounted on an Arduino and assembling them in

a mechanical chassis. This shows the versatility of the teaching framework, which is valid for different platforms as long as they incorporate the Arduino microprocessor.

4.1.2 Robot based on Raspberry Pi

As already mentioned in Section 1.4, a more powerful platform is needed and, above all, one that permits the use of a camera to implement the vision algorithms included in the academic program of this educational framework, and whose motivation has already been described in Section 1.1.3.

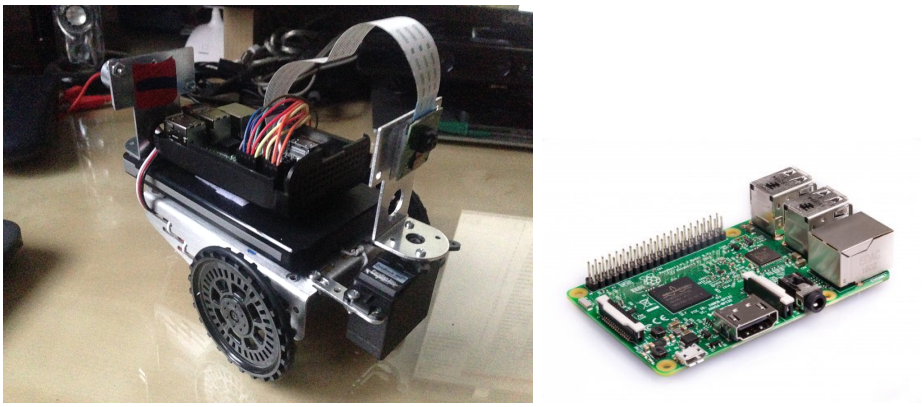


Figure 4.3: Robotic platform PiBot (left) based on Raspberry Pi 3 (right)

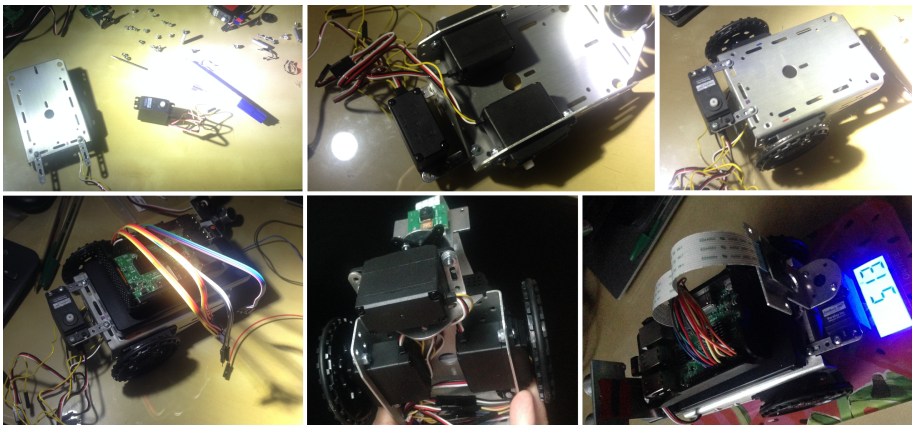


Figure 4.4: Assemblage of components of the robotic prototype, PiBot

Thus, a robotic platform was developed, `PiBot` (Figure 4.3 left), whose computational core is the Raspberry Pi 3 controller board (Figure 4.3 right). This was mounted on a chassis, to which were added a battery of 20,000 mAh and two servos model Feedback 360° from the Parallax company (Figure 4.4), in addition to a caster. The main sensors mounted were an ultrasound sensor model HC-SR04 (Figure 4.5 left) and the Raspberry PiCamera camera (Figure 4.5 right) mounted on another servo, giving it a degree of freedom of movement. The main technical details are included in Table 4.1. Other sensors that are also supported are the bumper and the infrared sensors.

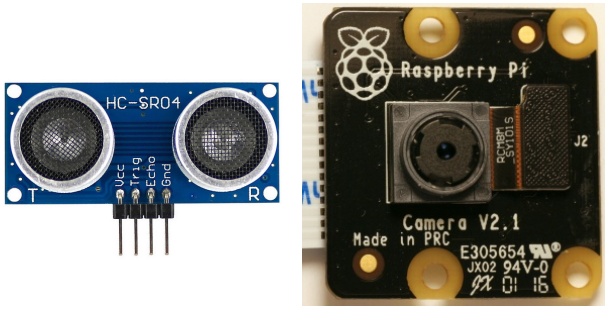


Figure 4.5: Ultrasonic sensors model HC-SR04 and Raspberry PiCamera camera

As with the `mBot` robot previously described, for `PiBot` the counterpart for the Gazebo simulator was also programmed in `JdeRobot-Kids` (Figure 4.6).

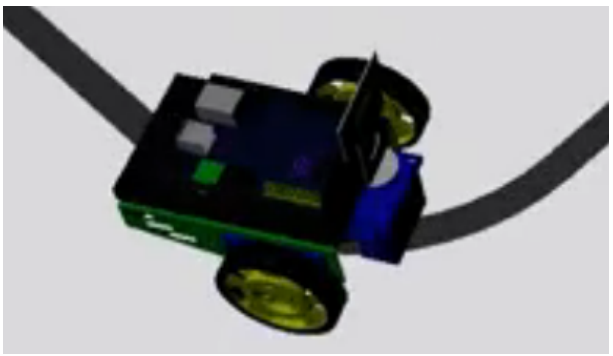


Figure 4.6: PiBot robot simulated in Gazebo

PiCamera parameters	Values
Sensor type	Sony IMX219PQ[7] CMOS 8-Mpx
Sensor size	3.674 x 2.760 mm (1/4" format)
Pixel Count	3280 x 2464 (active pixels)
Pixel Size	1.12 x 1.12 um
Lens	f=3.04 mm, f/2.0
Angle of View	62.2 x 48.8 degrees
SLR lens equivalent	29 mm

Table 4.1: PiCamera (v2.1 board) technical intrinsic parameters

Several factors led to the development of this hardware platform based on Raspberry Pi. First, the power and versatility offered by a controller that has a complete and functional operating system based on Linux; specifically, the Raspbian Stretch distribution[3], Desktop version with graphical interface (GUI) was used. Secondly, it provided the possibility of controlling a powerful camera, `PiCamera`, connected to the board by its own dedicated data bus. And finally, the freedom offered by the board to connect and control varied devices thanks to its numerous ports `GPIO`[4] (*General Purpose Input/Output*), configurable to serve as input and output of data ([Balachandran, 2009]).

4.2 Language and software infrastructure

Arduino is normally programmed by Arduino IDE or by Scratch (or some of its variants such as mBlock of `mBot`). In `JdeRobot-Kids`, Python was chosen as a programming language because of its simplicity, its expressive power and because it is widely used in higher levels of education and programming. It is a text language, interpreted and object oriented. This language is easier to learn than the Arduino language (very similar to C/C++) and at the same time it has great power. It is also used in university education, together with more powerful libraries.

Two questions arise with this approach: (a) the difficulty of learning to program in a programming language that is not visual; and (b) the

[3] `https://www.raspberrypi.org/downloads/raspbian`
[4] `https://www.raspberrypi.org/documentation/usage/gpio`

high economic and logistical cost of acquiring a considerable amount of robotic equipment for a high school class, which usually has around thirty students.

As the Python language is not supported by the manufacturer of the `mBot`, an entire infrastructure was created in `JdeRobot-Kids` ([Vega and Cañas, 2016]). The Arduino microprocessor is too limited to run an on-board Python interpreter, so other options had to be explored. The chosen design is shown in Figure 4.7 and gives priority to simplicity of use, which required making the underlying infrastructure quite sophisticated.

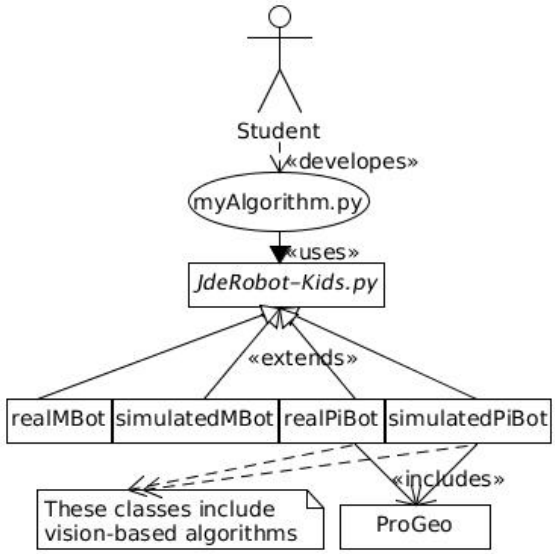


Figure 4.7: The student uses the JdeRobot-Kids.py library in his program

An specific library was developed that provides the programming interface (API), `JdeRobot-Kids.py`[5]. This simple and natural interface includes methods to read the measurements from the sensors and methods to give commands to the actuators of both the `mBot` and the `PiBot` (Table 4.2).

The use is as simple as programming an application in Python to use these methods to control the robot. In this way, students concentrate on

[5] `https://github.com/JdeRobot/JdeRobot/blob/master/src/interfaces/python/JdeRobotKids.py`

the algorithm they are developing, avoiding the low level details such as ports, connectivity with the robot, etc. which are stored in the library configuration file.

Acting	**Sensory**
mover (V, W)	leerIntensidadLuz
avanzar (V)	leerUltrasonido
retroceder (V)	leerIntensidadSonido
girarIzquierda (W)	leerPotenciometro
girarDerecha (W)	leerMandoIR
parar	dameSonarVisual
moverServo (θ)	dameImagen
encenderLed	dameObjetoDeColor
apagarLed	
escribirTexto (T)	
playTono (Fs)	

Table 4.2: JdeRobot-Kids.py programming interface

The most important API methods for `JdeRobot-Kids.py` are detailed in Table 4.2. They allow access to each of the usual sensors, for example US sensor, IR sensors or light sensors, and also to the camera (in the case of `PiBot`). With respect to the motors, each of them can be governed individually (raw methods). Movement orders for the whole robot (cooked methods) can also be sent, which are simpler to use. In this case, it is the library which translates the desired combined movement into the orders for each of the two motors that carry it out.

The students program their exercises in Python by writing the file `myAlgorithm.py`, for example, with a text editor. From this program all the methods provided by this library may be used. `JdeRobot-Kids.py` includes below two different modules that exactly perform the same API functions. One module implements the interface for the management of the real robot and another for the simulated robot in Gazebo, for both `mBot` and `PiBot`. The final robot in each case is selected by specifying it on the library configuration file. As the programming interface is the same in both cases, the application is identical and works interchangeably on the physical platform and on the simulated one.

4.2.1 Modules for mBot

The module for the real robot is called `mBotReal` and was programmed as a Python library that runs on the computer and communicates continuously with the physical robot `mBot` via the Firmata protocol[6], in which an intermediary program is executed on the native Arduino firmware (Figure 4.8).

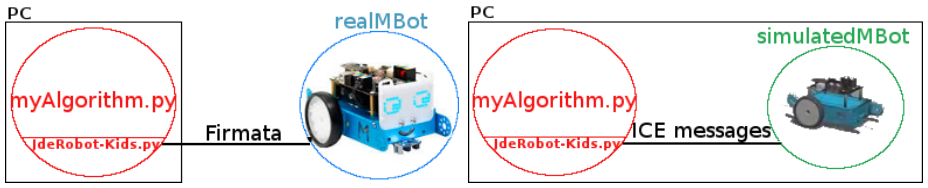


Figure 4.8: Connection of the JdeRobot-Kids.py library with the real mBot and the simulated mBot

The orders issued from the student's application arrive at the library, which transmits them (via USB or via Bluetooth 2.4G) —following the Firmata protocol— to the intermediary program on board the robot, written in Arduino language, which executes them on the motors. The read requests of sensors from the application arrive at the library, which takes the last readings received from the intermediary program on board and delivers them to the application.

The second module developed, `mBotGazebo`, allows access to the simulated robot inside *Gazebo* (Figures 4.7 and 4.8). In this case the methods of the API `JdeRobot-Kids.py` are translated to send messages to the Gazebo developed plugin, in C ++, which controls the sensors and actuators emulated in Gazebo. These messages were implemented with the ICE communications middleware. The simulator works on Linux computers natively and on MS-Windows or MacOS computers using *docker* containers.

4.2.2 Modules for PiBot

To support `PiBot`, the developed robotic prototype, other homonymous modules were programmed that include all the necessary functions to send commands to the servos and read values from the most common sensors,

[6] `https://github.com/firmata/protocol`

such as ultrasound or infrared, for both the real robot (Figure 4.9 left) and the simulated one (Figure 4.9 right).

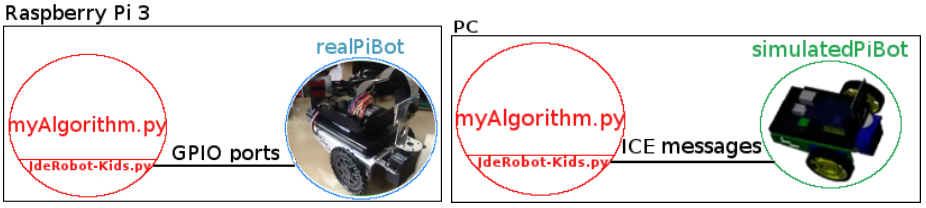


Figure 4.9: Connection of the JdeRobot-Kids.py library with the real PiBot and with the simulated PiBot

However, the greatest novelty of this prototype, as already discussed in Section 4.1.2, is the support developed for the powerful camera mounted on board, the PiCamera. A complete software library was implemented so that the student may use this camera, or even a WebCam, as the main sensor of this robotic platform.

The main use of this camera as a sensor is for navigation tasks in real environments with obstacles. As already seen in Section 2.3, the state of the art in the use of vision for navigation shows an extended use of a stereo pair of cameras. However, based on the *ground hypothesis* already successfully developed, and shown in Chapter 3, a single camera can be used to estimate distances to objects and, furthermore, using it, the robot can perform navigation by making intelligent decisions.

4.2.3 Modules for PiCamera

A library was developed to support the Raspberry PiCamera camera. It contains an abstract model of the camera and implementations of several projective geometry algorithms. An image server of any camera was also implemented with the communications middle-ware *ICE*, `piCamServer`.

4.2.3.1 Pin-hole camera model

Firstly, the camera model for the PiCamera was implemented (Code 4.1). The camera model, assuming an ideal camera, is called Pin-hole model. When an image is taken using a pin-hole camera, we lose some

important information, the depth of the image; i.e. how far each point in the image is from the camera, because it is a 3D-to-2D conversion. Thus, an important question is how to find the depth information using cameras. One answer is to use more than one camera. Our eyes work in a similar way to the use of two cameras (two eyes). This is called stereo vision. Figure 4.10 shows a basic setup with two cameras taking an image from the same scene.

```
class PinholeCamera:
    position = Punto3D() # camera 3d position in mm
    foa = Punto3D() # camera 3d focus of attention in mm
    roll = None # camera roll position angle in rads
    fdistx = None # focus x distance in mm
    fdisty = None # focus y distance in mm
    u0 = None # pixels
    v0 = None
    skew = None # angle between the x and y pixel axes in rads
    rows = None # image height in pixels
    columns = None # image width in pixels
    K = numpy.array ([(0,0,0,0),(0,0,0,0),(0,0,0,0)]) # camera
        intrinsic parameters
    RT = numpy.array ([(0,0,0,0),(0,0,0,0),(0,0,0,0),(0,0,0,0)]) #
        camera rotation and translation matrix
```

Code 4.1: PiCamera class

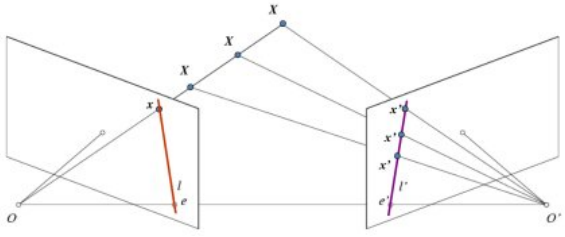


Figure 4.10: Cameras taking an image from the same scene

4.2.3.2 Epipolar Geometry and ground hypothesis

As `PiBot` uses a single camera, the system can not find the 3D point corresponding to the pixel x in an image because every point on the line

OX projects to the same pixel on the image plane. Two cameras would get two images and it would be then possible to triangulate the right 3D point. In this case, to find the matching point in another image, it is not necessary to search the whole image, but only along the epiline. This is called Epipolar Constraint. Similarly, all points will have their corresponding epilines in the other image. The plane XOO' is called Epipolar Plane (see Figure 4.10). The intersections between the plane XOO' and the image planes form the *epipolar* lines.

To convey a depth map from the robot to the surrounding objects using a single camera, the *ground hypothesis* is assumed. It considers all the objects are on the floor, on the ground plane, which is a known location (on plane $Z = 0$). In this way, the 3D point corresponding to every point in the single image can be obtained (Figure 4.11).

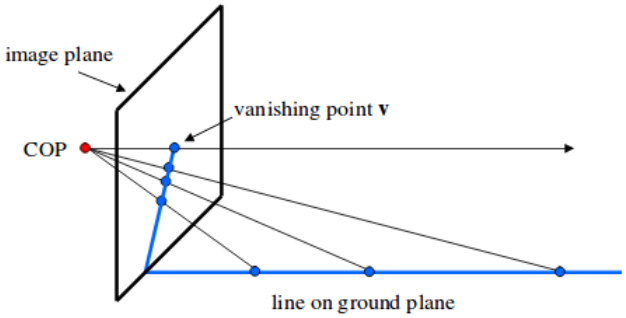


Figure 4.11: Ground Hypothesis assumes all objects are on the floor

Let us consider the pixels corresponding to the filtered border (Figure 4.12 top right) below the obstacles (Figure 4.12 top left), so that the distances can be calculated later ((Figure 4.12 down). The code corresponding to solve this feature is shown in Code 4.2.

4.2.3.3 Coordinate systems transformations

To compute the rays back-projecting from the pixels and the rays projecting from 3D points into pixels, the position of the camera needs to be calculated. A camera is defined and positioned according to its matrices $K * R * T$ (Table 4.3).

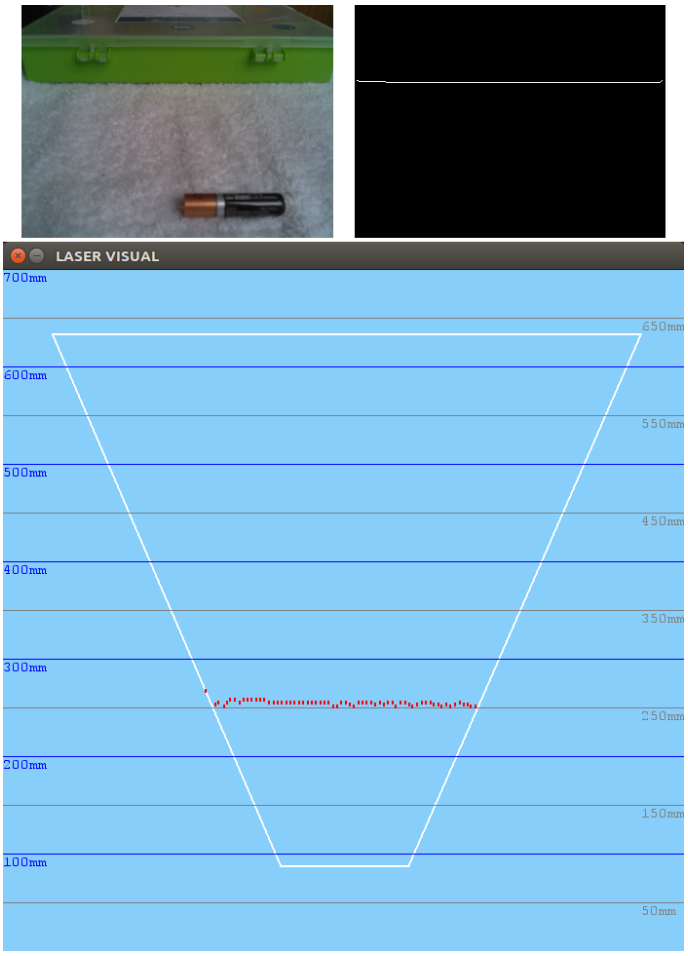


Figure 4.12: Ground Hypothesis example

Camera parameters	Definitions
K (3x3)	intrinsic parameters
R (3x3)	camera rotation
T (3x1)	camera translation
X	forward shift
Y	left shift
Z	upward shift

Table 4.3: Definition of pin-hole camera position parameters and orientation

Instead of matrices R and T, a single matrix $RT(4x4)$ can be used,

which includes RT, so would therefore be $4x4$, as shown in Equations 4.1, 4.2, 4.3, corresponding to X, Y and Z rotation axes.

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & cos(\theta) & sin(\theta) & Y \\ 0 & -sin(\theta) & cos(\theta) & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

Equation 4.1: RT camera matrix rotating in X-axis

$$\begin{bmatrix} cos(\theta) & 0 & -sin(\theta) & X \\ 0 & 1 & 0 & Y \\ sin(\theta) & 0 & cos(\theta) & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Equation 4.2: RT camera matrix rotating in Y-axis

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & X \\ sin(\theta) & cos(\theta) & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Equation 4.3: RT camera matrix rotating in Z-axis

These three angles are used to build their three corresponding matrices, which are subsequently multiplied.

Let us consider an example using R and T matrices separately. Given a point in ground plane coordinates $P_g = [X, Y, Z]$, its coordinates in camera frame (P_c) are given by Equation 4.4.

The camera center is $C_c = [0, 0, 0]$ in camera coordinates. Its ground coordinates are computed in Equation 4.5, where R' is the transpose of R and assuming, for simplicity, that the ground plane is $Z = 0$.

Let K be the matrix of intrinsic parameters, whose PiCamera values are obtained using the developed `PiCamCalibrator`[7] tool (Code 4.3). Given a pixel $q = [u, v]$, it can be written in homogeneous image coordinates as

[7] `https://github.com/JdeRobot/JdeRobot/tree/master/src/tools/piCamCalibrator`

$$P_c = R * P_g + T \tag{4.4}$$

Equation 4.4: Camera frame coordinates

$$C_g = -R' * T \tag{4.5}$$

Equation 4.5: Camera center in ground coordinates

$Q = [u, v, 1]$. Its location in camera coordinates (Q_c) is shown in Equation 4.6, where $K_i = inv(K)$ is the inverse of the intrinsic parameters matrix. The same point in world coordinates (Q_g) is given by Equation 4.7.

$$Q_c = K_i * Q \tag{4.6}$$

Equation 4.6: Pixel in camera coordinates

All the points $Pg = [X, Y, Z]$ that belong to the ray going from the camera center through that pixel, expressed in ground coordinates, are then on the *epipolar* line given by Equation 4.8, where the θ value goes from 0 to positive infinity.

Due to the *ground hypothesis* being assumed (see Section 4.2.3.2), this *epipolar* line is intersected with ground plane (Code 4.4), where objects are supposed to lie.

4.2.3.4 PiBot camera rotation and translation

Every time the camera is moved with respect to several axes (as shown in Figure 4.13), camera matrices (Table 4.3) must be multiplied again and again. Thus, the following steps are needed for a complete translation of the camera:

(M_1) Considering robot encoders with information of X, Y and Θ, the robot is moved with respect to the absolute axis $(0, 0)$ of the world, and rotated with respect to the Z axis, so the RT matrix of the robot would be as shown in Equation 4.3.

(M_2) Assuming the camera is mounted over a Pan unit (servo), it will be

$$Q_g = R' * Q_c - R' * t \tag{4.7}$$

Equation 4.7: Pixel translated to the 3D world coordinates

$$P_g = C_g + \theta * (Q_g - C_g) \tag{4.8}$$

Equation 4.8: Epipolar line in ground coordinates corresponding to the pixel

moved along the Z axis with respect to the base of the robot (which is on the ground level).

(M_3) Furthermore, the Pan axis is rotated with respect to the Z axis according to the Pan angle (Equation 4.3).

(M_4) The Pan support is also rotated with respect to the Y axis according to the Tilt angle (Equation 4.2), needed to perceive close objects.

(M_5) Finally, the optical center of the camera is translated in X and in Z with respect to the Tilt axis.

Thus, to obtain the absolute position of the camera in world coordinates, the five different matrices previously described are multiplied following Equation 4.9, and coded in Code 4.5.

$$\begin{aligned} M_a &= M_1 * M_2 \\ M_b &= M_a * M_3 \\ M_c &= M_b * M_4 \\ M_d &= M_c * M_5 \end{aligned} \tag{4.9}$$

Equation 4.9: Obtaining camera position in world coordinates

The absolute position $[X, Y, Z]$ of the camera is given by M_d, in cells $[[0, 3], [1, 3], [2, 3]]$. The camera position and orientation can be expressed using M_d and Focus of Attention (FOA). In this case, a column corresponding to the relative FOA $[X, Y, Z]$ is multiplied by the M_d matrix resulting in an absolute FOA given by Equation 4.10.

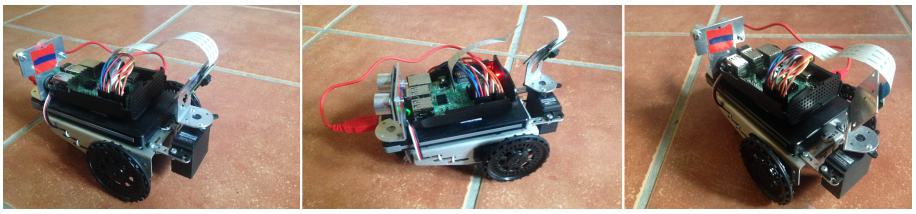


Figure 4.13: Robot and camera are continously moving with respect to several axes

$$FOA_{absolute} = M_d * FOA_{relative} \tag{4.10}$$

Equation 4.10: Obtaining absolute camera position and orientation

4.3 Academic program

A plan of activities for the subject *Programming, Robotics and Technology* was designed and implemented in different years of Secondary Education and for a course of extracurricular activities. Since the students in these year groups have no notion of computer programming, they have to start from a very basic level until they ultimately achieve the development of a complex project consisting of a classic robotics task.

The academic program was divided into four phases of progressive learning:

1. 14 sessions: Basic notions of programming using the visual language Scratch: loops, conditions, variables, etc.

2. 10 sessions: Introduction to Python language, with basic practice activities using loops, conditions, variables, functions, etc.

3. 20 sessions: Robotics practice programming with sensors and actuators individually.

4. 10 sessions: Programming behaviors in a robot. Final project that encompasses all the above.

Each phase is described below, indicating what it includes and what practical tasks students develop as the academic program progresses.

4.3.1 Basics of programming

In the first part of the course, basic notions of computer programming are acquired. In this way, students understand the way a computer works internally; and, therefore, they understand the reason for the use of variables or functions. Furthermore, concepts such as loops or conditionals are totally new to them. Hence, this first contact with the subject is so important. Depending on the students, this usually lasts about four sessions.

After that, another five sessions are dedicated to materializing the basic notions learned in a language close to them, an intuitive language, such as the visual language Scratch, already shown in Section 2.5.2. Here some aspects of syntax are presented in broad strokes, as well as novel concepts that continue to appear, such as that of a counter, the use of sets or vectors. They also understand why a variable must be defined, and other minor topics. It is a very important phase where students internalize the structure, organization and restrictions of a programming language.

The practical exercises that the students carry out in Scratch to achieve the aforementioned objectives, which usually take about ten sessions, are the following:

1. *Introduction to Scratch.* Designing an interactive character so that, when clicked on, there will be some visual effect, a movement, a sound, and a change of appearance.

2. *Use of variables.* Developing a game in which the previously designed character picks up objects distributed around the scene.

3. *Dynamic objects (loops).* Adding to the game objects that move constantly in some cyclic movement. In case they touch the character, it will lose a life; if it reaches 0, the game ends.

4. *Final project.* Continuing the game with different screens (phases) through which the character progresses and which will be accessed through passages, pipes and secret access.

4.3.2 Introduction to the Python language

In this second phase the basic notions of the Python language are specified. It focuses on knowing keywords of the language, as well as purely syntactic issues that are typical of Python. Here the students are already prepared to perform some classic exercises of initiation to programming such as:

1. Printing the numbers from 1 to 100.
2. Printing the numbers from 100 to 0.
3. Printing the even numbers between 0 and 100.
4. Printing the sum of the first 100 numbers.
5. Printing the odd numbers to 100 and how many there are.
6. Printing the natural numbers from 1 to another entered by keyboard.

The realization, correction and explanation of these exercises take ten sessions.

4.3.3 Robotic practice activities: handling of sensors and actuators

In this third phase the students carry out a total of ten activities directly related to Robotics. They begin assembling different components on an Arduino board (in the case of the homemade prototype by pieces) and review some basic concepts of electronics so that they have no problems when connecting the different devices. Step by step, they begin by installing simple components on a protoboard mounted on Arduino, such as a buzzer, or LEDs, and their corresponding software developments, to move to more complex ones such as light, infrared or ultrasound sensors. Once they have mastered the electronics of these components, they master the use of the sensors and actuators already pre-installed in a `mBot`, as well as the `PiBot` servos. Finally, they begin with the use of the camera as a sensor and the treatment of images that it provides. All this covers about twenty sessions, as follows:

1. Arduino. Sound reproduction by buzzer (Figure 4.14 left).

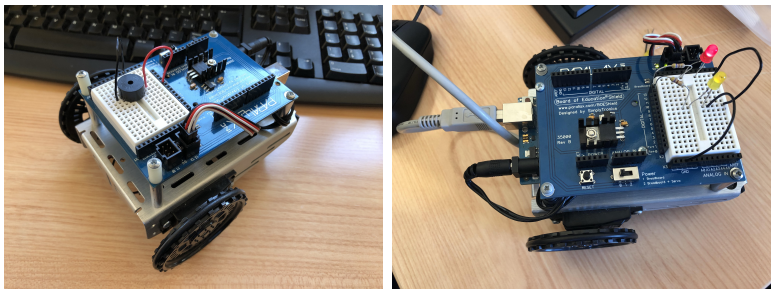


Figure 4.14: Practice tasks with Arduino to handle buzzer and leds

2. Arduino. Use of push button with LED (Figure 4.15 left).

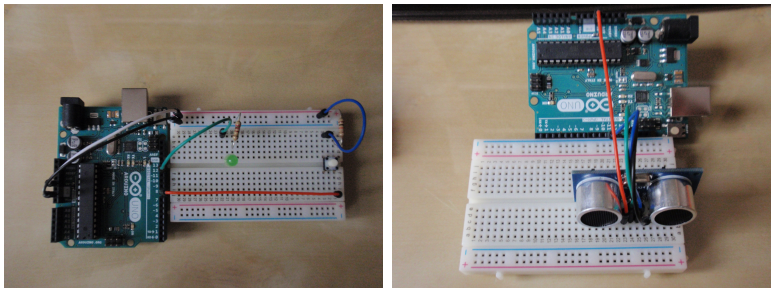


Figure 4.15: Practice task with Arduino to operate a push button with LED and ultrasonic

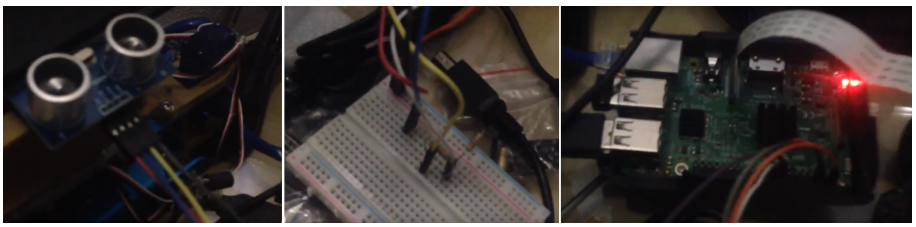


Figure 4.16: Practice activity with PiBot to handle an ultrasonic sensor

3. Arduino. Ultrasonic sensor reading (Figure 4.15 right y Figure 4.16).

4. mBot. Use of push-button with LEDs (Figure 4.17 left) and microphone (Figure 4.17 right).

5. mBot. Using LED array (Figure 4.18).

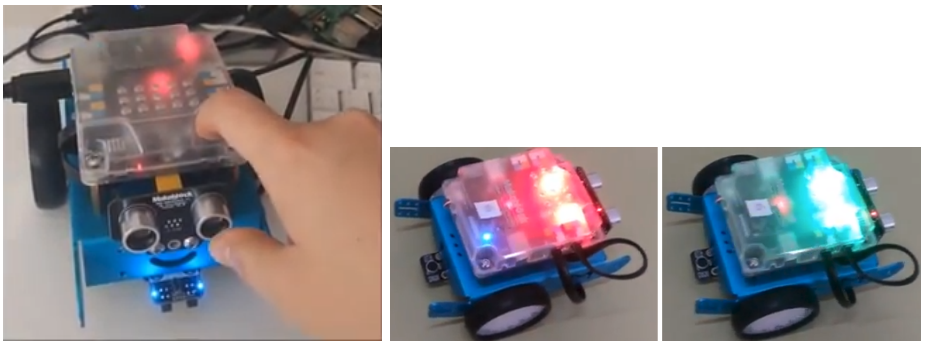


Figure 4.17: Practice tasks with mBot to operate push-button and LED, and microphone with LED to recognize sounds

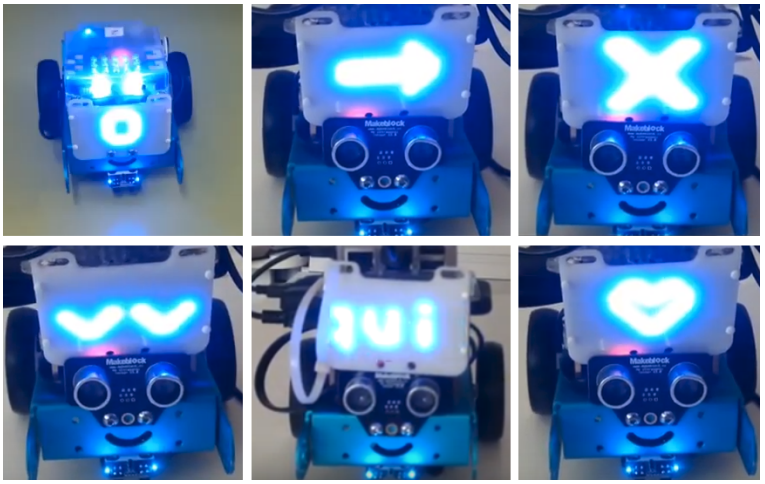


Figure 4.18: Practice task with the LED matrix actuator

6. mBot and PiBot. Control of servos (Figure 4.19).
7. mBot. Gripper control on pan-tilt (Figure 4.20).
8. Reading from file and static image sample.
9. PiBot. Reading and displaying images from WebCam and PiCamera (Figure 4.21).
10. PiBot. Visual sonar reading from PiCamera (Figure 4.22).

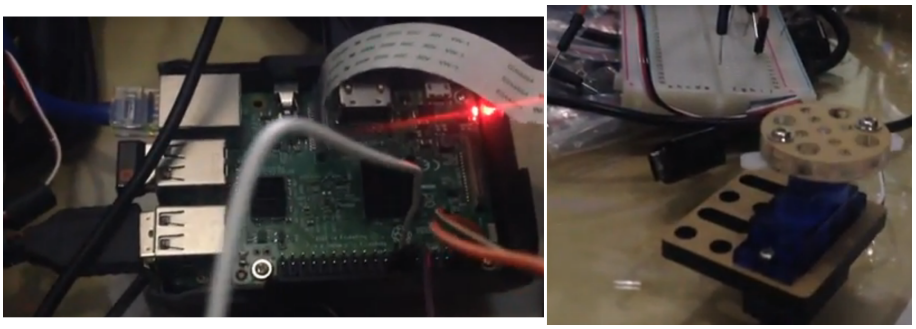


Figure 4.19: Practice task with PiBot to operate a servo

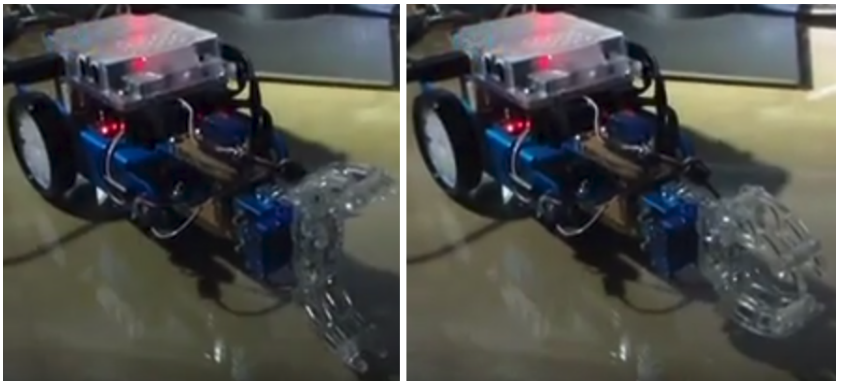


Figure 4.20: Practice with the mBot claw actuator

4.3.4 Robotic practice activities: autonomous behaviors

The last step of this learning pyramid consists of a complete robotics project where students combine everything previously learnt. For example, some projects developed are:

1. Navigation following a line (Figures 4.23, 4.24 and 4.25).
2. Navigation avoiding obstacles by means of ultrasounds (Figures 4.26 and 4.27). Its implementation in JdeRobot-Kids is shown for any platform in Code 4.6.
3. Navigation following the light projected by the flash of a mobile phone.
4. Sumo fight between two mBots.

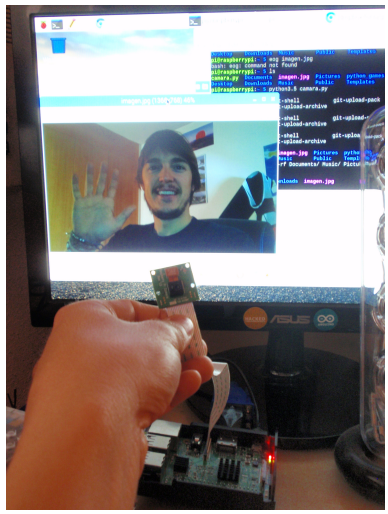


Figure 4.21: Reading and displaying images from WebCam and PiCamera

5. Navigation by clapping.
6. Stone, paper or scissors game, using the LED array.
7. Following an object using a camera sensor (Figure 4.28).
8. Navigation avoiding obstacles using a camera sensor (Figure 4.29).

It takes another ten sessions to finish this final project.

4.4 Constructivist methodology in robotics

My own experience teaching robotics to pre-university students for several years, the analysis of teaching methodologies carried out in Section 2.5.4, and especially the on-site study of how to implement this teaching in Finland, within the European Erasmus+ project mentioned in Preamble, have served to refine the proposed teaching methodology.

It is recommended to use the `JdeRobot-Kids` teaching framework within the constructivist methodology described in this section. This is based on the premise that knowledge is in the participants, and that these —which could be called *thinking subjects*— have no alternative but to build their

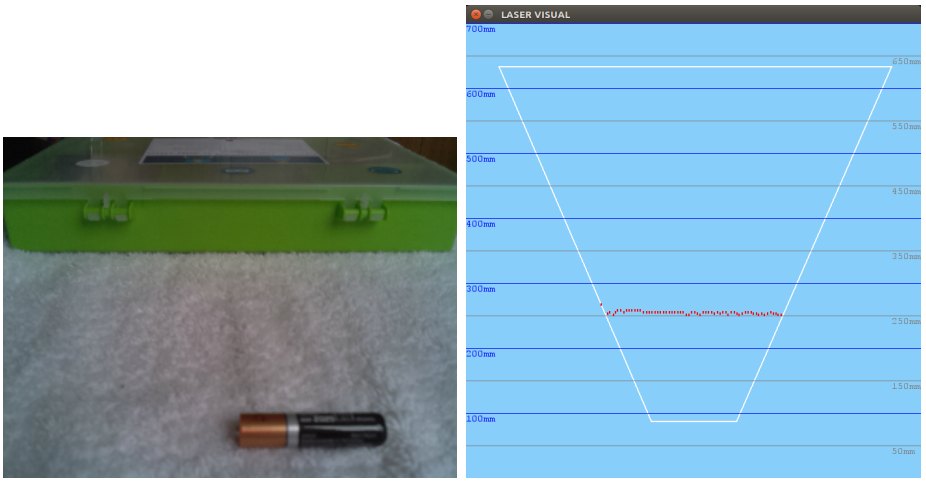


Figure 4.22: Example of visual sonar reading with 25 cm object shown using 3D scene simulator

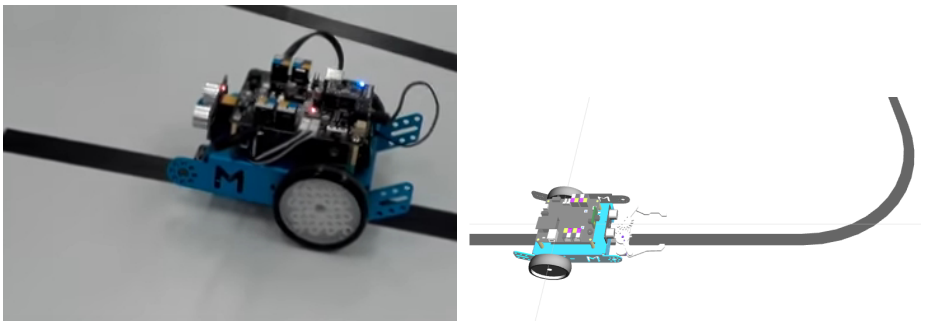


Figure 4.23: Practice line tracking task in real and simulated mBot robot

own procedures or learning paths based on what their own experience dictates.

The father of this approach is Ernst von Glasersfeld ([von Glasersfeld, 1995, Steffe and Gale, 1995]). According to this theory, students learn more when they are given the opportunity to explore and create knowledge that is of personal interest to them ([Lefoe, 1998]). This approach fits perfectly with the teaching of robotics, since students can experiment with a physical device, make mistakes and learn from them while working, thus building their own knowledge ([Vega and Cañas, 2014]).

In the sessions described in the academic program there is no differentiation between theoretical and practical. At the beginning of each class,

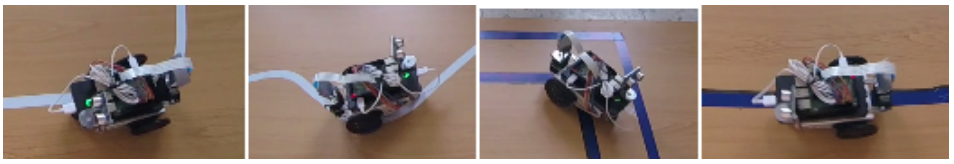


Figure 4.24: Practice line tracking task in PiBot robot using infrared

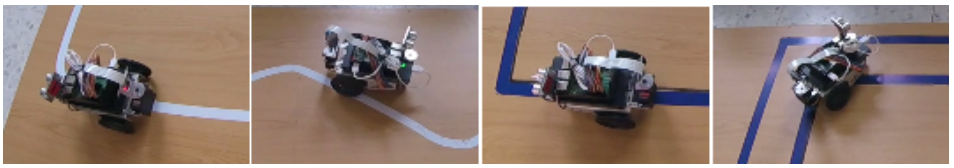


Figure 4.25: Practice line tracking task in PiBot robot using vision

what was learned in the previous session is remembered, some concepts that will be seen during the current session are mentioned, the objectives that will be reached (or should be) by the end of the session are explained, and all of this is contextualized with a challenge students have to pursue. This takes between five and ten minutes of class time. Subsequently, they are given full freedom to access all the available tools (computers, robots, and components) so they can decide how to distribute the time and what to do first. They can be corrected or advised if they stray from the path that will lead them to reach the proposed objectives.

In this way, the teacher becomes a guide rather than a strict setter of norms, guidelines and knowledge to be assimilated. Moreover, and continuing with the philosophy of cooperative learning, students always work on robotics in groups, because in this way they help each other and are not frustrated by failures, because there will always be some members who are sure of what to do.

Fifteen minutes before finishing the class, they are notified of the time left to finish the session. They then know they have five more minutes to finish, or save the work they are doing, since the last ten minutes are always reserved for reflection on how each group has learned and what each of them is learning individually. This final moment is suitable for clarifying issues and introducing (if necessary) some detailed and theoretical concepts. In this way the students acquire useful notions which are

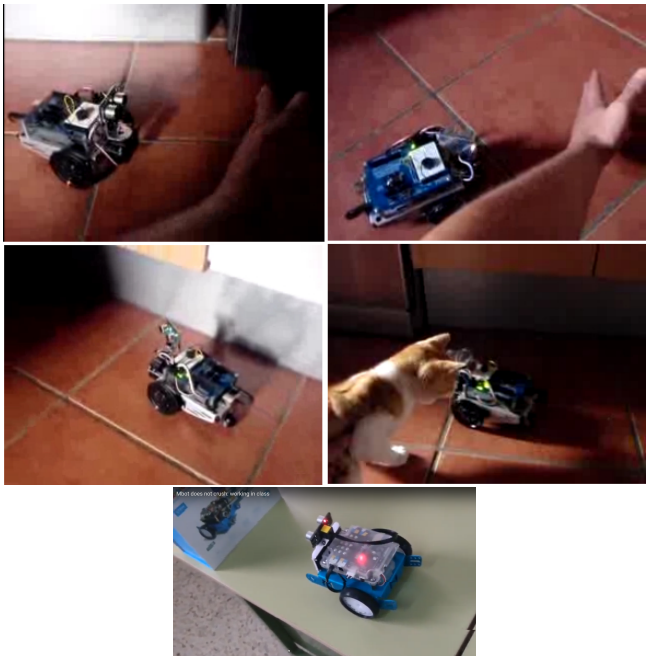


Figure 4.26: Navigation practice avoiding obstacles through US in Arduino and in mBot

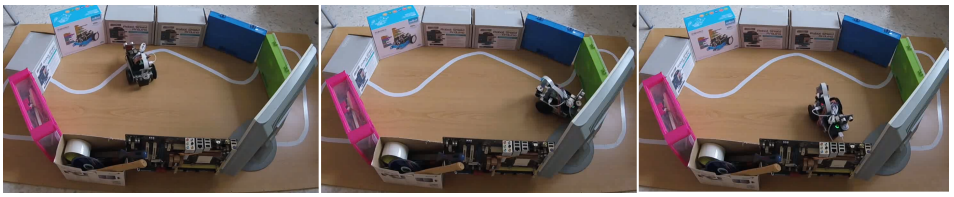


Figure 4.27: Navigation practice avoiding obstacles through US in PiBot

then fixed in their memory, since they have used them to solve a specific difficulty which they have experienced. Thus, in addition to the teacher ensuring they have a solid base of knowledge, the teacher becomes a *learning supervisor*.

Following this line of constructivist learning, regular assessments of knowledge lack meaning. In a certain way, evaluation is reversed, since the students assess themselves daily by giving a grade as a group, on how they consider they have been able to tackle the problem and solve it (in their case), as well as individually evaluating their contribution to the group. The teacher combines this student self-evaluation together with

Figure 4.28: Navigation exercise of following a color object using vision in PiBot

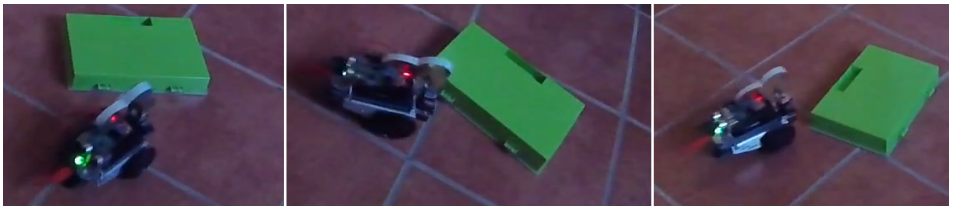


Figure 4.29: Exercise of avoiding obstacles by visual depth estimation in PiBot

their own assessment, based on the observation of the class both at group and individual levels, taking into account the potential of each student and the degree of effort of each one.

At the end of each topic a session is dedicated to reviewing what has been presented, what they have learned, what difficulties have been encountered and how they have been resolved. Likewise, the teacher comments on both the group and individual work and, consequently, the grade for each student in that unit. Thus, students are always aware of their strengths and weaknesses so they can try to balance them in the following units.

4.5 Deployment and results

In the 2016/2017 academic year, the proposed academic program was deployed with the `JdeRobot-Kids` teaching framework at the Franciscanas de Montpellier Foundation, which has six schools spread across Spain. In addition, it was also deployed in annual extracurricular subjects at the Ntra. Sra. Sagrado Corazón School in Madrid and the Villa de Móstoles School. In the 2017/2018 academic year, this program was continued in

the six schools of the Foundation and at Colegio Rihondo de Alcorcón.

As for the operating system used, 61.7% used Linux Ubuntu, 31.9% Microsoft Windows, and a small percentage (6.4%) used mobile device (iOS, Android). This was possible thanks to the fact that the proposed teaching framework operates seamlessly in any operating system, which greatly facilitated its deployment in the different educational centers.

The results were measured giving surveys to both, teachers and students, during the last two courses (2016/2017 and 2017/2018). Specifically, 2,050 students from the six schools of the Franciscan Foundation of Montpellier, the Villa de Móstoles School and the Rihondo School of Alcorcón were surveyed. All of these were of Secondary Education, distributed across curricular subjects (53.2%), extracurricular activities (36.2%), and a small percentage of specific events (10.6%) that are usually organized in the schools: Open Days, Family Days, etc. In total, nine teachers were responsible for delivering this content (six from the Foundation and three for extracurricular activities), who were also surveyed.

4.5.1 Student Surveys

In the question of whether it was easy to learn, more than 54% of students gave scores of 8-10, while a little fewer than 26% gave scores of 5-7. Taking into account that their initial level was very low or zero, and that the objectives of the educational proposal are quite ambitious, the results are more than positive: the framework is easy to learn.

More than 70% reported finding robotics very interesting (scoring between 8-10). More than 60% scored the materials received, the `JdeRobot-Kids` manual, between 8-10 while slightly fewer than 40% of students rated it between 5-7. More than 70% found the practice activities performed, i.e. the exercises, very interesting (8-10).

With all the above, the overall assessment given to the course with `JdeRobot-Kids` is shown in Figure 4.30.

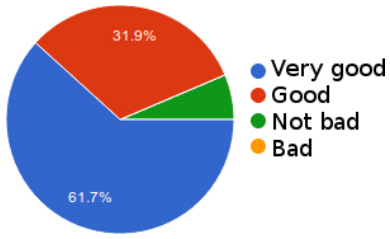


Figure 4.30: General assessment of JdeRobot-Kids by students

4.5.2 Teacher Surveys

The evaluation of the teaching staff regarding the deployment of our educational proposal is also very positive. The overall assessment of the educational proposal presented here is shown in Figure 4.31

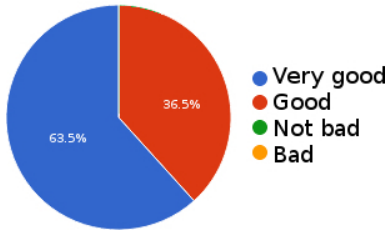


Figure 4.31: General assessment of the teaching staff on the educational proposal

In Franciscanas Foundation, five teachers scored 4/4 on the question of whether the students follow classes easily; another scored it with 3/4.

In all cases, they considered that the academic performance of their students improved: half of the respondents consider that the academic performance of their students improved, rating it 4/4, since the average grade of the class improved 2 points; while the other half rated it 3/4, given that in their cases the average score improved 1 point.

4.5.3 Discussion

The results were satisfactory. However, the surveys show slightly different ratings among students in the curricular and extracurricular classes. Arguable, because in the first case the students had a more limited time

and usually showed a high interest in the classes, while in the extracurricular classes they had more time but tended to be less interested.

In two of the schools, there had been no previous use of robotics; in two others little use, in another one moderate use and, finally, in two others a considerable use. Another positive indication is that after the deployment of our educational proposal, all the schools, without exception, have embraced robotics with great enthusiasm and have also held various competitions and workshops throughout the academic year.

```
# Below objects' border: camera image is processed from bottom to
    top (i=rows) and from left to right (j=columns)
while (j < ANCHO_IMAGEN): # processing columns
   i = LARGO_IMAGEN-1
   esFrontera = None
   while ((i>=0) and (esFrontera == None)): # processing rows
      pos = i*ANCHO_IMAGEN+j # actual position

      pix = bnImg[i, j] # value 0 or 255 (b/w border previously
          filtered image)

      if (pix != 0):
         esFrontera = True
         c = j - 1
         row = i
         v1 = row*ANCHO_IMAGEN+c

         if (not((c >= 0) and (c < ANCHO_IMAGEN) and
         (row >= 0) and (row < LARGO_IMAGEN))):
            pix = 0
         else:
            pix = bnImg[row, c]
         if (esFrontera == True):
            pixel.x = j
            pixel.y = i
            pixel.h = 1
            fronteraImg[i,j] = 255
            # backproject and intersect it with plane Z = 0
            pixelOnGround3D = getIntersectionZ (pixel)
            fronteraArray[puntosFrontera][0] = pixelOnGround3D.x
            fronteraArray[puntosFrontera][1] = pixelOnGround3D.y
            puntosFrontera = puntosFrontera + 1
      i = i - 1
   j = j + 5
```

Code 4.2: Getting frontier border below objects in image

```
images = glob.glob('chess_board/*.png')

for file_name in images:
    image = cv2.imread(file_name)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    h, w = gray.shape[:2]

    # find chess board corners
    ret, corners = cv2.findChessboardCorners(gray, (9, 6), None)

    # add object points, image points
    if ret:
        object_points.append(object_point)
        cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1),
            criteria)
        image_points.append(corners)

        # draw and display the corners
        cv2.drawChessboardCorners(image, (9, 6), corners, ret)
        cv2.imshow('image', image)
        cv2.waitKey(500)

# calibration
retval, cameraMatrix, distCoeffs, rvecs, tvecs =
    cv2.calibrateCamera(object_points, image_points, (w, h), None,
    None)
```

Code 4.3: PiCamera calibrator using OpenCV library

```
def getIntersectionZ (p2d):
   p3d = Punto3D ()
   res = Punto3D ()
   p2d_ = Punto2D ()

   x = myCamera.position.x
   y = myCamera.position.y
   z = myCamera.position.z

   p2d_ = pixel2optical(p2d)
   result, p3d = backproject(p2d_, myCamera)

   # Check division by zero
   if((p3d.z-z) == 0.0):
      res.h = 0.0
      return

   # Linear equation (X-x)/(p3d.X-x) = (Y-y)/(p3d.Y-y) =
       (Z-z)/(p3d.Z-z)
   xfinal = x + (p3d.x - x)*(zfinal - z)/(p3d.z-z)
   yfinal = y + (p3d.y - y)*(zfinal - z)/(p3d.z-z)
   zfinal = 0. # Ground plane Z = 0

   res.x = xfinal
   res.y = yfinal
   res.z = zfinal
   res.h = 1.0

   return res
```

Code 4.4: Intersection between optical ray and below border of object

```
thetaY = 86.7*DEGTORAD # camera is rotated 86.7 degrees over Y axis
thetaZ = 0*DEGTORAD # camera is not rotated over Z...
thetaX = 0*DEGTORAD # ...nor X axis

# R_y is a 3x3 rotation matrix
R_y = numpy.array
    ([(numpy.cos(thetaY),0,-numpy.sin(thetaY)),(0,1,0),
  (numpy.sin(thetaY),0,numpy.cos(thetaY))])
# R_z is a 3x3 rotation matrix
R_z = numpy.array ([(numpy.cos(thetaZ),-numpy.sin(thetaZ),0),
  (numpy.sin(thetaZ),numpy.cos(thetaZ),0),(0,0,1)])
# R_x is a 3x3 rotation matrix
R_x = numpy.array
    ([(1,0,0),(0,numpy.cos(thetaX),numpy.sin(thetaX)),
  (0, -numpy.sin(thetaX),numpy.cos(thetaX))])

R_subt = numpy.dot (R_y, R_z)
R_tot = numpy.dot (R_subt, R_x)

# T is a 3x4 traslation matrix
T = numpy.array ([(1,0,0,0),(0,1,0,0),(0,0,1,-110)])
Res = numpy.dot (R_tot,T)
# RT is a 4x4 matrix
RT = numpy.append(Res, [[0,0,0,1]], axis=0)
# K is a 3x4 matrix with intrinsics values got from
    piCamCalibrator tool
K = numpy.array ([(313.89382026,0,117.5728043,0),
(0,316.64906146,158.04145907,0),(0,0,1,0)])
```

Code 4.5: Operations with camera matrices to get absolute position

```
from mBotReal import MBotReal
from mBotGazebo import MBotGazebo
from piBot import PiBot

if __name__ == "__main__":
   ic = EasyIce.initialize(sys.argv)
   props = ic.getProperties()
   propPrefix = "JdeRobotKids"
   robot = props.getProperty(propPrefix + ".Robot")

   myJdeRobotKids = init(robot)

   # TODO: add your code below here
   print ("Codigo choca-gira con robot ", myJdeRobotKids.tipo)

   vel = 100
   while True:

   val = myJdeRobotKids.leerUltrasonido()
     print(val)
     if (val < 10):
      myJdeRobotKids.retroceder(vel)
      time.sleep(0.5)
      myJdeRobotKids.girarDerecha(vel)
      time.sleep(0.3)
     myJdeRobotKids.avanzar(vel)
     time.sleep(0.3)
```

Code 4.6: Shock-tour using JdeRobot-Kids on any platform

Chapter 5

Conclusions

In this chapter the main conclusions and contributions of the presented work are recapitulated. It is organized in four sections. In the first one, Section 5.1, the conclusions are presented, after evaluation of the academic environment put into practice. Then, in Section 5.2, the contributions made in the central theme of the thesis are listed. Section 5.3 takes a tour of the different publications that derive from this research. Finally, the future lines that follow this dissertation are described in Section 5.4.

5.1 Conclusions

This research is focused on incorporating Robotics and robots with vision in the classroom to train pre-university students, satisfying the demands imposed by the *Digital Age Society* and the motivational needs detected in students, who still study in a system of training still to be adapted to the so-called *Industrial Revolution 4.0*.

Although there are numerous educational Robotics kits on the market, most of them are aimed at younger students. They are generally based on building *their* robotic platforms with *their* own programming environments, far from employing more standardized programming languages. In addition, they usually have a somewhat low level of complexity, which

means that these tools tend to be —in the short term— trigger a low level of motivation in students. Furthermore, given the complexity involved in the treatment of a sensor such as the camera, despite its great versatility, it is not usually included in these educational environments.

Based on this, and in the first place, several algorithms have been implemented that use a camera as the main sensor to solve the fundamental problems of Robotics, such as navigation and location. Both fronts have been solved by developing a visual memory in which the robot is able to incorporate and abstract complex elements following different patterns: arrows, parallelograms or human faces. On this visual memory, an attentive system capable of attending to the different elements that surround the robot is mounted. This has resulted in an autonomous and intelligent system, which navigates through a changing and real environment, in which it is constantly located. This visual perception system was validated in both real and simulated robots. The memory accurately represents the environment of the robot using the images of the mobile camera, whose movement is controlled by the attention mechanism. Memory is dynamic but has some persistence to treat temporary occlusions. The location works in real time, providing position errors below 15 cm and 5 degrees and is robust enough to recover from seizures or estimation errors in symmetric environments.

Secondly, after investigating the market situation of the existing Robotics educational kits and conducting an in-depth analysis what the future holds in the short and mid-term in terms of demands of the labor market, the author, as an experienced Secondary Education teacher, detected a deficiency in the teaching-learning process of Robotics at pre-university curricular level. Therefore, a complete educational environment was developed, which includes:

- A robotic platform based on the free hardware controller board Raspberry Pi 3. This platform was chosen for several reasons: low cost, power, versatility, standardization and inclusion of a camera with its own data bus, the PiCamera. Thus, a fully functional robot was built, the `PiBot`, to which —thanks to the GPIO ports on the board— various sensors and actuators have been connected, in addi-

tion to its own camera.

- A software infrastructure developed in Python language, JdeRobot-Kids, which facilitated students' programming of the robot, with simple and intuitive functions to handle the different sensors and actuators, but at the same time of great potential, as those corresponding to the handling of a camera as a sensor.

- A wide repertoire of practice activities that serve as a support to students for their progression in the learning of the programming of robots with vision.

- An academic program that includes the plan of activities for the subject *Programming, Robotics and Technology* in different Secondary Education levels and for a course of extracurricular activities. This program is divided into four phases, starting from a very basic level up to the development of a project that solves a classic Robotics task, such as a line-follow or crashes-turns robot using the camera as a sensor.

The teaching environment `JdeRobot-Kids` was used in the Franciscanas de Montpellier Foundation during the 2016/2017 and 2017/2018 academic years in the subject *Programming, Robotics and Technology* in 1st, 2nd and 4th years of Compulsory Secondary Education (CSO) and in annual extra-curricular subjects at the Ntra. Sra. Sagrado Corazón School in Madrid, as well as in extra-curricular subjects at the Villa de Móstoles School during the 2016/2017 academic year and at the Rihondo School in Alcorcón during the 2017/2018 academic year.

In total, the environment was followed by some $2,050$ students and a dozen teachers in the last two academic years. Its impact was measured through surveys and the results were very satisfactory (Section 4.5). They show students and teachers received the subject well and were highly satisfied. In addition, the robotic projects carried out by the students demonstrate a high level of assimilation of concepts, while the dynamics of the classes were very pleasant.

5.2 Contributions

From this work derive, in addition to the publications listed in Section 5.3, numerous software and hardware developments related to Robotics with vision and education in Robotics. The main ones are listed below:

(C_1) *Attentive visual system*[1] on a mechanical neck, which has allowed the robots to have a perceptive system with a wide field of vision of the entire surrounding scene, greater than the instantaneous field of view of a camera. In addition, this system is essential to indicate to the robot which is the next area to look at, in addition to focusing on the object that is being focused on at a certain time, and follow it.

(C_2) *Visual memory*[2], which is responsible for maintaining information about the different objects of interest that the robot finds in the environment it navigates, which allows it to make smarter decisions in real time about the surrounding circumstances. It includes several main developments: (a) the object detector, in charge of identifying basic forms (concepts) such as arrows, parallelograms and human faces; (b) the element prediction mechanism, which allows the system to predict previously memorized elements, alleviating the computational cost; and (c) the algorithm for generating perceptual hypotheses, responsible for abstracting complex objects, which has allowed the perceptual system to successfully resolve possible occlusions that can occur in a real environment.

(C_3) *Auxiliary vision library*[3], which includes all the functionality to satisfy the different changes in coordinated systems between the objects detected in the real world and the pixels corresponding to the 2D image in a camera. The development of a class to completely abstract the model of an ideal Pin-Hole camera, has allowed students who start in robots with vision to perform complex vision practical sessions without the need for advanced knowledge in the subject. It

[1] `http://jderobot.org/VisualSonar`
[2] `http://jderobot.org/VisualMemory`
[3] `http://jderobot.org/RobotVision`

also includes algorithms based on the Floor Hypothesis and the implementation of the Solis line extractor ([Solis et al., 2009]), which have allowed the distances to objects to be estimated using a single camera.

(C_4) *Design and construction of PiBot*[45] as a robotic platform based on free and standard hardware, specifically on the Raspberry Pi 3 board. The power, versatility and inclusion of the PiCam camera with its own dedicated data bus has allowed the development of vision algorithms. In addition, the low cost of this platform allows schools to acquire a considerable number of prototypes in order to satisfy a whole class. The inclusion of powerful servos gives rise to a robust, reliable and agile mobile robotic platform to navigate in a real environment.

(C_5) *Repertoire of practices in Python on mBot and Arduino standard.* It allowed students and teachers to have an extensive collection of fully functional examples, implemented in a real language such as Python, and all sensors and actuators that can be coupled to an Arduino UNO board as well as those already included as standard in mBot robot. This greatly facilitated the inclusion of Robotics in Secondary Education classrooms.

(C_6) *Software infrastructure to support PiBot*[6], implemented in Python language. It allowed the inclusion of the powerful robotic platform PiBot in the classroom, which can be used by young students who lack the necessary knowledge to manipulate such a sophisticated robot. This infrastructure facilitates the complete management of the robot: servos, sensors (ultrasound, infrared) and varied actuators (push-buttons, LEDs) and, most importantly, a real camera, such as the PiCam or any WebCam, which was supported by the

[4] `http://jderobot.org/JulioVega_PhD#2018.03.03._Introducing_the_new_PiBot_v3.0`

[5] `https://github.com/JdeRobot/JdeRobot/tree/master/assets/gazebo/models/pibot`

[6] `https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/PiBot`

driver `piCamServer`[7]. This allowed students to program numerous challenging practice activities, using a simple camera as the main sensor.

(C_7) *Repertoire of practice sessions in Python on PiBot.* Practice sessions similar to those of the mBot but adapted to this other platform and extended with several exercises that require using the camera on board.

(C_8) *Full educational environment*[8]. With all the above, a complete educational program was designed and followed, with an academic program of adequate complexity for pre-university students. All the sessions followed, with their contents and objectives, were detailed and organized, as well as the methodology that has been found to be the most appropriate for the Teaching-Learning process of Robotics, based on the experience of these years in Spanish schools and a research stay in Finland.

In total, more than 200,000 lines of code were implemented among the different software developments described. The distribution of these is reflected in Figure 5.1.

The progress of all the work carried out for this dissertation can be followed in the dedicated Wiki[9], where all the implemented codes can be accessed. These works have been integrated into the JdeRobot[10] platform of the Robotics Group of the Rey Juan Carlos University under the framework of several major projects: GuideRobot[11], RobotVision[12], VisualMemory[13], VisualSonar[14] and JdeRobot-Kids[15].

The software contributions include the use of different languages such as `C`, `C++` and `Python`, as well as numerous libraries and tools, among which

[7] `https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/piCamServer_py`
[8] `http://kids.jderobot.org`
[9] `http://jderobot.org/JulioVega_PhD`
[10] `https://jderobot.org`
[11] `http://jderobot.org/GuideRobot`
[12] `http://jderobot.org/RobotVision`
[13] `http://jderobot.org/VisualMemory`
[14] `http://jderobot.org/VisualSonar`
[15] `http://kids.jderobot.org`

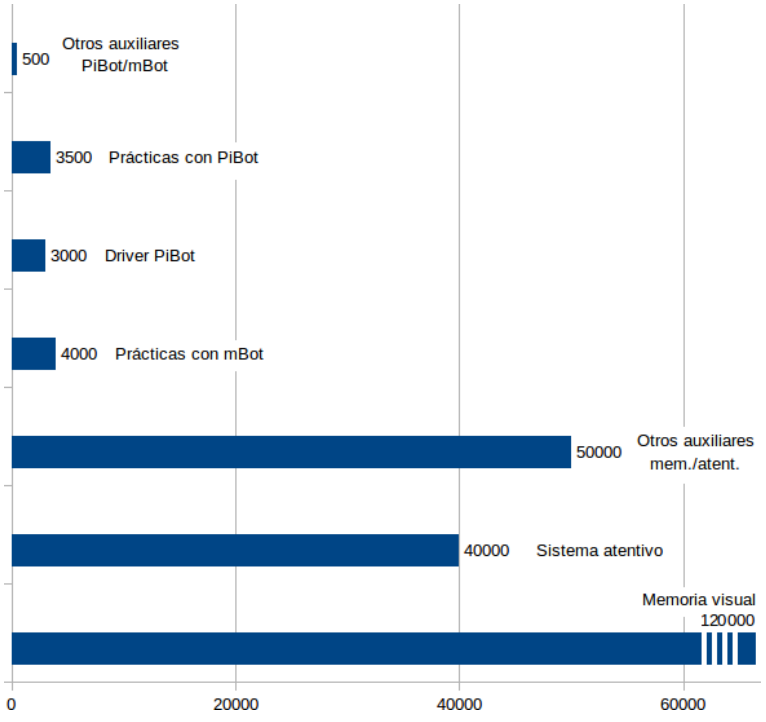


Figure 5.1: Distribution of the lines of code implemented

the following stand out: `Susan`, `XForms`, `Glade`, `GTK`, `FreeGLUT`, `OpenGL`, `OpenCV`, `v4l`, `GSL`, `ICE` , `Math`, `PiGPIO`, `PyGame`, `IMUtils`, `NumPy`, `SVN`, `GIT`, `UMLet`, etc.

5.3 Publications

The publications that derive from this work are the following:

- Journals:
  - Julio Vega, José María Cañas, Fran Pérez and Aitor Martínez. (2018). *JdeRobot-Kids: entorno docente de robótica para niños.* **Revista Iberoamericana de Automática e Informática Industrial (RIAI)**. Under review. Impact factor (JCR 2016): 0,500.

  - Julio Vega, Eduardo Perdices and José María Cañas. (2013). *Robot evolutionary localization based on attentive visual short-term memory*, pages 1268-1299. **Sensors**. ISSN: 1424-8220. Impact factor (JCR 2013): 2,048.

  - Julio Vega, José María Cañas and Eduardo Perdices. (2012). *Local robot navigation based on an active visual short-term memory*, pages 21-30. **Journal of Physical Agents**. ISSN: 1888-0258. Impact factor (SJR 2012): 0,171.

- Book chapters:

  - Julio Vega, Eduardo Perdices and José María Cañas. (2012). *Attentive Visual Memory for Robot Localization*, pages 408-438. **Robotic Vision: Technologies for Machine Learning and Vision Applications**. Ed. IGI-GLOBAL. ISBN: 978-84-694-6730-5.

- Workshops:

  - Julio Vega and José María Cañas. (2016). *Entorno docente con Arduino y Python para Educación Robótica en Secundaria*. **JITICE 5th Workshop, Educational Innovation and ICT**. ISBN: 978-84-697-0892-7.

  - Julio Vega and José María Cañas. (2014). *Curso de Robótica en Educación Secundaria usando Constructivismo Pedagógico*. **JITICE 4th Workshop, Educational Innovation and ICT**. ISSN: 2172-6620.

  - José María Cañas, Laura Martín and Julio Vega. (2014). *Innovating in robotics education with Gazebo simulator and JdeRobot framework*. **CUIEET XXII Congreso Universitario de Innovación Educativa en Enseñanzas Técnicas**. ISSN: 2172-6620.

- Borja Menéndez, José María Cañas, Eduardo Perdices and Julio Vega. (2013). *Programming a Humanoid Social Robot Using the JdeRobot Framework.* **RoboCity2030 11th Workshop, Robots sociales**. ISBN:978-84-695-7212-2.

- Julio Vega, Eduardo Perdices and José María Cañas. (2012). *Robot evolutionary localization based on attentive visual short term memory.* **IEEE Intelligent Vehicles Symposium Workshops, Perception in Robotics**. ISBN: 978-84-695-3472-4.

- Julio Vega and José María Cañas. (2011). *Attentive visual memory for robot navigation.* **WAF2011 XII Physical Agents Workshop**. ISBN: 978-84-694-6730-5.

- Eduardo Perdices, José María Cañas, Julio Vega, Carlos Agüero and Francisco Martín. (2010). *Localización visual de robots en la RoboCup mediante algoritmos evolutivos.* **Robocity 2030**. ISBN: 84-693-6777-3.

- Julio Vega, José María Cañas, Pablo Miangolarra and Eduardo Perdices. (2010). *Memoria visual atentiva basada en conceptos para un robot móvil.* **Robocity 2030**. ISBN: 84-693-6777-3.

- Julio Vega and José María Cañas. (2009). *Sistema de atención visual para la interacción persona-robot.* **RoboCity2030, Interacción persona-robot**. ISBN: 978-84-692-5987-0.

5.4 Future lines

It is expected that the teaching environment developed, `JdeRobot-Kids`, and the methodology followed will contribute in the long term to improving results on educational indicators in Spain, to reduce the gap in educational quality with other countries such as Finland.

The main lines of work in the short term are the following: *(a)* to unify the programming interface for the real PiBot and the existing one for the simulated PiBot, so that students' practical sessions can be executed without modifications in the real robot or the simulated one; *(b)* to develop new practical sessions with vision such as the detection and monitoring of people's faces, and materialize in the PiBot a visual attentive system and a visual memory; and *(c)* to disseminate the developed environment.

Regarding this last point, during the month of July of this year a Robotics workshop is to be taught at the Campus of Fuenlabrada of the Rey Juan Carlos University. In addition, a manual is being prepared to serve as a textbook for the teacher, with all the theoretical concepts of Robotics and Electronics, the installation of the environment and the whole repertoire of practical sessions developed for the different hardware platforms: Arduino, mBot and PiBot. It is also intended to publish an article in a high-impact journal with all these advances, and fundamentally with the new platform developed, `PiBot`.

Looking to the next academic year, we intend to continue implementing this teaching environment in the schools already using it. It will also be extended to other schools such as the Carpe Diem Secondary School in Fuenlabrada and many others where the environment will be followed in extracurricular activities thanks to collaboration with the company Logix5[16].

Work is also being done to extend the environment to a Web-IDE of `JdeRobot-Kids`, which will allow students to work in the cloud, without the need for them to install a library on their computer, greatly facilitating work in the classroom. This will facilitate programming practices with the real PiBot platform, by downloading in the Raspberry the code developed in the IDE of the cloud; and with simulations, thanks to the simulator incorporated in the web environment.

Finally, the project `PyOnArduino`[17] is being developed, whose objective is that the practice sessions developed in Python for mBot use a Python library that runs on the personal computer and that is in continuous communication (wired or wireless) with the mBot. That is, they do not run

[16] `http://www.logix5.com/roboticaeducativa`
[17] `https://github.com/JdeRobot/PyOnArduino`

directly on the Arduino processor of the mBot, but on the PC. On board the mBot, a data trafficator is executed that continuously captures the sensors and sends them to the Python library on the PC; in turn, it receives commands from the PC for the actuators and orders them to the engines on board. A possible continuation of this thesis is to develop translator from Python to Arduino language, so that student applications written in Python can be translated into Arduino language and thus loaded on board to actually run inside the mBot.

Apéndice A

Resumen en castellano

En este apéndice se resume el contenido de este trabajo en castellano. Está organizado según los siguientes apartados. En la Sección A.1 se muestra una descripción general de la situación del área de Robótica, su papel en la sociedad actual y del futuro cercano y de cómo la visión de robots es fundamental en esta nueva era de robots inteligentes. La Sección A.3 proporciona un breve análisis del auge en todo el mundo de la Robótica y la programación a nivel educativo. En la Sección A.4 se presenta la propuesta educativa desarrollada en esta disertación. La Sección A.5 describe los principales objetivos de este trabajo. En la Sección A.6 se hace un recorrido por la estructura según la cual está vertebrada esta tesis. En los cuatro últimos apartados se recapitulan las principales conclusiones y aportaciones del trabajo presentado. En la Sección A.7 se presentan las conclusiones, previa evaluación del entorno académico puesto en práctica. Seguidamente, en la Sección A.8, se enumeran las contribuciones hechas en el tema central de la tesis. La Sección A.9 hace un recorrido por las distintas publicaciones que se derivan de esta investigación. Por último, las líneas futuras que se abren paso tras esta disertación se describen en la Sección A.10.

A.1 Tecnología y Robótica

En la última década, la tecnología se ha convertido cada vez más común en la mayoría de los aspectos de la vida cotidiana e industrial. En los hogares se dispone cada vez más de un mayor número de dispositivos tecnológicos, tales como ordenadores, *tablets*, teléfonos inteligentes —o *smartphones*—, sistemas de domótica, etc. Todos ellos intercontectados a través de Internet. A nivel industrial, cada vez son más las factorías que incorporan en sus cadenas de producción ya no máquinas autónomas, sino robots inteligentes con sofisticados sistemas sensoriales, con la visión como principal mecanismo de percepción.

A.1.1 La Robótica en el ámbito doméstico

En casa es habitual disponer de numerosos elementos tecnológicos; la aparición de dispositivos robóticos en el mercado masivo como las aspiradoras y mopas robóticas (Figura A.1-a,b), así como las numerosas aplicaciones y servicios de domótica existentes (Figura A.1-c), hacen que esta tecnología esté cada vez más presente en la rutina diaria de la sociedad, por no hablar de otras tareas automatizadas y que se dan frecuentemente: sacar dinero del cajero automático del banco, el pago automático en supermercados, o el uso masivo de Internet para comunicarnos, realizar compras, llevar a cabo gestiones bancarias, y mucho más.

Figura A.1: iRobot Roomba y Jet Braava, y aplicación domótica Wattio

Por otro lado, los coches autónomos o los drones hacen más visible la utilidad de esta tecnología y refuerzan su atractivo. Los grandes fabricantes de automoción están empujando estos nuevos avances, tienen prototipos avanzados de coches autónomos y grandes empresas de software como Goo-

gle (Figura A.2) o Apple, o nuevas como Tesla se han posicionado muy bien en este sector.

Figura A.2: Coche autónomo de Google y dron dedicado a tareas de agricultura

Por su parte, el uso de drones como ocio está quedándose en un segundo plano para dar paso a su cada vez más extendido uso a nivel profesional (Figura A.2) para labores de emergencia, eventos, búsqueda de personas, control fiscal, vigilancia fronteriza, agricultura, vigilancia de tráfico, etc. En España ya existen casi 3.000 empresas habilitadas en este sector y el Ministerio de Fomento ha impulsado recientemente el Plan Estratégico del uso de drones 2018-2021 ([Fomento, 2018]) para el desarrollo del sector civil de los drones, que establece la hoja de ruta a seguir para impulsar el desarrollo de este sector incipiente y con un alto potencial de crecimiento.

Debido a esta tendencia hacia la automatización de casi todas las tareas cotidianas así como un aumento en la presencia de dispositivos robóticos presentes en el día a día, cada vez se hace más imprescindible el conocimiento del uso de las tecnologías.

A.1.2 La Robótica y la Industrialización

La Robótica a nivel industrial tiene sus orígenes en la Revolución Industrial de 1800 cuando, por primera vez, productos y servicios eran desarrollados por máquinas; teniendo como buque insignia la máquina de vapor, que hizo que muchos empleados fueran reemplazados por ellas.

La segunda revolución industrial comenzó con la electricidad, a finales del siglo XIX, cuyo principal concepto novedoso fue la línea de montaje, que se utilizó por primera vez en la industria del automóvil (Figura A.3-a).

La tercera revolución industrial comenzó en la década de 1970 y se distinguió por una mayor automatización a través de la electrónica. Comienzan a integrarse en la sociedad los primeros ordenadores personales,

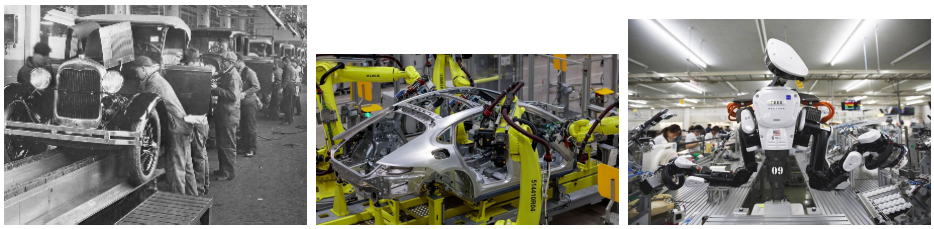


Figura A.3: Revoluciones industriales: (a) Línea de montaje de Ford, (b) Producción en serie de Porsche, (c) Robots inteligentes de Glory Ltd.

Internet y el acceso global a información. A nivel laboral el trabajo humano es reemplazado por máquinas que, programadas, fabrican productos en serie (Figura A.3-b), donde la velocidad, la precisión y la fiabilidad son primordiales.

La denominada *Industrialización 4.0* supone la integración de sistemas robotizados complejos en las factorías (Figura A.3-c), la logística y el llamado *Internet de las cosas*, donde sofisticados autómatas manejan una inmensa cantidad de datos para tomar decisiones estratégicas para las empresas.

El futuro a corto y medio plazo está/estará marcado por una producción industrial dominada por máquinas inteligentes. La presencia de humanos en estas *factorías inteligentes* tiende a ser cada vez menor y llegará a ser simbólica y puntual. No hay duda de que la capacidad de toma de decisiones óptimas en tiempo real de una máquina, que maneja una enorme gran cantidad de datos simultáneamente, dista mucho de la capacidad de un ser humano.

A.1.3 Robots inteligentes en entornos complejos

Estos robots móviles e inteligentes necesitan, además de una gran capacidad de cómputo, un complejo sistema sensorial para *actuar* inteligentemente ya no sólo en las factorías sino en la interacción robot-humano a nivel general ([Vega and Cañas, 2009]). La automatización fija de cadenas de producción estructuradas deja paso al mundo impredecible y nada estructurado de *la realidad* donde se hace evidente la necesidad de una amplia gama complementaria de sensores y actuadores para lograr una

completa autonomía ([Arbel and Ferrie, 2001]).

Esta lucha por la autonomía en entornos complejos, desestructurados, impredecibles y cohabitados por los humanos constituye actualmente un profundo campo de investigación en Robótica, la *Robótica Inteligente*, donde la percepción por visión, el razonamiento y la actuación están íntimamente ligados a realizar tareas útiles con poca intervención humana. Cabe mencionar en este punto el asistente inteligente Duplex, que la compañía Google presentaba en mayo de este año y que es fiel reflejo de cómo están evolucionando las máquinas.

A.2 Visión en Robótica

La visión computacional es la rama de la Inteligencia Artificial que incluye las técnicas y métodos aplicados a una cámara como sensor. Aunque esta modalidad sensorial no ha sido la más empleada hasta hace unos años en robótica móvil (sónar y/o láser han sido mayoritariamente usados como sensores, Figura A.4-a,b), actualmente se ha convertido en el sensor que más se está usando y —sin duda— se usará con mayor profusión a largo plazo, debido a las posibilidades que ofrece y a la potencia de cálculo de los ordenadores hoy en día. Son dispositivos de bajo coste y potencialmente muy ricos, ya que ofrecen mucha información.

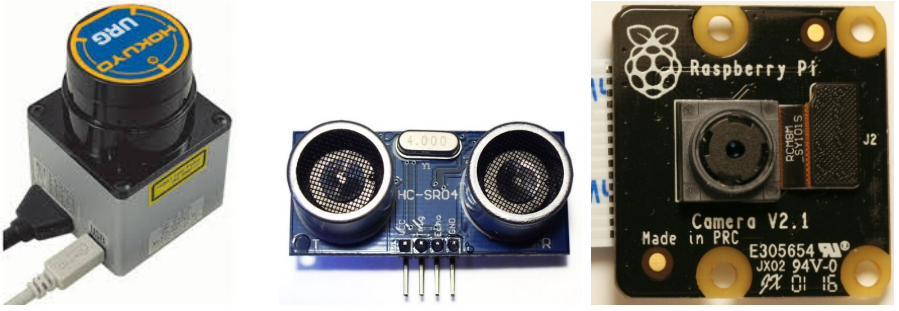


Figura A.4: Sensores láser de Hokuyo, ultrasonidos HC-SR04 y cámara PiCam

Pero la capacidad visual en robots, al contrario que en los animales, no resulta una técnica fácil ([Ramachandran, 1990]). La principal dificultad radica en extraer información útil del gran caudal de datos que vierte una cámara (Figura A.4-c), para lo cual se necesitan buenos algoritmos. Aun-

que debemos ser cautos a la hora de comparar un robot con un organismo biológico ([Nehmzow, 1993]), lo que sí está claro es que la vista es el sentido principal en que se apoyan los animales para moverse por el entorno ([Tinbergen, 1951]).

Figura A.5: Sistemas avanzados de percepción visual Bumblebee, Mobile Ranger y Kinect

Aunque en el mercado existen diversos dispositivos de percepción que combinan visión e infrarrojos para obtener una mayor información útil del entorno sin necesidad de un gran procesamiento de los datos (Figura A.5, en este trabajo de investigación se han usado cámaras sencillas, centrando los esfuerzos en desarrollar complejos algoritmos capaces de extraer información de interés de los datos vertidos por estas.

En la literatura existen diversos frentes de investigación abiertos en cuanto a visión de robots. A continuación se describen los usos más habituales.

Detección de estímulos mediante redes neuronales

En las últimas décadas surgieron modelos de redes neuronales artificiales (RNA) postuladas de diversas teorías del funcionamiento de redes neuronales biológicas. Desde entonces, han sido numerosos los trabajos propuestos donde se aplican RNAs en diferentes áreas de la ingeniería. Un área donde se utilizan ampliamente dichas redes es la de procesamiento de imágenes, donde existe una gran cantidad de trabajos propuestos con RNAs ([Schmidhuber, 2015]).

Por ejemplo, los coches autónomos aplican técnicas de redes neuronales para gestionar las trayectorias y los controles de posición y orientación ([Caceres et al., 2017]).

Control visual

Otro uso extendido de la visión de robots es el control visual, por ejemplo para detección de caras ([Vega and Cañas, 2009]), sistemas de videovigilancia ([Srinivasan et al., 2006]), control de tráfico (cámaras con sistema de detección de cinturón de seguridad, etc.), en coches autónomos para detectar cambio de carril ([Cho et al., 2014]). También, existe un enorme campo de investigación en Biometría, para reconocimiento de firmas, caracteres y estudio del trazado de la escritura ([Pinto et al., 2015]).

Reconstrucción 3D

La adquisición de modelos tridimensionales es desde hace algunos años uno de los frentes de investigación que presenta mayor actividad. El uso de estos escenarios 3D es muy amplio, desde desarrollar entornos de realidad virtual ([Barrera et al., 2005]) hasta videojuegos ([Richter et al., 2016]), así como en la reconstrucción de edificios antiguos ([Murgul, 2015]) o modernos ([Fathi et al., 2015]) para su análisis.

Autolocalización visual

La autolocalización visual o basada en imágenes se refiere a la recuperación de la posición y orientación de una cámara en el mundo en función de las imágenes que graba. Esta aplicación es de interés en entornos donde los sistemas basados en GPS no están disponibles o son imprecisos, como en interiores o en ciudades densas ([Antequera et al., 2017]).

Así, aplicaciones que realizan sus tareas en interiores, como *Roomba* de *iRobot*, emplean estas técnicas para saber en todo momento su posición en el entorno en el que están efectuando su tarea y evitar, por ejemplo, comportamientos poco inteligentes como repetir en zonas en las que ya ha estado, etc.

También, los coches autónomos incluyen técnicas de autolocalización visual ([Wolcott and Eustice, 2014]), lo que les permite navegar con seguridad en zonas o momentos donde la recepción de satélites GPS es de poca calidad, como ciudades con edificios altos, túneles o momentos en los que por inclemencias meteorológicas es imposible visualizar estos satélites.

Atención visual

A la hora de tratar y seleccionar detalles útiles de toda la información percibida por una cámara, resulta crucial fijarse en cómo funcionan los sistemas visuales de los organismos que existen en la naturaleza ([Zaharescu et al., 2005]). Los humanos disponemos de un preciso sistema de visión activa ([Bajcsy, 2009, Sawides et al., 2018]). Esto significa que podemos concentrarnos en determinadas regiones de interés de la escena que nos rodea ([Marocco and Floreano, 2002]) gracias al movimiento de los ojos ([Murray et al., 2003]) y/o de la cabeza ([Vega and Cañas, 2009]), o simplemente repartiendo la mirada ([Arbel and Ferrie, 2001]) en distintas zonas dentro de la imagen actual que estemos percibiendo ([Itti and Koch., 2005]).

La atención visual es una tarea clave en la Robótica autónoma, pues un robot con visión que interactúa en un entorno real ha de ser reactivo, por lo que ha de incluir sistemas de visión rápidos que extraigan información útil en tiempo real de los datos que vierte una cámara.

Memoria visual

Si un robot con visión además de tratar la información que recibe en cada momento de su sistema visual puede almacenar esta en una memoria, podrá tomar decisiones más inteligentes a la hora de moverse por el entorno. Es por ello que en este frente de investigación existen desde hace unos años numerosos trabajos.

Los coches autónomos, las aplicaciones domésticas como *Roomba*, o los sistemas autónomos de navegación aérea incorporan esta memoria.

En el tema de la tesis se desarrollan métodos sobre qué y dónde se encuentran los objetos que el robot va encontrando a su paso ([Zaharescu et al., 2005]), lo que permitirá al robot navegar por el entorno circundante de forma inteligente y estar localizado en él.

A.3 Robótica educativa

Como se ha descrito en la Sección A.1.2 el avance de la Inteligencia Artificial (AI), la Robótica y la automatización en la sociedad, el futuro del trabajo y la industria en particular ([Mies and Zentay, 2017]) confluyen en lo que ya se denomina la cuarta revolución industrial ([Schwab, 2016]).

Según los análisis de la Universidad de Oxford ([Frey and Osborne, 2013]) y la firma de servicios profesionales Deloitte ([Deloitte, 2015]), casi la mitad de todos los puestos de trabajo serán ocupados por robots en los próximos 25 años. Además, como señala en su último informe sobre economía global ([Institute, 2017]) el instituto Mckinsey([1]), los robots realizarán la labor de cerca de 800 millones de puestos de trabajos en 2030.

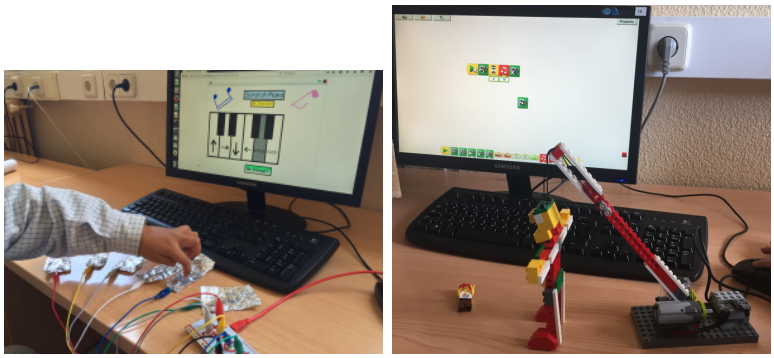


Figura A.6: Diferentes prototipos robóticos para trabajar distintas áreas educativas

Por ello resulta de vital importancia incorporar la Tecnología, y en concreto la Robótica, en el sistema educativo preuniversitario ya que serán los más jóvenes de ahora los que tengan que enfrentarse en una década a un mercado laboral que demandará perfiles relacionados con la automatización de sistemas ([UK-RAS, 2016]). Desde el punto de vista educativo, la Robótica es un campo transversal donde concurren muchas áreas: electrónica, física (Figura A.6-a), mecánica (Figura A.6-b), informática, telecomunicaciones, matemáticas, etc.

Además, el uso de las tecnologías en las escuelas públicas y privadas también ayudará a reducir la *brecha digital* que existe actualmente en to-

[1] `https://www.mckinsey.com`

Figura A.7: Alumnos trabajando en un prototipo robótico

do el mundo. La brecha digital es el término para describir las grandes diferencias en el uso de la tecnología entre los diferentes grupos étnicos y socio-económicos ([Schiller, 1996], [Wresch, 1996]).

Por otro lado, esta pujanza creciente de la tecnología robótica recomienda formar profesionales en este sector, que incluso lleven más allá las fronteras actuales y ayuden a crear nuevas aplicaciones robóticas que sirvan a las personas (Figura A.8-a,b) y al progreso de la humanidad (Figura A.8-c). Actualmente la formación específica en Robótica aparece ligeramente en educación secundaria y se realiza fundamentalmente en la universidad, con titulaciones de grado y postgrados específicos.

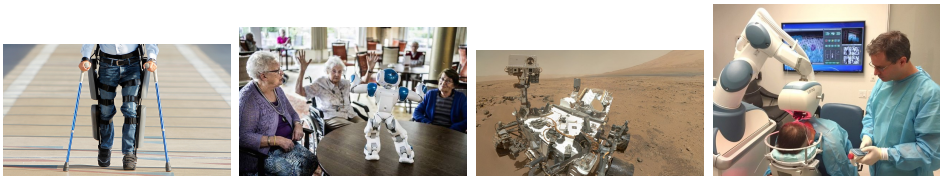


Figura A.8: La Robótica como profesión para la sociedad

No obstante, es un hecho que la Robótica está cobrando una importancia ascendente también en la educación preuniversitaria tanto en España como en otros países occidentales, bien como campo de conocimiento en sí mismo, bien como herramienta para exponer de modo atractivo la tecnología y otras materias a los niños. Además, la Robótica tiene poder de motivación en los estudiantes y eso permite acercar la tecnología a los niños y niñas ([Rodger and Walker, 1996]) usando la robótica como herramienta para exponerles a conceptos básicos de ciencias ([Altin and Pedaste,

2013]), tecnología, ingeniería y matemáticas (STEM *Science, Technology, Engineering and Math*) ([Mubin et al., 2013]). Los alumnos aprenden, casi mediante un juego (Figura A.7), nociones complejas difíciles de explicar o asimilar mediante la clásica clase magistral ([Cerezo and Sastrón, 2015, Jiménez et al., 2010]).

Por ejemplo, en docencia oficial la Comunidad de Madrid (Decreto 48/2015, de 14 de mayo, del Consejo de Gobierno) introdujo el curso 2015-16 la asignatura *Tecnología, programación y robótica* en el currículum oficial de Educación Secundaria Obligatoria. Además hay una demanda creciente de robótica para actividades extraescolares que está creando un ecosistema de empresas que la satisfacen. Fuera de España la implantación de la Robótica en educación es un hecho. En los E.E.U.U. seis estados (Iowa, Nevada, Wisconsin, Washington, Idaho y Utah) han anunciado planes e inversiones con este objetivo en los últimos cinco meses. Asimismo, recientemente cuatro países: Canadá, Irlanda, Nueva Zelanda, y Rumanía han anunciado planes en el mismo sentido, con una inversión que en total alcanza los 300 millones de dólares. Japón en su informe `New Robot Strategy` ([Japan-Economic, 2015]) dejaba claro que invertir en la Robótica es fundamental para el crecimiento del país.

En este crecimiento confluyen tanto la enseñanza de robótica en sí misma como la enseñanza de otras disciplinas (p.e. programación) utilizando la robótica como herramienta vehicular ([Magnenat et al., 2014, Merkouris et al., 2017, Kubilinskiene et al., 2017]).

Figura A.9: Campeonato robótico RoboCampeones

Otra muestra de la importancia creciente de la robótica en la educación son los campeonatos robóticos para adolescentes, que motivan el interés por la tecnología. Por ejemplo, el campeonato robótico RoboCampeones (Figura A.9) de la Comunidad de Madrid, que reunió a más de 2000 estudiantes en la última edición, con pruebas como el sigue líneas, el sumo entre robots, etc. Igualmente, a nivel internacional también se organizan numerosos campeonatos para reunir a estudiantes de todo el mundo a que aprendan, compartan experiencia y se diviertan mediante el desarrollo de artefactos robóticos. Cabría destacar la RoboCup Junior[2] ([Eguchi, 2016, Navarrete et al., 2016, Kandlhofer and Steinbauer, 2014]), con pruebas como el rescate o el fútbol robótico. También la First LEGO League (FLL) o las *VEX Robotics Competitions*[3]. En Finlandia el campeonato por excelencia, que además concentra a estudiantes de toda Europa ([Jormanainen and Korhonen, 2010]) y tiene acuerdos con centros de Sudáfrica ([Graven and Stott, 2011]), es el SciFest[4].

También en la comunidad académica e investigadora han aparecido un conjunto de congresos y conferencias que enfatizan el papel de la Robótica en la Enseñanza. Por ejemplo *Conference on Robotics in Education* (RiE), *Workshop on Teaching Robotics with ROS* (TRROS) dentro del *European Robotics Forum*[5]. También son frecuentes números especiales sobre educación en robótica en varias revistas científicas. Finalmente, cabe destacar algunas revistas de alto impacto que versan sobre Robótica y Educación: *Computers and Education*, *British Journal of Educational Technology*, *International Journal of Robotics Research*, *Journal of the Learning Sciences* o *Journal of Research in Science Teaching*.

A pesar esta pujanza creciente en robótica educativa, existen en el mercado pocos kits educativos que proporcionen un entorno docente útil y motivador para estudiantes preuniversitarios. Los kits existentes están enfocados a ser usados puntualmente en clase, con un bajo nivel de complejidad, lo que resulta en una *vida útil* corta y una escasa motivación de los alumnos, que alcanzan los desafíos planteados en unas pocas sesiones de

[2] `http://rcj.robocup.org`
[3] `https://www.vexrobotics.com/vexedr/competition`
[4] `http://www.scifest.fi`
[5] `http://www.eu-robotics.net/robotics_forum`

trabajo. Tampoco existe ningún sistema, aún menos guiado, que mantenga de forma constante la motivación y el nivel de desafío, ni donde la visión juegue un papel importante.

A.4 Propuesta

Después de haber descrito en la Sección A.1 el papel que tiene la Robótica en la sociedad de hoy y de cara a los próximos años y tras fundamentar en la Sección A.3 la consecuente importancia que juega la educación como medio por el cual las nuevas generaciones se han de preparar para el futuro a corto y medio plazo, se ha identificado un enorme salto entre el nivel de la formación académica que se imparte a nivel universitario en las carreras del ámbito científico-tecnológico y el currículum oficial implantado en los niveles preuniversitarios, concretamente en las asignaturas de ciencias de los cursos de Educación Secundaria.

Por un lado, se propone con este trabajo adquirir un profundo bagaje, para su posterior aplicación pedagógica sobre los principales problemas inherentes a la Robótica, como son la navegación y localización, empleando para ello la visión como sensor principal; así como el desarrollo de técnicas de robots con visión, como el control visual, la memoria visual o un sistema atentivo visual.

Por otro lado, para minimizar ese salto académico detectado entre la formación preuniversitaria y la ofrecida por las universidades se propone desarrollar un completo entorno docente de Robótica con visión inexistente en la actualidad que integre: (a) una adecuada metodología pedagógica acorde al nuevo perfil de *Alumnado de la Era Digital*, donde cada alumno es más que nunca un individuo con sus propias inquietudes ([Rose and Meyer, 2002, Selber, 2004]) dentro del enorme universo de posibilidades que brinda esta Era de la Información ([Solove, 2004]); (b) una infraestructura software que sea sencilla e intuitiva de manejar por los jóvenes estudiantes pero a la vez potente y versátil, incorporando las suficientes librerías de apoyo como para realizar las suficientes prácticas, en número y en complejidad, de programación de robots con visión, como para motivar continuamente a los alumnos ([Benitti, 2012]), así como diversos ejemplos

que sirvan de muestra; (c) un experimentado programa académico que pueda ser seguido durante un curso completo y que incluya las sesiones suficientes y debidamente escalonadas para la correcta asimilación por los estudiantes ([Ainley et al., 2008]); (d) una plataforma hardware versátil, económicamente asequible por los centros de educación secundaria para satisfacer las necesidades de una clase completa, pero a la vez estandarizada y potente, que permita la ejecución de algoritmos de Robótica con visión.

En la actualidad no existe nada más allá que kits educativos principalmente enfocados para ser usados puntualmente en clase, siguiendo de este modo un enfoque que consideramos obsoleto, y que en un corto intervalo de tiempo resulta poco motivante para los estudiantes. De hecho, la mayoría de estos kits existentes en el mercado están diseñados para que los alumnos más jóvenes despierten su interés por la Robótica, pero no para que los alumnos que están en los cursos que preceden a la formación universitaria adquieran una correcta y completa formación en programación, algo tan demandado y extendido hoy día en casi cualquier carrera. Si bien es cierto que existen otros kits más especializados en determinadas áreas de la Ciencia ([Schweikardt and Gross, 2006]), el entorno propuesto va más allá y ofrece todas las herramientas necesarias tanto para el alumnado como el profesorado ([Bers et al., 2002]) para desarrollar de forma versátil un curso académico completo poniendo a su disposición numerosos y sofisticados algoritmos, incluyendo visión, con un interfaz ameno e intuitivo.

A.5 Objetivos

El objetivo principal de este trabajo es el desarrollo, puesta en práctica y evaluación de un completo entorno docente de Robótica. Este entorno disminuirá la gran brecha detectada por este autor durante los años de experiencia en docencia de las asignaturas científico-tecnológicas en los niveles académicos preuniversitarios y los curricula de las carreras técnicas. De este modo, las nuevas generaciones podrán estar mejor formadas en las habilidades demandadas por la sociedad actual y de los próximos años.

El primer objetivo es desarrollar los algoritmos que den solución a los

principales problemas existentes dentro del marco de los Robots con visión y que abordarán cuestiones como la navegación y la localización usando como sensor principal la visión. Esto supondrá a su vez determinar e implementar los problemas inherentes al tratamiento y selección de información útil de una imagen, así como la generación de una compleja memoria visual que vaya incorporando los elementos que se detecten en la escena circundante al robot, resultando finalmente en un comportamiento por parte de éste de mayor inteligencia, que es lo que se demandará en la Robótica de los próximos años.

El segundo objetivo es integrar lo anterior en un completo entorno educativo que incluirá todo un programa académico a seguir durante la docencia de asignaturas como Tecnología y TICs en cursos preuniversitarios. Para ello desarrollaremos una infraestructura software que facilite a estos alumnos la programación de robots con visión, de forma que ésta resulte sencilla e intuitiva para desarrollar soluciones a los problemas clásicos de robots: navegación, visión, etc.

Por otro lado, las plataformas robóticas físicas a considerar serán aquellas que tengan un bajo coste, para que resulte viable su adquisición por los centros de Educación Secundaria para un aula completa. Es por ello que los desarrollos se basarán en plataformas existentes en el mercado, estandarizadas y de bajo coste. Además, y para satisfacer los anteriores objetivos de poner en práctica algoritmos robóticos y de visión de robots, se desarrollará una plataforma robótica física que permita computacionalmente la ejecución de estos así como el manejo de una cámara como sensor principal.

Este entorno educativo se pretende implantar en diferentes centros educativos de la Comunidad de Madrid, donde la Robótica está oficialmente integrada en el currículum de Educación Secundaria durante más de un curso académico. Se investigarán las diferentes metodologías pedagógicas existentes en la literatura para analizar cuál puede ser la idónea para la puesta en práctica del proceso de Enseñanza-Aprendizaje de la Robótica y que, por tanto, será la elegida para llevar a cabo las sesiones que conformen el programa educativo que desarrollemos.

Por último, se validará el entorno docente propuesto con estudiantes

reales, se evaluarán los resultados académicos y de satisfacción por parte del alumnado y los profesores que empleen el entorno educativo de esta tesis frente a aquellos grupos que sigan los programas tradicionales de las asignaturas de Tecnología y TICs. Analizaremos los resultados de esta evaluación y que serán obtenidos mediante entrevistas diarias, calificaciones académicas obtenidas por los alumnos, así como encuestas sobre el grado de satisfacción por ambas partes: alumnos y profesores.

A.6 Estructura de la tesis

Este documento de tesis tiene la siguiente estructura:

Se da comienzo con el presente Capítulo A en el cual se ha introducido el contexto y la motivación de la tesis. En el Capítulo 2 se describe el estado del arte de la Robótica con visión con sus distintos frentes principales de investigación. También se estudia el estado del arte de Robótica educativa.

El Capítulo 3 describe los problemas inherentes a la navegación y localización de robots usando visión, así como los distintos desarrollos software que hemos aportado para dar solución a los mismos con mecanismos de atención selectiva y una memoria visual de corto plazo. Además, en este capítulo se exponen de forma detallada los procedimientos matemáticos que hemos empleado para integrar una cámara convencional como un potente sensor visual.

El Capítulo 4 está dedicado a describir el diseño y el desarrollo que hemos llevado a cabo durante estos años de la infraestructura docente hardware y software creada en esta tesis, siguiendo el currículum académico elegido. Asimismo se hace un recorrido por las distintas corrientes pedagógicas para, a continuación, describir la metodología en la que nos hemos centrado. También se detallan los resultados de validación del entorno después de su implantación con más de 2000 estudiantes.

Por último, en el Capítulo 5 se detallan las conclusiones extraídas de este trabajo. Todo ello acompañado por las publicaciones derivadas de esta disertación. Se cierra el capítulo con las posibles líneas futuras a seguir.

A.7 Conclusiones

Esta investigación está focalizada en incorporar la Robótica y los robots con visión en el aula para formar a los estudiantes preuniversitarios, satisfaciendo las demandas que impone la *Sociedad de la Era Digital* y las necesidades de motivación detectadas en los alumnos, que todavía estudian en un sistema de formación aún por adaptar a esta denominada *Revolución Industrial 4.0*.

Aunque existen en el mercado numerosos kits educativos de Robótica, la mayoría de estos están enfocados a los alumnos más jóvenes. Generalmente se basan en construir de *sus* plataformas robóticas con *sus* propios entornos de programación, lejos de emplear lenguajes de programación más estandarizados. Además, normalmente tienen un nivel de complejidad no muy elevado, lo que conlleva a que estas herramientas suelen resultar —a corto plazo— en una escasa motivación por parte del alumnado. Por otro lado, dada la complejidad que supone el tratamiento de un sensor como la cámara, a pesar de su gran versatilidad no suele ser incluido en estos entornos educativos.

En base a esto, y en primer lugar, se han implementado diversos algoritmos que emplean una cámara como sensor principal para dar solución a los problemas fundamentales de la Robótica, como son la navegación y la localización. Ambos frentes han sido resueltos mediante el desarrollo de una memoria visual en la que el robot es capaz de incorporar y abstraer elementos complejos siguiendo diferentes patrones: flechas, paralelogramos o caras humanas. Sobre esta memoria visual se monta un sistema atentivo capaz de atender los distintos elementos que circundan al robot. Así se ha conseguido un sistema autónomo e inteligente, que navega por un entorno cambiante y real, en el cual se localiza constantemente. Este sistema de percepción visual ha sido validado tanto en robots reales como en simulados. La memoria representa muy bien el entorno del robot utilizando las imágenes de la cámara móvil, cuyo movimiento es controlado por el mecanismo de atención. La memoria es dinámica pero tiene cierta persistencia para tratar las oclusiones temporales. La localización funciona en tiempo real, proporcionando errores de posición por debajo de 15 cm y 5 grados y

es lo suficientemente robusta para recuperarse de los secuestros o errores de estimación en entornos simétricos.

En segundo lugar, tras investigar la situación en el mercado de los kits educativos de Robótica existentes y analizar profundamente qué depara el futuro a corto y medio plazo en cuanto a demandas del mercado laboral se refiere, el autor, como experimentado docente de Educación Secundaria, detecta una deficiencia en el proceso de enseñanza-aprendizaje de Robótica en el nivel curricular preuniversitario. Por ello, se ha desarrollado un completo entorno educativo que incluye:

- Plataforma robótica basada en la placa controladora de hardware libre Raspberry Pi 3. Esta plataforma ha sido elegida por varios motivos: bajo coste, potencia, versatilidad, estandarización e inclusión de una cámara con su propio bus de datos, la PiCamera. Así se ha construido un robot totalmente funcional, el `PiBot`, al que —gracias a los puertos GPIO de la placa— se le han conectado diversos sensores y actuadores, además de su propia cámara.

- Infraestructura software desarrollada en lenguaje Python, JdeRobot-Kids, que ha facilitado al alumnado la programación del robot, con funciones sencillas e intuitivas para manejar los distintos sensores y actuadores, pero a la vez de gran potencial, como las correspondientes al manejo una cámara como sensor.

- Amplio repertorio de prácticas que han servido de apoyo a los alumnos para su progresión en el aprendizaje de la programación de robots con visión.

- Programa académico que incluye el plan de actividades para la asignatura *Programación, Robótica y Tecnología* en diferentes cursos de Educación Secundaria y para un curso de actividades extraescolares. Este programa se ha dividido en cuatro fases, partiendo de un nivel muy básico hasta el desarrollo de un proyecto que de solución a alguna tarea clásica de Robótica, como por ejemplo un robot sigue-líneas o choca-gira usando la cámara como sensor.

El entorno docente `JdeRobot-Kids` se ha usado en la Fundación Franciscanas de Montpellier durante los cursos 2016/2017 y 2017/2018 en la asignatura *Programación, Robótica y Tecnología* de 1.º, 2.º y 4.º de E.S.O. y en asignaturas extraescolares anuales del Colegio Ntra. Sra. Sagrado Corazón de Madrid. También en asignaturas extraescolares del Colegio Villa de Móstoles en 2016/2017 y en el Colegio Rihondo de Alcorcón en 2017/2018.

En total, el entorno se ha seguido por unos 2.050 alumnos y una decena de profesores en los dos últimos cursos académicos. Se ha medido su impacto a través de encuestas y los resultados han sido muy satisfactorios (4.5). Muestran una gran acogida y satisfacción entre los estudiantes y los profesores. Además, los proyectos robóticos realizados por los alumnos demuestran un alto nivel de asimilación de conceptos, a la par que la dinámica de las clases ha resultado muy amena.

A.8 Contribuciones

De este trabajo derivan, además de las publicaciones enumeradas en la Sección A.9, numerosos desarrollos software y hardware relacionados con la Robótica con visión y la educación en Robótica. A continuación se enumeran los principales:

(C_1) *Sistema visual atentivo*[6] sobre cuello mecánico, lo que ha permitido disponer en los robots de un sistema perceptivo con un amplio campo de visión de toda la escena circundante al robot, mayor que el campo visual instantáneo de una cámara. Además, este sistema es fundamental para indicar al robot cuál es la siguiente zona a la que dirigir la mirada, además de centrarla sobre el objeto al que se está prestando atención en un determinado momento, y seguirlo.

(C_2) *Memoria visual*[7], que es la responsable de mantener información sobre los distintos objetos de interés que el robot va encontrando por el entorno por el que navega, lo que le permite tomar decisiones más in-

[6] `http://jderobot.org/VisualSonar`
[7] `http://jderobot.org/VisualMemory`

teligentes en tiempo real sobre las circunstancias que le rodean. Incluye varios desarrollos principales: (a) el detector de objetos, encargado de identificar formas básicas (conceptos) como flechas, paralelogramos y caras humanas; (b) el mecanismo de predicción de elementos, que permite al sistema predecir elementos ya memorizados con anterioridad, aliviando el coste computacional; y (c) el algoritmo de generación de hipótesis perceptivas, responsable de abstraer objetos complejos, que ha permitido al sistema perceptivo solventar con éxito las posibles oclusiones que se pueden dar en un entorno real.

(C_3) *Librería auxiliar de visión*[8], que incluye toda la funcionalidad para satisfacer los distintos cambios de sistemas de coordenadas entre los objetos detectados del mundo real y los píxeles correspondientes a la imagen 2D que vierte una cámara. El desarrollo de una clase para abstraer completamente el modelo de una cámara ideal Pin-Hole, ha permitido a los estudiantes que se inician en los robots con visión realizar complejas prácticas de visión sin necesidad de conocimientos avanzados en la materia. Incluye además los algoritmos basados en la Hipótesis suelo e implementación propia de extractor de líneas de Solis, que han permitido estimar distancias a los objetos usando una única cámara.

(C_4) *Diseño y construcción de PiBot*[9][10] como plataforma robótica basada en hardware libre y estándar, concretamente en la placa Raspberry Pi 3. La potencia, versatilidad y la inclusión de la cámara PiCam con bus dedicado han permitido el desarrollo de algoritmos de visión; por otro lado, el bajo coste de esta plataforma permite que los centros educativos puedan adquirir un número considerable de prototipos para poder satisfacer en número a un aula completa. La inclusión de potentes servos dan lugar a una plataforma robótica móvil robusta, fiable y ágil para navegar en un entorno real.

[8] `http://jderobot.org/RobotVision`

[9] `http://jderobot.org/JulioVega_PhD#2018.03.03._Introducing_the_new_PiBot_v3.0`

[10] `https://github.com/JdeRobot/JdeRobot/tree/master/assets/gazebo/models/pibot`

(C_5) *Repertorio de prácticas en Python sobre mBot y Arduino estándar.* Ha permitido que alumnos y profesores puedan disponer de una extensa batería de ejemplos totalmente funcionales, e implementados en un lenguaje real como Python, de todos los sensores y actuadores que pueden acoplarse a una placa Arduino UNO así como los que ya tiene incluido de serie el mBot. Esto ha facilitado enormemente la inclusión de la Robótica en el aula de centros de Educación Secundaria.

(C_6) *Infraestructura software para dar soporte al PiBot*[11], implementada en lenguaje Python. Ha permitido la inclusión de la potente plataforma robótica PiBot en el aula, y que pueda ser utilizada por los jóvenes estudiantes que carecen de los conocimientos necesarios para manipular un robot de ese calibre. Esta infraestructura facilita el manejo completo del robot: servos, sensores (ultrasonidos, infrarrojos) y actuadores variados (pulsadores, leds) y, lo más importante, una cámara real, como la PiCam o cualquier WebCam a las que se ha dado soporte mediante el driver `piCamServer`[12], lo que ha permitido la programación por parte de los alumnos de numerosas y desafiantes prácticas, empleando de forma sencilla una cámara como sensor principal.

(C_7) *Repertorio de prácticas en Python sobre PiBot.* Prácticas similares a las del mBot pero adaptadas a esta otra plataforma y extendida con varios ejercicios que requieren usar la cámara a bordo.

(C_8) *Entorno educativo completo*[13]. Con todo lo anterior, se ha diseñado y seguido un programa educativo completo, con un programa académico de complejidad adecuada para estudiantes de cursos preuniversitarios. Se han detallado y organizado todas las sesiones seguidas, con sus contenidos y objetivos, así como la metodología que se ha analizado ser la más adecuada para el proceso de Enseñanza-Aprendizaje

[11] `https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/PiBot`

[12] `https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/piCamServer_py`

[13] `http://kids.jderobot.org`

de la Robótica, en base a la experiencia de estos años en centros españoles y la estancia en centros finlandeses.

En total se han implementado más de 200.000 líneas de código entre los distintos desarrollos software descritos. La distribución de las mismas se refleja en la figura A.10.

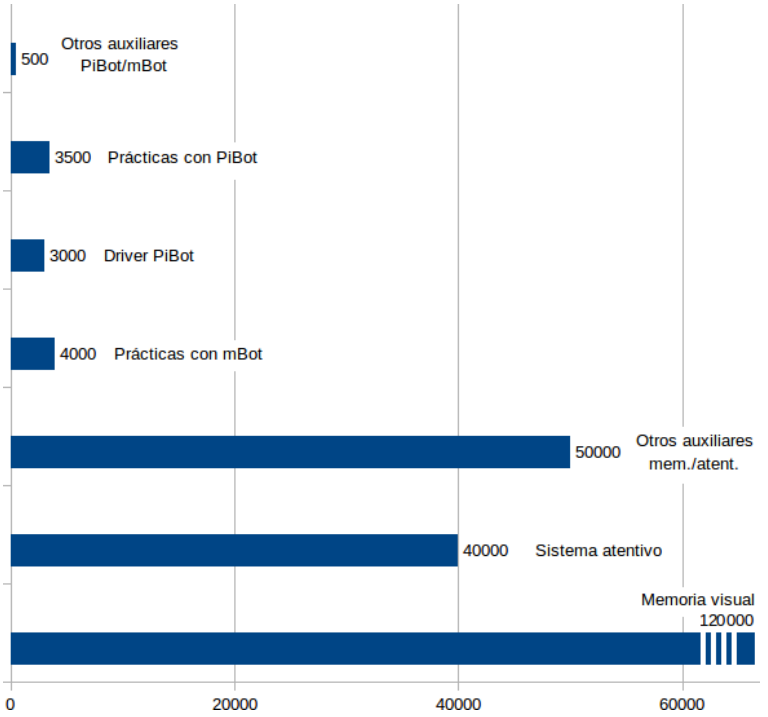


Figura A.10: Distribución de las líneas de código implementadas

El progreso de todo el trabajo realizado para esta disertación se puede seguir en la WiKi[14] dedicada a tal efecto, donde además se puede acceder a todos los códigos implementados. Estos trabajos han sido integrados en la plataforma JdeRobot[15] del Grupo de Robótica de la Universidad Rey Juan Carlos bajo el marco de distintos grandes proyectos: GuideRobot[16],

[14] `http://jderobot.org/JulioVega_PhD`
[15] `https://jderobot.org`
[16] `http://jderobot.org/GuideRobot`

RobotVision[17], VisualMemory[18], VisualSonar[19] y JdeRobot-Kids[20].

Los aportes software incluyen el uso de distintos lenguajes como `C`, `C++` y `Python`, así como numerosas librerías y herramientas entre las que cabe destacar las siguientes: `Susan`, `XForms`, `Glade`, `GTK`, `FreeGLUT`, `OpenGL`, `OpenCV`, `v4l`, `GSL`, `ICE`, `Math`, `PiGPIO`, `PyGame`, `IMUtils`, `NumPy`, `SVN`, `GIT`, `UMLet`, etc.

A.9 Publicaciones

Las publicaciones que derivan de este trabajo son las siguientes:

- Artículos:
  - Julio Vega, José María Cañas, Fran Pérez y Aitor Martínez. (2018). *JdeRobot-Kids: entorno docente de robótica para niños*. En revisión. **Revista Iberoamericana de Automática e Informática Industrial, RIAI**. Factor de Impacto (JCR 2016): 0,500.
  - Julio Vega, Eduardo Perdices y José María Cañas. (2013). *Robot evolutionary localization based on attentive visual short-term memory*, pages 1268-1299. **Sensors**. ISSN: 1424-8220. Factor de Impacto (JCR 2013): 2,048.
  - Julio Vega, José María Cañas y Eduardo Perdices. (2012). *Local robot navigation based on an active visual short-term memory*, pages 21-30. **Journal of Physical Agents**. ISSN: 1888-0258. Factor de Impacto (SJR 2012): 0,171.
- Capítulos de libro:
  - Julio Vega, Eduardo Perdices y José María Cañas. (2012). *Attentive Visual Memory for Robot Localization*, pages 408-438.

[17] `http://jderobot.org/RobotVision`
[18] `http://jderobot.org/VisualMemory`
[19] `http://jderobot.org/VisualSonar`
[20] `http://kids.jderobot.org`

Robotic Vision: Technologies for Machine Learning and Vision Applications. Ed. IGI-GLOBAL. ISBN: 978-84-694-6730-5.

- Congresos:
  - Julio Vega y José María Cañas. (2016). *Entorno docente con Arduino y Python para Educación Robótica en Secundaria*. **JITICE 5th Workshop, Educational Innovation and ICT**. ISBN: 978-84-697-0892-7.
  - Julio Vega y José María Cañas. (2014). *Curso de Robótica en Educación Secundaria usando Constructivismo Pedagógico*. **JITICE 4th Workshop, Educational Innovation and ICT**. ISSN: 2172-6620.
  - José María Cañas, Laura Martín y Julio Vega. (2014). *Innovating in robotics education with Gazebo simulator and JdeRobot framework*. **CUIEET XXII Congreso Universitario de Innovación Educativa en Enseñanzas Técnicas**. ISSN: 2172-6620.
  - Borja Menéndez, José María Cañas, Eduardo Perdices y Julio Vega. (2013). *Programming a Humanoid Social Robot Using the JdeRobot Framework*. **RoboCity2030 11th Workshop, Robots sociales**. ISBN:978-84-695-7212-2.
  - Julio Vega, Eduardo Perdices y José María Cañas. (2012). *Robot evolutionary localization based on attentive visual short term memory*. **IEEE Intelligent Vehicles Symposium Workshops, Perception in Robotics**. ISBN: 978-84-695-3472-4.
  - Julio Vega y José María Cañas. (2011). *Attentive visual memory for robot navigation*. **WAF2011 XII Physical Agents**

Workshop. ISBN: 978-84-694-6730-5.

– Eduardo Perdices, José María Cañas, Julio Vega, Carlos Agüero y Francisco Martín. (2010). *Localización visual de robots en la RoboCup mediante algoritmos evolutivos*. **Robocity 2030**. ISBN: 84-693-6777-3.

– Julio Vega, José María Cañas, Pablo Miangolarra y Eduardo Perdices. (2010). *Memoria visual atentiva basada en conceptos para un robot móvil*. **Robocity 2030**. ISBN: 84-693-6777-3.

– Julio Vega y José María Cañas. (2009). *Sistema de atención visual para la interacción persona-robot*. **RoboCity2030, Interacción persona-robot**. ISBN: 978-84-692-5987-0.

A.10 Líneas futuras

Se espera que el entorno docente desarrollado, `JdeRobot-Kids`, y la metodología seguida contribuyan a largo plazo a mejorar los resultados en los indicadores educativos de España, a reducir la brecha de la calidad educativa con otros países como Finlandia.

La principales líneas de trabajo a corto plazo son las siguientes: *(a)* unificar el interfaz de programación para el PiBot real y el existente para el PiBot simulado, de modo que las prácticas de los alumnos puedan ejecutarse sin modificaciones en el robot real o el simulado; *(b)* desarrollar nuevas prácticas con visión como la detección y seguimiento de caras de personas, y materializar en el PiBot un sistema atentivo visual y una memoria visual; y *(c)* difundir el entorno desarrollado.

Respecto a este último punto, durante el mes de julio del presente año se imparte un taller de Robótica en el Campus de Fuenlabrada de la Universidad Rey Juan Carlos. Por otro lado, se está elaborando un manual para que sirva de libro de texto del profesor, con toda la parte de conceptos teóricos de Robótica y Electrónica, la instalación del entorno y todo el repertorio de prácticas desarrollado para las distintas plataformas hardware:

Arduino, mBot y PiBot. También se pretende publicar un artículo en alguna revista de alto impacto con todos estos avances, y fundamentalmente con la novedosa plataforma desarrollada, `PiBot`.

De cara al próximo curso se pretende seguir empleando este entorno docente en los centros que ya se está usando; además se extendrá a otros centros como el Instituto Carpe Diem de Fuenlabrada y muchos otros donde se seguirá el entorno en actividades extraescolares gracias a la colaboración con la empresa Logix5[21].

También se está trabajando en la extensión del entorno a un Web-IDE de `JdeRobot-Kids`, lo que permitirá a los estudiantes trabajar en la nube, sin necesidad de que el alumno se instale ninguna librería en su equipo de trabajo, facilitando enormemente el trabajo en el aula. Esto facilitará las prácticas de programación con la plataforma PiBot real, mediante la descarga en la Raspberry del código desarrollado en el IDE de la nube; y con el simulado, gracias al simulador incorporado en el entorno web.

Y, por último, se está desarrollando el proyecto `PyOnArduino`[22], cuyo objetivo es que las prácticas desarrolladas en Python para mBot utilicen una biblioteca en Python que se ejecuta en el ordenador personal y que está en continua comunicación (alámbrica o inalámbrica) con el mBot. Esto es, no ejecutan directamente sobre el procesador Arduino del mBot, sino en el PC. A bordo del mBot se ejecuta un trasegador de datos que continuamente captura los sensores y se los envía a la biblioteca Python en el PC; a su vez recibe del PC comandos para los actuadores y se los ordena a los motores a bordo. Una posible continuación de esta tesis es desarrollar un traductor de Python a lenguaje Arduino, de modo que las aplicaciones de los estudiantes escritas en Python se puedan traducir a lenguaje Arduino y de este modo cargar a bordo para ejecutar realmente dentro del mBot.

[21] `http://www.logix5.com/roboticaeducativa`
[22] `https://github.com/JdeRobot/PyOnArduino`

Bibliography

[Acosta-Nassar, 2014] Acosta-Nassar, C. A. (2014). El uso de una estrategia híbrida entre aprendizaje basado en problemas y clases magistrales para mejorar aprendizajes. *Educare Electronic Journal*. 34

[Afari and Khine, 2017] Afari, E. and Khine, M. (2017). Robotics as an educational tool: Impact of LEGO mindstorms. *IJIET*, 7(6):437–442. 28

[Ainley et al., 2008] Ainley, J., Enger, L., and Searle, D. (2008). *Students in a Digital Age: Implications of ICT for Teaching and Learning*, pages 63–80. Springer US, Boston, MA. 12, 145

[Altin and Pedaste, 2013] Altin, H. and Pedaste, M. (2013). Learning approaches to applying robotics in science education. *Journal of baltic science education*, 12(3):365–377. 10, 142

[Antequera et al., 2017] Antequera, M. L., Petkov, N., and Jimenez, J. G. (2017). City scale continuous visual localization. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6. 6, 139

[Araujo et al., 2015] Araujo, A., Portugal, D., Couceiro, M. S., and Rocha, R. P. (2015). Integrating arduino-based educational mobile robots in ros. *J Intell Robot Syst (2015) 77:281–298*. 28

[Arbel and Ferrie, 2001] Arbel, T. and Ferrie, F. (2001). Entropy-based gaze planning. *Image and Vision Computing, vol. 19, no. 11, pp. 779-786*. 4, 7, 20, 137, 140

[Atanasov et al., 2015] Atanasov, N., Ny, J. L., Daniilidis, K., and Pappas, G. J. (2015). Decentralized active information acquisition: Theory and application to multi-robot slam. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 25

[Badal et al., 1994] Badal, S., Ravela, S., Draper, B., and Hanson, A. (1994). A practical obstacle detection and avoidance system. *Proc. of 2nd IEEE Workshop on Applications of Computer Vision, pages 97–104*. 22

[Bain, 2007] Bain, K. (2007). *Lo que hacen los mejores profesores universitarios*. 33

[Bajcsy, 2009] Bajcsy, R. (2009). Active perception. *Proc. of the IEEE 76, pp. 996-1005*. 7, 17, 140

[Balachandran, 2009] Balachandran, S. (2009). General purpose input output (gpio). technical report, ece 480 design team 3. Available in the College of Engineering, Michigan State University website. 92

[Ballard, 1991] Ballard, D. H. (1991). Animate vision. *Artificial Intelligence 48, pp. 57-86*. 19

[Balogh, 2010] Balogh, R. (2010). Educational robotic platform based on arduino. In *Proceedings of the 1st international conference on Robotics in Education, RiE2010. FEI STU, Slovakia*, pages 119–122. 28

[Barrera et al., 2005] Barrera, P., Cañas, J., and Matellán, V. (2005). Visual object tracking in 3d with color based particle filter. In *Int. Journal of Information Technology*. 6, 139

[Bauer, 2015] Bauer, R. (2015). *Computational Modeling of Visual Attention*. 20

[Benitti, 2012] Benitti, F. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers and Education*, 58(3):978 – 988. 12, 145

[Bers et al., 2002] Bers, M. U., Ponte, I., Juelich, C., Viera, A., and Schenker, J. (2002). Teachers as designers: Integrating robotics in early childhood education. *Information Technology in Childhood Education Annual*, 2002(1):123–145. 13, 146

[Beyers and van der Merwe, 2017] Beyers, R. N. and van der Merwe, L. (2017). Initiating a pipeline for the computer industry: Using Scratch and LEGO robotics. In *Information Communication Technology and Society (ICTAS), Conference on*, pages 1–7. IEEE. 28, 31

[Biswas, 2016] Biswas, P. (2016). *Exploring the use of eye gaze controlled interfaces in automotive environments*. 17

[Blosch et al., 2010] Blosch, M., Weiss, S., Scaramuzza, D., and Siegwart, R. (2010). Vision based mav navigation in unknown and unstructured environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 21–28. 24

[Borji and Itti, 2013] Borji, A. and Itti, L. (2013). State-of-the-art in visual attention modeling. 21

[Borji et al., 2013] Borji, A., Sihite, D., and Itti, L. (2013). What stands out in a scene? a study of human explicit saliency judgment. *Vision Research*. 19

[Borji et al., 2014] Borji, A., Sihite, D. N., and Itti, L. (2014). What/where to look next? modeling top-down visual attention in complex interactive environments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(5):523–538. 19

[Burgard and Fox, 1997] Burgard, W. and Fox, D. (1997). Active mobile robot localization by entropy minimization. *Proc. of the Euromicro Workshop on Advanced Mobile Robots, Los Alamitos, CA, pp. 155-162.* 24

[Bylinskii et al., 2015] Bylinskii, Z., DeGennaro, E., Rajalingham, R., Ruda, H., Zhang, J., and Tsotsos, J. (2015). Towards the quantitative evaluation of visual attention models. *Vision Research*, 116:258 – 268. Computational Models of Visual Attention. 21

[Cañas et al., 2014] Cañas, J., Martín, L., and Vega, J. (2014). Innovating in robotics education with gazebo simulator and jderobot framework. In *XXII Congreso Universitario de Innovación Educativa en Enseñanzas Técnicas, CUIEET-2014 Vol II, pp 1483-1496, Almadén, September 17-19th 2014. ISBN: 978-84-9044-108-4.* 88

[Cañas et al., 2008] Cañas, J., Martínez de la Casa, M., and González, T. (2008). Overt visual attention inside jde control architecture. *International Journal of Intelligent Computing in Medical Sciences and Image Procesing. Volume 2, Number 2, pp 93-100, ISSN: 1931-308X. TSI Press, USA.* 19

[Caceres et al., 2017] Caceres, C., Rosario, J. M., and Amaya, D. (2017). Approach of kinematic control for a nonholonomic wheeled robot using artificial neural networks and genetic algorithms. In *2017 International Conference and Workshop on Bioinspired Intelligence (IWOBI)*, pages 1–6. 6, 138

[Cerezo and Sastrón, 2015] Cerezo, F. and Sastrón, F. (2015). Laboratorios virtuales y docencia de la automática en la formación tecnológica de base de alumnos preuniversitarios. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 12(4):419–431. 10, 142

[Chaudhary et al., 2016] Chaudhary, V., Agrawal, V., Sureka, P., and Sureka, A. (2016). An experience report on teaching programming and computational thinking to elementary level children using lego robotics education kit. In *Technology for Education (T4E), 2016 IEEE Eighth International Conference on*, pages 38–41. IEEE. 28

[Chen and Birchfield, 2009] Chen, Z. and Birchfield, S. T. (2009). Qualitative vision-based path following. *IEEE Transactions on Robotics*, pages 749–754. 23

[Cho et al., 2014] Cho, H., Seo, Y. W., Kumar, B. V. K. V., and Rajkumar, R. R. (2014). A multi sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE*

International Conference on Robotics and Automation (ICRA), pages 1836–1843. 6, 139

[Clark and Stark, 1975] Clark, M. and Stark, L. (1975). Time optimal behavior of human saccadic eye movement. *IEEE Transactions on Automatic Control*, 20(3):345–348. 17

[Dellaert et al., 1999] Dellaert, F., Burgard, W., Fox, D., and Thrun, S. (1999). Using the CONDENSATION algorithm for robust, vision-based mobile robot localization. In *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR-99*, volume 2, pages 588–594, Fort Collins, Colorado (USA). IEEE Computer Society. 24

[Deloitte, 2015] Deloitte (2015). *From brawn to brains: The impact of technology on jobs in the UK*. 8, 140

[Demetriou, 2011] Demetriou, G. A. (2011). Mobile robotics in education and research. In *Mobile Robots-Current Trends*. InTech. 28

[Dewey, 2007] Dewey, J. (2007). *Experience and education*. 33

[Dimitrov et al., 2015] Dimitrov, V., Wills, M., and Padir, T. (2015). Realization of vision based navigation and object recognition algorithms for the sample return challenge. In *2015 IEEE Aerospace Conference*. 23

[Duckett, 2003] Duckett, T. (2003). A genetic algorithm for simultaneous localization and mapping. In *In proceedings of the IEEE international conference on robotics and automation (ICRA'2003)*. 25

[Eguchi, 2012] Eguchi, A. (2012). *Educational Robotics Theories and Practice*, pages 1–30. IGI Global. 26

[Eguchi, 2016] Eguchi, A. (2016). RoboCupJunior for promoting STEM education, 21st century skills, and technological advancement through robotics competition. *Robotics and Autonomous Systems*, 75:692–699. 11, 143

[Faessler et al., 2016] Faessler, M., Fontana, F., Forster, C., Mueggler, E., Pizzoli, M., and Scaramuzza, D. (2016). Autonomous, vision based flight and live dense 3d mapping with a quadrotor micro aerial vehicle. *Journal of Field Robotics*, 33(4):431–450. 24

[Farroni et al., 2002] Farroni, T., Csibra, G., Simion, F., and Johnson, M. H. (2002). Eye contact detection in humans from birth. *Proceedings of the National Academy of Sciences*, 99(14):9602–9605. 17

[Fathi et al., 2015] Fathi, H., Dai, F., and Lourakis, M. (2015). Automated as built 3d reconstruction of civil infrastructure using computer vision: Achievements, opportunities, and challenges. *Advanced Engineering Informatics*, 29(2):149 – 161. 6, 139

[Filippov et al., 2017] Filippov, S., Ten, N., Shirokolobov, I., and Fradkov, A. (2017). Teaching robotics in secondary school. *IFAC-PapersOnLine*, 50(1):12155–12160. 28, 29, 33

[Fomento, 2018] Fomento, M. (2018). Plan estratégico para el desarrollo del sector civil de los drones en españa, 2018-2021. 3, 135

[Fox et al., 1999] Fox, D., Burgardy, W., Dellaert, F., and Thrun, S. (1999). Monte carlo localization: Efficient position estimation for mobile robots. *In Proc. of the National Conference on Artificial Intelligence.* 66

[Frey and Osborne, 2013] Frey, C. and Osborne, M. (2013). *The future of employment: how susceptible are jobs to computerisation?* 8, 140

[Fuentes-Pacheco et al., 2015] Fuentes-Pacheco, J., Ruiz-Ascencio, J., and Rendón-Mancha, J. M. (2015). Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review.* 25

[Gartshore et al., 2002] Gartshore, R., Aguado, A., and Galambos, C. (2002). Incremental map buildig using an occupancy grid for an autonomous monocular robot. *Proc. of Seventh International Conference on Control, Automation, Robotics and Vision ICARCV, pages 613–618.* 26

[Gaspar et al., 2000] Gaspar, J., Winters, N., and Santos-Victor, J. (2000). Vision based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898. 19

[Gerardo Carrera and Davison, 2011] Gerardo Carrera, A. A. and Davison, A. J. (2011). Lightweight slam and navigation with a multi-camera rig. In *Proceedings of the 5th European Conference on Mobile Robots ECMR 2011*, pages 77 – 82. 25

[Goldberg et al., 2002] Goldberg, S., Maimone, M., and Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. *Proc. of IEEE Aerospace conference Proceedings, pages 5–2025,5–2036.* 18, 22

[Goldman et al., 2004] Goldman, R., Eguchi, A., and Sklar, E. (2004). Using educational robotics to engage inner-city students with technology. In *Proceedings of the 6th International Conference on Learning Sciences*, ICLS '04, pages 214–221. International Society of the Learning Sciences. 26

[Graven and Stott, 2011] Graven, M. and Stott, D. (2011). Exploring online numeracy games for primary learners : sharing experiences of a scifest africa workshop. *Learning and Teaching Mathematics*, 2011(11):10–15. 11, 144

[Hornung et al., 2010] Hornung, A., Bennewitz, M., and Strasdat, H. (2010). Efficient vision based navigation. *Autonomous Robots.* 23

[Hulse et al., 2009] Hulse, M., McBride, S., and Lee, M. (2009). Implementing inhibition of return; embodied visual memory for robotic systems. *Proc. of the 9th Int. Conf. on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems. Lund University Cognitive Studies, 146, pp. 189 - 190.* 20

[Institute, 2017] Institute, M. (2017). Jobs lost, jobs gained: workforce transitions in a time of automation. 8, 140

[Itti and Koch, 2001] Itti, L. and Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews Neuroscience 2, pp. 194-203.* 18, 19

[Itti and Koch., 2005] Itti, L. and Koch., C. (2005). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 7, 50, 140

[Jamieson, 2012] Jamieson, P. (2012). Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? *Miami University, Oxford, OH, 45056.* 28

[Japan-Economic, 2015] Japan-Economic (2015). New robot strategy. 10, 143

[Jensfelt and Kristensen, 2001] Jensfelt, P. and Kristensen, S. (2001). Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Trans. on Robotics and Automation, vol. 17, no. 5, pp. 748-760.* 25

[Jiménez et al., 2010] Jiménez, E., Bravo, E., and Bacca, E. (2010). Tool for experimenting with concepts of mobile robotics as applied to children education. *IEEE Trans. Education*, 53(1):88–95. 10, 28, 33, 142

[Jormanainen and Erkki, 2013] Jormanainen, I. and Erkki, S. (2013). Supporting teachers in unpredictable robotics learning environments. *Publications of the University of Eastern Finland, Dissertations in Forestry and Natural Sciences, September 20th 2013. ISBN: 978-952-61-1230-5.* 34

[Jormanainen and Korhonen, 2010] Jormanainen, I. and Korhonen, P. (2010). Science festivals on computer science recruitment. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 72–73, New York, NY, USA. ACM. 11, 144

[Junior et al., 2013] Junior, L. A., Neto, O. T., Hernandez, M. F., Martins, P. S., Roger, L. L., and Guerra, F. A. (2013). A low-cost and

simple arduino-based educational robotics kit. *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Robotics and Control (JSRC), December edition*, 3(12):1–7. 28

[Kandlhofer and Steinbauer, 2014] Kandlhofer, M. and Steinbauer, G. (2014). Evaluating the impact of robotics in education on pupils' skills and attitudes. In *Proceedings of 4th International Workshop Teaching Robotics*, 5th International Conference Robotics in Education, pages 101–109. 11, 143

[Kubilinskiene et al., 2017] Kubilinskiene, S., Zilinskiene, I., Dagiene, V., and Sinkevièius, V. (2017). Applying robotics in school education: a systematic review. *Baltic Journal of Modern Computing*, 5(1):50. 10, 143

[Lang et al., 2003] Lang, S., Kleinehagenbrock, M., Hohenner, S., Fritsch, J., Fink, G. A., and Sagerer, G. (2003). Providing the basis for human-robot-interaction: A multi-modal attention system for a mobile robot. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 28–35. ACM. 17

[Lefoe, 1998] Lefoe, G. (1998). Creating constructivist learning environments on the web: the challenge in higher education. *ASCILITE Conference Proceedings*. 110

[Magnenat et al., 2014] Magnenat, S., Shin, J., Riedo, F., Siegwart, R., and Ben-Ari, M. (2014). Teaching a core CS concept through robotics. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 315–320. ACM. 10, 28, 143

[Mariottini and Roumeliotis, 2011] Mariottini, G. and Roumeliotis, S. (2011). Active vision based robot localization and navigation in a visual memory. *Proc. of ICRA*. 26

[Marocco and Floreano, 2002] Marocco, D. and Floreano, D. (2002). Active vision and feature selection in evolutionary behavioral systems. *Proc. of Int. Conf. on Simulation of Adaptive Behavior (SAB-7), pp. 247-255*. 7, 18, 19, 140

[Menéndez et al., 2013] Menéndez, B., Cañas, J., Perdices, E., and Vega, J. (2013). Programming a humanoid social robot using the jderobot framework. In *RoboCity2030 11th Workshop, Robots sociales, pp 71-94, U.Carlos III, March 14th 2013. ISBN:978-84-695-7212-2.* 17

[Merkouris et al., 2017] Merkouris, A., Chorianopoulos, K., and Kameas, A. (2017). Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Transactions on Computing Education (TOCE)*, 17(2):9. 10, 143

[Mies and Zentay, 2017] Mies, G. and Zentay, P. (2017). Industrial robots meet industry 4.0. In *XII Hadmernok*. 8, 140

[Mondada et al., 2017] Mondada, F., Bonani, M., Riedo, F., Briod, M., Pereyre, L., Rétornaz, P., and Magnenat, S. (2017). Bringing robotics to formal education: the thymio open-source hardware robot. *IEEE Robotics & Automation Magazine*, 24(1):77–85. 28

[Moreno et al., 2002] Moreno, L., Armingol, J., Garrido, S., De la Escalera, A., and Salichs, M. (2002). A genetic algorithm for mobile robot localization using ultrasonic sensors. *Journal of Intelligent and Robotic Systems*, 34(2):135–154. 25

[Morón, 2015] Morón, C. (2015). *Tesis: la mejora de la práctica docente a través de la metodología de proyectos de investigación.* 34

[Mubin et al., 2013] Mubin, O., Stevens, C. J., and Shahid, S. (2013). A review of the applicability of robots in education. *Technology for Education and Learning, 2013.* 10, 142

[Murgul, 2015] Murgul, V. (2015). Reconstruction of the courtyard spaces of the historical buildings of saint petersburg with creation of atriums. *Procedia Engineering*, 117:808 – 818. 6, 139

[Murray et al., 2003] Murray, R. F., Beutter, B. R., Eckstein, M. P., and Stone, L. S. (2003). Saccadic and perceptual performance in visual search tasks. ii. letter discrimination. *J. Opt. Soc. Am. A*, 20(7):1356–1370. 7, 140

[Navarrete et al., 2016] Navarrete, P., Nettle, C. J., Oliva, C., and Solis, M. A. (2016). Fostering science and technology interest in chilean children with educational robot kits. In *Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016 XIII Latin American*, pages 121–126. IEEE. 11, 28, 32, 33, 143

[Naya et al., 2017] Naya, M., Varela, G., Llamas, L., Bautista, M., Becerra, J. C., Bellas, F., Prieto, A., Deibe, A., and Duro, R. J. (2017). A versatile robotic platform for educational interaction. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 9th IEEE International Conference on*, volume 1, pages 138–144. IEEE. 31

[Nehmzow, 1993] Nehmzow, U. (1993). Animal and robot navigation. *The Biology and Technology of Intelligent Autonomous Agents*. 5, 138

[Newcombe and Davison, 2010] Newcombe, R. A. and Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, pages 1498 – 1505, San Francisco, California (USA). 25

[Nobre, 2015] Nobre, K. (2015). *The Oxford handbook of Attention*. 18

[OECD, 2015] OECD (2015). *Results in Focus*. 34

[O'Hara and Kay, 2003] O'Hara, K. J. and Kay, J. S. (2003). Investigating open source software and educational robotics. *J. Comput. Sci. Coll.*, 18(3):8–16. 26

[Olabe et al., 2011] Olabe, J., Olabe, M., Basogain, X., and Castaño, C. (2011). Programming and robotics with Scratch in primary education. *Education in a Technological World: Communicating current and Emerging Research and Technological Efforts*, pages 356–363. 31

[Opoku and Lehtinen, 2015] Opoku, E. and Lehtinen, E. (2015). Responding to changing student demographics in finland: a study of teachers' developing cultural competence. *Publications of the University of Turku, Finland 2015. ISBN: 978-951-29-6336-2*. 34

[Parks et al., 2014] Parks, D., Borji, A., and Itti, L. (2014). Augmented saliency model using automatic 3d head pose detection and learned gaze following in natural scenes. *Journal of Vision Research.* 17

[Perdices et al., 2010] Perdices, E., Cañas, J., Vega, J., Aguero, C., and Martín, F. (2010). Localización visual de robots en la robocup mediante algoritmos evolutivos. In *Robocity 2030, pp 107-128, Madrid, September 15th 2010. ISBN: 84-693-6777-3.* 62

[Pinto et al., 2015] Pinto, A., Schwartz, W. R., Pedrini, H., and d. R. Rocha, A. (2015). Using visual rhythms for detecting video based facial spoof attacks. *IEEE Transactions on Information Forensics and Security*, 10(5):1025–1038. 6, 139

[Plaza et al., 2017] Plaza, P., Sancristobal, E., Carro, G., Castro, M., Blázquez, M., Muñoz, J., and Álvarez, M. (2017). Scratch as educational tool to introduce robotics. In *International Conference on Interactive Collaborative Learning*, pages 3–14. Springer. 31

[Plaza et al., 2016] Plaza, P., Sancristobal, E., Fernandez, G., Castro, M., and Pérez, C. (2016). Collaborative robotic educational tool based on programmable logic and arduino. In *Technologies Applied to Electronics Teaching (TAEE), 2016*, pages 1–8. IEEE. 28

[Ramachandran, 1990] Ramachandran, V. S. (1990). Visual perception in people and machines. *A. Blake, editor, A.I. and the Eye, chapter 3. Wiley and Sons.* 5, 137

[Rautaray and Agrawal, 2015] Rautaray, S. S. and Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54. 17

[Remazeilles et al., 2006] Remazeilles, A., Chaumette, F., and Gros, P. (2006). 3d navigation based on a visual memory. *Proc. of ICRA*. 18, 22

[Richard and Zisserman, 2003] Richard, H. and Zisserman, Z. A. (2003). *Multiple View Geometry in Computer Vision.* Cambridge University Press. 42

[Richter et al., 2016] Richter, S. R., Vineet, V., Roth, S., and Koltun, V. (2016). Playing for data: Ground truth from computer games. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision ECCV 2016*, pages 102–118. Springer International Publishing. 6, 139

[Robinson, 1964] Robinson, D. A. (1964). The mechanics of human saccadic eye movement. *The Journal of Physiology*, 174(2):245–264. 17

[Rodger and Walker, 1996] Rodger, S. H. and Walker, E. L. (1996). Activities to attract high school girls to computer science. *National Science Foundation's Directorate for Education and Human Resources under the Model Projects for Woman and Girls*. 10, 142

[Rodríguez-Sánchez, 2010] Rodríguez-Sánchez, A. (2010). *Intermediate Visual Representations for Attentive Recognition Systems*. PhD thesis, York University, Toronto, Canadá. 21

[Rose and Meyer, 2002] Rose, D. and Meyer, A. (2002). *Teaching Every Student in the Digital Age: Universal Design for Learning*. 12, 145

[Roy et al., 2015] Roy, D., Gerber, G., Magnenat, S., Riedo, F., Chevalier, M., Oudeyer, P.-Y., and Mondada, F. (2015). IniRobot: a pedagogical kit to initiate children to concepts of robotics and computer science. In *RIE 2015*. 28

[Royer et al., 2007] Royer, E., Lhuillier, M., Dhome, M., and Lavest, J. M. (2007). Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, pages 237–260. 24

[Sawides et al., 2018] Sawides, L., Gambín-Regadera, A., de Castro, A., and Artal, P. (2018). High speed visual stimuli generator to estimate the minimum presentation time required for an orientation discrimination task. *Biomed. Opt. Express*, 9(6):2640–2647. 7, 140

[Schiller, 1996] Schiller, H. (1996). Information inequety. *Nueva York: Routledge*. 9, 142

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117. 6, 138

[Schul, 2011] Schul, J. (2011). Revisiting an old friend: The practice and promise of cooperative learning for the twenty-first century. *The Social Studies*, 102(2):88–93. 34

[Schwab, 2016] Schwab, K. (2016). *The Fourth Industrial Revolution*. World Economic Forum. 8, 140

[Schweikardt and Gross, 2006] Schweikardt, E. and Gross, M. D. (2006). roblocks: A robotic construction kit for mathematics and science education. In *Proceedings of the 8th International Conference on Multimodal Interfaces*, ICMI '06. 13, 146

[Selber, 2004] Selber, S. (2004). *Multiliteracies for a Digital Age*. 12, 145

[Sentance, 2015] Sentance, S. (2015). Annual national survey 2015: Results. technical report, computing at school. Available in https://community.computingatschool.org.uk/files/6098/original.pdf. 32

[Sidner et al., 2004] Sidner, C. L., Kidd, C. D., Lee, C., and Lesh, N. (2004). Where to look: A study of human-robot engagement. In *Proceedings of the 9th International Conference on Intelligent User Interfaces*, IUI '04, pages 78–84. ACM. 17

[Sklar et al., 2003] Sklar, E., Eguchi, A., and Johnson, J. (2003). Robocupjunior: Learning with educational robotics. In Kaminka, G. A., Lima, P. U., and Rojas, R., editors, *RoboCup 2002: Robot Soccer World Cup VI*, pages 238–253. Springer Berlin Heidelberg. 26

[Smith et al., 2018] Smith, J., Wang, J., Chu, F., and Vela, P. (2018). Learning to navigate: Exploiting deep networks to inform sample based planning during vision based navigation. *CoRR*, abs/1801.05132. 24

[Solis et al., 2009] Solis, A., Nayak, A., Stojmenovic, M., and Zaguia, N. (2009). Robust line extraction based on repeated segment directions on image contours. *Proc. of the IEEE Symposium on CISDA*. 41, 126

[Solove, 2004] Solove, D. (2004). *The Digital Person: Technology and Privacy in the Information Age.* 12, 145

[Srinivasan et al., 2006] Srinivasan, V., Thurrowgood, S., and Soccol, D. (2006). An optical system for guidance of terrain following in uavs. *Proc. of the IEEE International Conference on Video and Signal Based Surveillance (AVSS), pages 51–56.* 6, 22, 139

[Steffe and Gale, 1995] Steffe, L. and Gale, J. (1995). *Constructivism in Education.* 110

[Stone and Farkhatdinov, 2017] Stone, A. and Farkhatdinov, I. (2017). Robotics education for children at secondary school level and above. In *Conference Towards Autonomous Robotic Systems*, pages 576–585. Springer. 29, 31, 33

[Tinbergen, 1951] Tinbergen, N. (1951). The study of instinct. *Clarendon University Press, Oxford UK.* 5, 138

[Tirri, 2014] Tirri, K. (2014). The last 40 years in finnish teacher education. *Journal of Education for Teaching, 2014, Vol. 4, pp: 600-607, Finland. DOI: 10.1080/02607476.2014.956545.* 34

[Tirri and Kuusisto, 2013] Tirri, K. and Kuusisto, E. (2013). How finland serves gifted and talented pupils. *Journal for the Education of the Gifted, March 1st 2013, Vol. 36, pp: 84-96, Finland. DOI: 10.1177/0162353212468066.* 34

[Tong and Meng, 2007] Tong, F. and Meng, M. Q.-H. (2007). Genetic algorithm based visual localization for a robot pet in home healthcare system. *Int. J. of Information Acquisition*, 4(2):141–160. 25

[Tsotsos et al., 1995] Tsotsos, J. K., Culhane, S., Wai, W., Lai, Y., and Davis, N. (1995). Modeling visual attention via selective tuning. *Artifical Intelligence 78, pp. 507-545.* 18

[Tweddle et al., 2016] Tweddle, B., Setterfield, T., Saenz, A., and Miller, D. (2016). An open research facility for vision based navigation onboard

the international space station. *Journal of Field Robotics*, 33(2):157–186. 24

[UK-RAS, 2016] UK-RAS (2016). Manufacturing robotics: The next robotic industrial revolution. 8, 141

[Vega, 2011] Vega, J. (2011). *El humor en el aula de matemáticas.* 33

[Vega and Cañas, 2009] Vega, J. and Cañas, J. (2009). Sistema de atención visual para la interacción persona-robot. In *Workshop on Interacción persona-robot, Robocity 2030, pp. 91-110. ISBN: 978-84-692-5987-0.* 4, 6, 7, 136, 139, 140

[Vega and Cañas, 2011] Vega, J. and Cañas, J. (2011). Attentive visual memory for robot navigation. In *WAF2011 XII Physical Agents Workshop, pp 87-94, Albacete, September 6th 2011. ISBN: 978-84-694-6730-5.* 38

[Vega and Cañas, 2014] Vega, J. and Cañas, J. (2014). Curso de robótica en educación secundaria usando constructivismo pedagógico. In *JITICE 4th Workshop, Educational Innovation and ICT, U. Rey Juan Carlos, November 26th 2014. ISSN: 2172-6620.* 110

[Vega and Cañas, 2016] Vega, J. and Cañas, J. (2016). Entorno docente con arduino y python para educación robótica en secundaria. In *JITICE 5th Workshop, Educational Innovation and ICT, U. Rey Juan Carlos, October 25th 2016. ISBN: 978-84-697-0892-7.* 93

[Vega et al., 2010] Vega, J., Cañas, J., Miangolarra, P., and Perdices, E. (2010). Memoria visual atentiva basada en conceptos para un robot móvil. In *Robocity 2030, pp 107-128, Madrid, September 15th 2010. ISBN: 84-693-6777-3.* 40

[Vega et al., 2012a] Vega, J., Cañas, J., and Perdices, E. (2012a). Local robot navigation based on an active visual short-term memory. *Journal of Physical Agents. Volume 6, Number 1, pp 21-30. ISSN: 1888-0258.* 39

[Vega et al., 2018] Vega, J., Cañas, J., and Pérez, F. y Martínez, A. (2018). Jderobot-kids: entorno docente de robótica para niños. *Revista Iberoamericana de Automática e Informática Industrial, RIAI. (Currently under review.)*. 89

[Vega et al., 2012b] Vega, J., Perdices, E., and Cañas, J. (2012b). *Attentive Visual Memory for Robot Localization*, pages 408–438. 39

[Vega et al., 2012c] Vega, J., Perdices, E., and Cañas, J. (2012c). Robot evolutionary localization based on attentive visual short term memory. In *IEEE Intelligent Vehicles Symposium Workshops, Perception in Robotics, Alcalá de Henares, Spain, 3rd june 2012. ISBN: 978-84-695-3472-4*. 39

[Vega et al., 2013] Vega, J., Perdices, E., and Cañas, J. (2013). Robot evolutionary localization based on attentive visual short-term memory. *Sensors, 21st January 2013, 13, 1268-1299; doi:10.3390/s130101268. ISSN: 1424-8220*. 37

[Viola and Jones., 2001] Viola, P. and Jones., M. (2001). Rapid object detection using a boosted cascade of simple features. 46

[von Glasersfeld, 1995] von Glasersfeld, E. (1995). *Radical Constructivism: A Way of Knowing and Learning. Studies in Mathematics Education Series: 6*. 110

[Weiss et al., 2011] Weiss, S., Scaramuzza, D., and Siegwart, R. (2011). Monocular slam based navigation for autonomous micro helicopters in gps denied environments. *Journal of Field Robotics*, 28(6):854–874. 25

[Witherspoon et al., 2017] Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., and Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education (TOCE)*, 18(1):4. 30, 32

[Wolcott and Eustice, 2014] Wolcott, R. W. and Eustice, R. M. (2014). Visual localization within lidar maps for automated urban driving. In *2014 IEEE RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. 7, 139

[Wresch, 1996] Wresch, W. (1996). Disconnected. haves and have-nots in the information age. *New Brunswick, Nueva Jersey: Rutgers University Press.* 9, 142

[Zaharescu et al., 2005] Zaharescu, A., Rothenstein, A. L., and Tsotsos, J. K. (2005). Towards a biologically plausible active visual search model. *Proc. of Int. Workshop on Attention and Performance in Computational Vision WAPCV-2004, pp. 133-147.* 7, 8, 19, 139, 140