Robotic Vision:

Technologies for Machine Learning and Vision Applications

José García Rodríguez
University of Alicante, Spain

Miguel Cazorla
University of Alicante, Spain

Information Science
REFERENCE

Managing Director: Lindsay Johnston
Editorial Director: Joel Gamon
Book Production Manager: Jennifer Yoder
Publishing Systems Analyst: Adrienne Freeland
Development Editor: Christine Smith
Assistant Acquisitions Editor: Kayla Wolfe
Typesetter: Erin O'Dea
Cover Design: Nick Newcomer

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Robotic vision: technologies for machine learning and vision applications / Jose Garcia-Rodriguez and Miguel A. Cazorla Quevedo, editors.
pages cm
Summary: "This book offers comprehensive coverage of the current research on the fields of robotics, machine vision, image processing and pattern recognition that is important to applying machine vision methods in the real world"-- Provided by publisher.
Includes bibliographical references and index.
ISBN 978-1-4666-2672-0 (hardcover) -- ISBN 978-1-4666-2703-1 (ebook) -- ISBN 978-1-4666-2734-5 (print & perpetual access) 1. Computer vision. 2. Pattern recognition systems. 3. Image processing. 4. Robotics--Human factors. I. Garcia-Rodriguez, Jose, 1970- II. Cazorla Quevedo, Miguel A., 1970-
TA1634.R63 2013
629.8'92637--dc23
2012029113

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

List of Reviewers

Raed Almodani, *Wayne State University, USA*
Cecilio Angulo, *CETpD-UPC, Universitat Politècnica de Catalunya, Spain*
Roseli Aparecida, *University of Sao Paulo, Brazil*
Jorge Azorín, *University of Alicante, Spain*
Antonio Bandera, *University of Málaga, Spain*
Juan Pedro Bandera, *University of Málaga, Spain*
Douglas Brooks, *Georgia Institute of Technology, USA*
Ivan Cabezas, *Universidad del Valle, Colombia*
José María Cañas, *Rey Juan Carlos University, Spain*
Ming Dong, *Wayne State University, USA*
Andrés Fuster, *University of Alicante, Spain*
Manuel Graña, *Basque Country University (UPV/EHU), Spain*
Juan Manuel Guede, *Basque Country University (UPV/EHU), Spain*
Ayanna M. Howard, *Georgia Institute of Technology, USA*
Laura Igual, *MAiA-UB, Spain*
Jim Juola, *University of Kansas, USA*
Vicente Matellán, *University of León, Spain*
Neils Meins, *University of Hamburg, Germany*
Vicente Morell, *University of Alicante, Spain*
Ramon Moreno, *University of Alicante, Spain*
Lazaros Nalpantidis, *Royal Institute of Technology – KTH, Sweden*
Sergio Orts, *Universitat Autonoma de Barcelona, Spain*
Chung Park, *Georgia Institute of Technology, USA*
Hae Wong Park, *Georgia Institute of Technology, USA*
Xavier Perez-Sala, *CETpD-UPC, Universitat Politècnica de Catalunya, Spain*
Domenec Puig, *University of Rovira I Virgili, Spain*
Renato Ramos, *University of Sao Paulo, Brazil*
J.A. Rodríguez, *University of Málaga, Spain*
Sreela Sasi, *Gannon University, USA*
Marcelo Saval, *University of Alicante, Spain*
Mohan Sridharan, *Texas Tech University, USA*
Elena Torta, *Eindhoven University of Technology, The Netherlands*
Diego Viejo, *University of Alicante, Spain*
Wenjie Yan, *University of Hamburg, Germany*

Section 6
Visual Attention

The purpose of this chapter is both to review some of the most representative visual attention models, both theoretical and practical, that have been proposed to date, and to introduce the authors' attention model, which has been successfully used as part of the control system of a robotic platform. The chapter has three sections: in the first section, an introduction to visual attention is given. In the second section, relevant state of art in visual attention is reviewed. This review is organised in three areas: psychological based models, connectionist models, and features-based models. In the last section, the authors' attention model is presented.

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. This chapter proposes a visual perceptive system for a robot with a mobile camera on board that cope with these two issues. The system is composed of a dynamic visual memory that stores the information gathered from images, an attention system that continuously chooses where to look at, and a visual evolutionary localization algorithm that uses the visual memory as input. The visual memory is a collection of relevant task-oriented objects and 3D segments. Its scope and persistence is wider than the camera field of view and so provides more information about robot surroundings and more robustness to occlusions than current image. The control software takes its contents into account when making behavior or navigation decisions. The attention system considers the need of reobserving objects already stored, of exploring new areas and of testing hypothesis about objects in the robot surroundings. A robust evolutionary localization algorithm has been developed that can use both the current instantaneous images or the visual memory. The system has been programmed and several experiments have been carried out both with simulated and real robots (wheeled Pioneer and Nao humanoid) to validate it.

Computer vision systems have to deal with thousands, sometimes millions of pixel values from each frame, and the computational complexity of many problems related to the interpretation of image data is very high. The task becomes especially difficult if a system has to operate in real-time. Within the

SECTION 6: VISUAL ATTENTION

García, Rodríguez, and Matellán make a review of some of the most representative visual attention models, which can be used for reducing the time to process images by a robot. Vega, Perdices, and Cañas propose a visual perceptive system for a robot with a mobile camera on board that copes with two challenges arising when using cameras: to extract useful information from captured images and to manage the small field of view of regular cameras. The chapter by Antúnez, Haxhimusa, Marfil, Kropatsch, and A. Bandera proposes a visual attention model using a hierarchical grouping process that encodes the input image into a Combinatorial Pyramid.

José García Rodríguez
University of Alicante, Spain

Miguel Cazorla
University of Alicante, Spain

Chapter 21
Attentive Visual Memory for Robot Localization

Julio Vega
Rey Juan Carlos University, Spain

Eduardo Perdices
Rey Juan Carlos University, Spain

José María Cañas
Rey Juan Carlos University, Spain

ABSTRACT

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. This chapter proposes a visual perceptive system for a robot with a mobile camera on board that cope with these two issues. The system is composed of a dynamic visual memory that stores the information gathered from images, an attention system that continuously chooses where to look at, and a visual evolutionary localization algorithm that uses the visual memory as input. The visual memory is a collection of relevant task-oriented objects and 3D segments. Its scope and persistence is wider than the camera field of view and so provides more information about robot surroundings and more robustness to occlusions than current image. The control software takes its contents into account when making behavior or navigation decisions. The attention system considers the need of reobserving objects already stored, of exploring new areas and of testing hypothesis about objects in the robot surroundings. A robust evolutionary localization algorithm has been developed that can use both the current instantaneous images or the visual memory. The system has been programmed and several experiments have been carried out both with simulated and real robots (wheeled Pioneer and Nao humanoid) to validate it.

DOI: 10.4018/978-1-4666-2672-0.ch021

INTRODUCTION

Computer vision research is growing rapidly, both in robotics and in many other applications, from surveillance systems for security to the automatic acquisition of 3D models for Virtual Reality displays. The number of commercial applications is increasing, like traffic monitoring, parking entrance control, augmented reality videogames and face recognition. In addition, computer vision is one of the most successful sensing modalities used in mobile robotics. Cameras have been incorporated in the last years to robots as common sensory equipment. They are very cheap sensors and may provide much information to robots about their environment. However, extracting relevant information from the image flow is not easy. Vision has been used in robotics for navigation, object recognition, 3D mapping, visual attention, robot localization, etc.

Robots usually navigate autonomously in dynamic environments, and so they need to detect and avoid obstacles. There are several sensors which can detect obstacles in robot's path, such as infrared sensors, laser range finders, ultrasound sensors, etc. When using cameras, obstacles can be detected through 3D reconstruction. Recovering 3D-information has been the main focus of the computer vision community for decades. Stereo-vision methods are the classic ones, based on finding pixel correspondences between the two cameras and triangulation, despite they fail with untextured surfaces. Vision depth sensors like Kinect offer now a different technology for visual 3D reconstruction. In addition, structure from motion techniques builds three-dimensional structure of objects by analyzing local motion signals over time, even from only one camera (Richard & Zisserman, 2003).

Moreover, many works have also been presented in vision based navigation and control that generate robot behavior without explicit 3D reconstruction. The temporal occlusions of relevant stimuli inside the images are one hindrance in this approach. The control algorithm should be robust to the lack of time persistence of relevant stimuli in images. This also poses a problem when the objects lie beyond the current field of view of the camera. To solve it, some systems use omnidirectional vision. Others, like humanoids or robots with pantilt units, use mobile regular cameras that can be orientated at will and manage a visual memory of robot surroundings that integrate the information from the images taken from different locations. The visual representation of interesting objects around the robot beyond current field of view may improve the quality of robot's behavior as it handles more information when making decisions. The problem of selecting where-to-look-at at every time, known as gaze control or overt attention (Itti & Koch, 2001; Zaharescu *et al.*, 2005), arises there. Usually the need to quickly explore new areas and the need to reobserve known objects to update their positions, etc. influence that selection. This kind of attention is also present in humans, as we are able to concentrate on particular regions of interest in a scene by movements of the eyes and the head, just by shifting attention to different parts of it. By driving attention specifically to small regions which are important for the task at hand we avoid wasting effort and processing trying fully understand the whole surroundings, and devote as much as possible only to the relevant part.

Another relevant information that can be extracted from images is robot location. Robots need to know their location inside the environment in order to unfold the proper behavior. Using robot sensors (specially vision) and a map, the robot may estimate its own position and orientation inside a known environment. Robot self-localization has proven to be complex, especially in dynamic environments and in those with much symmetry, where sensors values can be similar at different positions.

In this chapter we propose a visual perceptive system 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 has been developed to select where the camera should look at at every time. It manages the movement of the camera to periodically reobserve objects already stored in the visual memory, explore the scene and 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 has been developed that uses current image or the contents of the memory to continuously estimate the robot position. It provides a robust localization estimation and has been specifically designed to bear with symmetries in the environment.

The remainder of this chapter is organized as follows. Second section reviews some of the related works. Third section presents the design of the proposed visual system, its components and connections. The next three sections describe its three building blocks: visual memory component, visual attention algorithm and localization component. The seventh section includes several experiments, both with simulated and real robots, performed to validate our system. Finally some conclusions are summarized.

RELATED WORKS

Many issues have been tackled in the intersection of computer vision and robotic fields: vision-based control or navigation, vision-based map building and 3D representation, vision-based localization, object recognition, attention and gaze control among others. We will review here some examples in the topics most related with the proposed visual perceptive system.

Regarding vision based control and navigation, Remazeilles (Remazeilles *et al.*, 2006) presented the design of a control law for vision-based robot navigation. The particularity of this control law is that it does not require any reconstruction of the environment, and it does not force the robot to converge towards each intermediary position in its path.

Recently, Srinivasan (Srinivasan *et al.*, 2006) presented a new system to increase accuracy in the optical flow estimation for insect-based flying control systems. A special mirror surface is mounted in front of the camera, which is pointing ahead instead of pointing to the ground. The mirror surface decreases the speed of motion and eliminates the distortion caused by the perspective. In this image objects move slower than in the camera downwards, simplifying the optical flow calculation and increasing its accuracy. Consequently, the system increases the speed range and the number of situations under which the aircraft can fly safely.

Regarding visual map building, representation of the environment and navigation, Badal (Badal *et al.*, 1994) reported a system for extracting range information and performing obstacle detection and avoidance in outdoor environments based on the computation of disparity from the two images of a stereo pair of calibrated cameras. The system assumes that objects protrude high from a flat floor that stands out from the background. Every point above the ground is configured as a potential object and projected onto the ground plane, in a local occupancy grid called Instantaneous Obstacle Map (IOM). The commands to steer the robot are generated according to the position of obstacles in the IOM.

Goldberg (Goldberg *et al.*, 2002) introduced a stereo vision-based navigation algorithm for the rover planetary explorer MER, to explore and map locally hazardous terrains. The algorithm computes epipolar lines between the two stereo frames to check the presence of an object, computes the Laplacian of both images and correlates the filtered images to match pixels from the left image with their corresponding pixels in the right image. The work also includes a description of

the navigation module GESTALT, which packages a set of routines able to compute actuation, direction, or steering commands from the sensor information.

Gartshore (Gartshore *et al.*, 2002) developed a map building framework and a feature position detector algorithm that processes images on-line from a single camera. The system does not use any matching. Instead, it computes probabilities of finding objects at every location. The algorithm starts detecting the object boundaries for the current frame using the Harris edge and corner detectors. Detected features are backprojected from the 2D image plane considering all the potential locations at any depth. The positioning module of the system computes the position of the robot using odometry data combined with image feature extraction. Color or gradient from edges and features from past images help to increase the confidence of the object presence in a certain location. Experimental results in indoor environments set the size of the grid cells to 0.25m x 0.25m and the robot moved 0.1m between consecutive images.

As we already mentioned, in autonomous robots it is important to perform a visual attention control. The cameras of the robots provide a large flow of data and the robot needs to select what is interesting and ignore what it is not. There are two approaches to visual attention: covert and overt attention. The first one selects interesting areas inside the image for further processing (Tsotsos *et al.*, 1995; Itti & Koch, 2001), (Marocco & Floreano, 2002). The second one selects interesting areas in the environment surrounding the robot, even beyond the current field of view, and looks at them (Cañas *et al.*, 2008).

One of the concepts widely accepted in this area is the salience map. It is found in (Itti & Koch, 2001), as a covert visual attention mechanism, independent of the particular task to be performed. This bottom-up attention builds in each iteration the conspicuity map for each one of the visual features that attract attention (as color, movement or edge orientations). There are competition dynamics inside each map and they are merged into a single representative salience map that drives the focus of attention to the area with highest value.

Hulse (Hulse *et al.*, 2009) presented an active robotic vision system based on the biological phenomenon of inhibition of return, used to modulate the action selection process for saccadic camera movements. They argued that visual information has to be subsequently processed by a number of cortical and sub-cortical structures that place it: 1) in context of attentional bias within egocentric salience maps; 2) the aforementioned IOR inputs from other modalities; 3) overriding voluntary saccades and 4) basal ganglia action selection. Thus, biologically there is a highly developed, context specific method for facilitating the most appropriate saccade as a form of attention selection.

The use of a camera in motion to facilitate object recognition was proposed by (Ballard, 1991), and has been used, for example, to distinguish between different forms in the images (Marocco & Floreano, 2002). Arbel and Ferrie presented in (Arbel & Ferrie, 2001) a gaze-planning strategy that moves the camera to another viewpoint around an object in order to recognize it. Recognition itself is based on the optical flow signatures that result from the camera motion. The new measurements, accumulated over time, are used in a one-step-ahead Bayesian approach that resolves the object recognition ambiguity, while it navigates with an entropy map.

Grid based probabilistic localization algorithms were successfully applied with laser or sonar data in small known environments (Burgard & 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 estimation. Particle filters use sampled probability functions and extend the techniques to larger environments, using them even with visual data as input (Dellaert *et al.*, 1999). At the beginning the maps were provided in advance but in the last years the SLAM techniques tackle localization simultaneously with the map building. There are

many particle filter based SLAM techniques. In addition, one of the most successful approaches is Mono-SLAM from Andrew Davison (Newcombe & Davison, 2010; Carrera *et al.*, 2011) based on a fast Extended Kalman Filter for continuous estimation of 3D points and camera position from relevant points in the image.

In (Mariottini & Roumeliotis, 2011) Mariottini and Roumeliotis presented a strategy for active vision-based localization and navigation of a mobile robot with a visual memory where previously visited areas are represented as a large collection of images. It disambiguates the location taking into account the sequence of distinctive images, while concurrently navigating towards the target image.

In (Jensfelt & Kristensen, 2011), Jensfelt and Kristensen presented an active global localization strategy that uses Kalman filtering (KF) to track multiple robot pose hypotheses. Their approach can be used even with incomplete maps and the computational complexity is independent on the size of the environment.

DESIGN

The proposed perceptive system is designed for autonomous robots that use a mobile camera, like that on the head of humanoids or in robots with pan-tilt units. The block diagram of the robot control architecture is showed in Figure 1. The three building blocks (visual memory, gaze control and localization algorithm) have been 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 objects around the robot or robot position. This information is provided to other actuation components like the navigation algorithm or other control units.

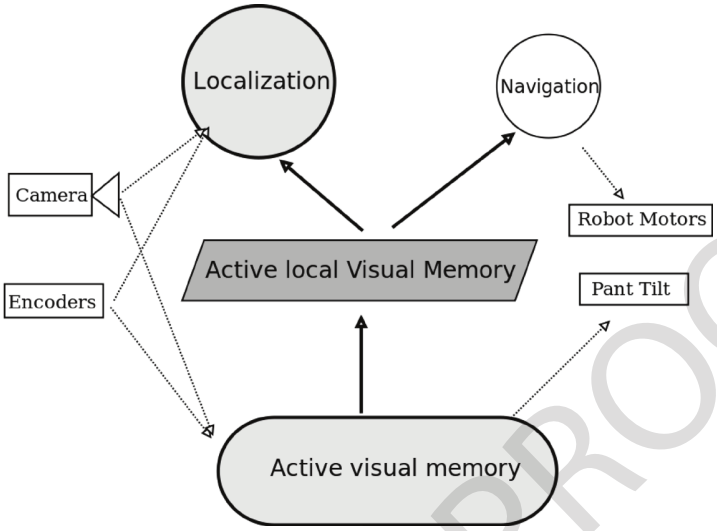
First, the active_visual_memory component builds a local visual memory of objects in the robot's surroundings. The memory is built analyzing each camera image looking for relevant objects (like segments, parallelograms, arrows, etc) and updating the object features already stored in the memory like their 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 deals only with objects on the floor plane (ground hypothesis) 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. 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 has been designed to continuously guide camera movements, choosing where to look at at every time. It has been 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

Figure 1. Block diagram of the proposed visual system



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 has been developed in the localization component. It uses a population of particles and an evolutionary algorithm to manage them and find the robot position. 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.

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

LOCAL VISUAL MEMORY

The goal of our 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 position to the robot and remove them from the memory once they have disappeared.

Figure 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 (we assume all objects are flat on the floor as simplifying hypothesis). Finally, the 3D memory module stores their position in 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.

The visual memory also creates perceptual hypothesis with the stored items, allowing the system to abstract complex objects. For instance, it groups a set of 3D segments into the parallelogram concept if some geometric properties are hold.

The visual memory has been 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 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 only analyze frames when the robot has moved away a certain distance or angle from its position when the last image processing was done.

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 releases we 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. In last releases we replaced this 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 smooth, thresholding, and Laplace edge detection to extract edge contours from input images. This solution is surprisingly more accurate, robust, faster and with less parameters than the widely used Hough Transform algorithm for detecting lines segments at any orientation and location.

To implement these techniques, we use the OpenCV library. A comparison can be seen in Figure 3 where the Solis algorithm extract 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.

Figure 2. Modules of the visual memory

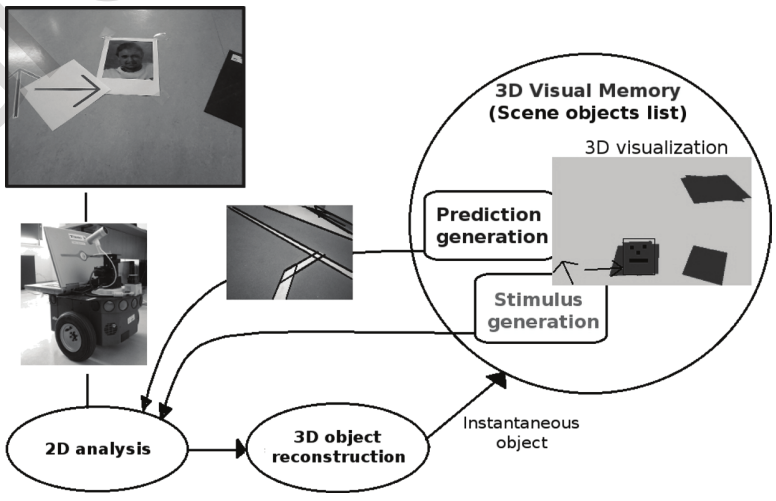
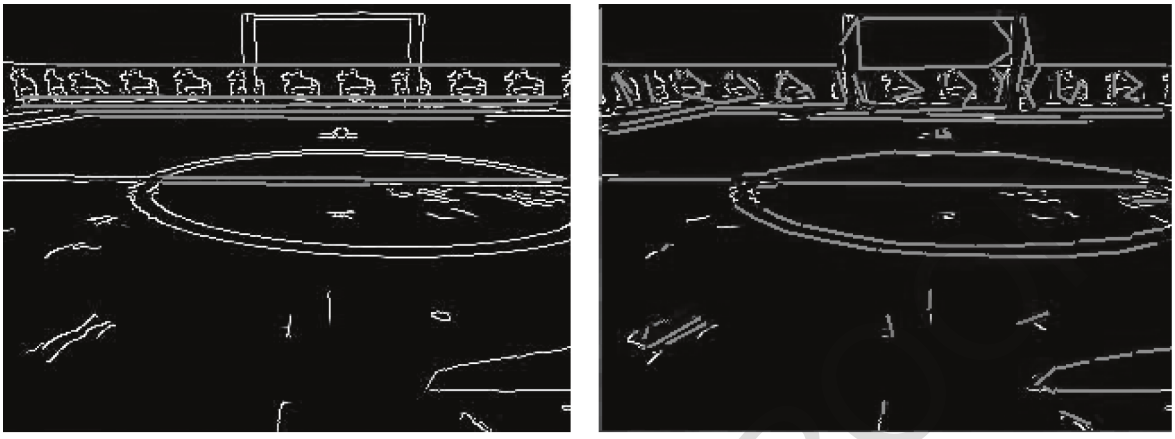


Figure 3. Differences between Canny+Hough (left) and Solis algorithm (right)



(a) (b)

The 2D analysis system is connected directly to the 3D visual memory 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 stored in the 3D memory which are visible from the current position. We use our projective geometry library to do this. Each stored 3D visible object is projected on the image plane as shown in Figure 4 (left). The system refutes/corroborates such predicted segments, comparing them with those coming from the 2D analysis on observed images. This comparison provides three sets of segments, as seen in Figure 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.

Reconstruction with 3D Segments

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

The are four relevant 3D coordinate systems in this approach, as shown in Figure 5. First, the absolute coordinate system whose origin lies somewhere in the world where the robot is moving. Second, the system located at the base of the robot. The robot odometry gives its position and orientation with respect to the absolute system. Third, the system relative to the base of the pan-tilt unit to which the camera is attached to. 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, displaced and oriented in a particular mechanical axis from the pan-tilt unit.

Once we have the 3D segments, and before including them on the 3D memory, some post-processing is needed to avoid duplicates in memory due to noise in the images. This postprocessing 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 6 shows the segments detected in the current image, the segments predicted from the current position, and all of them in the 3D observed scene.

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 arrows, parallelograms, triangles, circles or other objects.

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 as the time the segment remains in memory and is also taken into account and updated. 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, the 3D segments have an attribute named life which decreases as the time 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.

Complex Primitives in Visual Memory

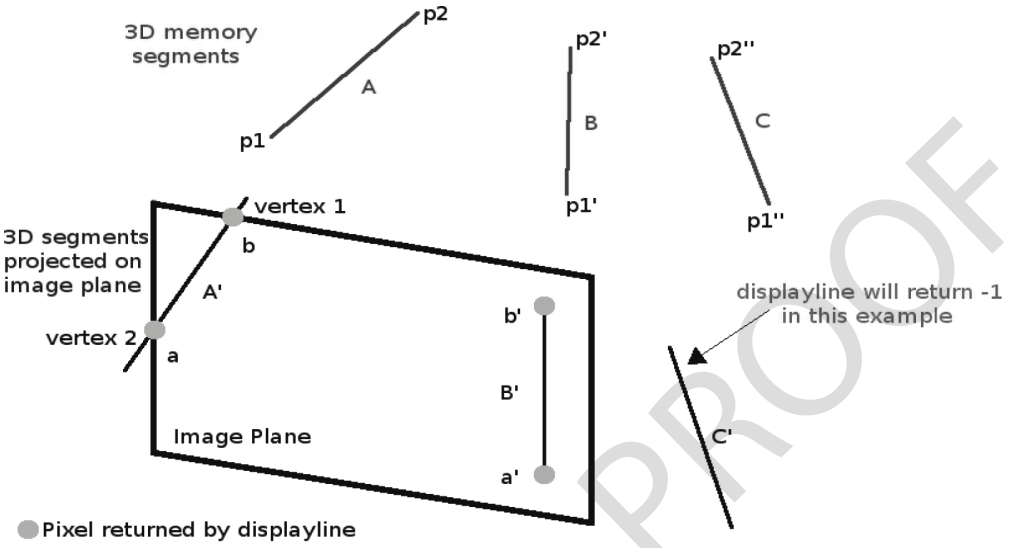
Visual memory manages simple 3D segments and other primitives like parelellograms. The segments and their corresponding vertices are used to detect parallelograms checking the connection between them and the parallelism conditions. The analysis of the angles formed by each segment provides information 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. This capability makes our algorithm robust against occlusions, which occur frequently in the real world.

Figure 7 illustrates an example of occlusion that is satisfactorily solved by our algorithm. The (a) and (b) images show the observation on incomplete parallelograms and include noise. The results of reconstruction of parallelograms can be seen in Figure 7-(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.

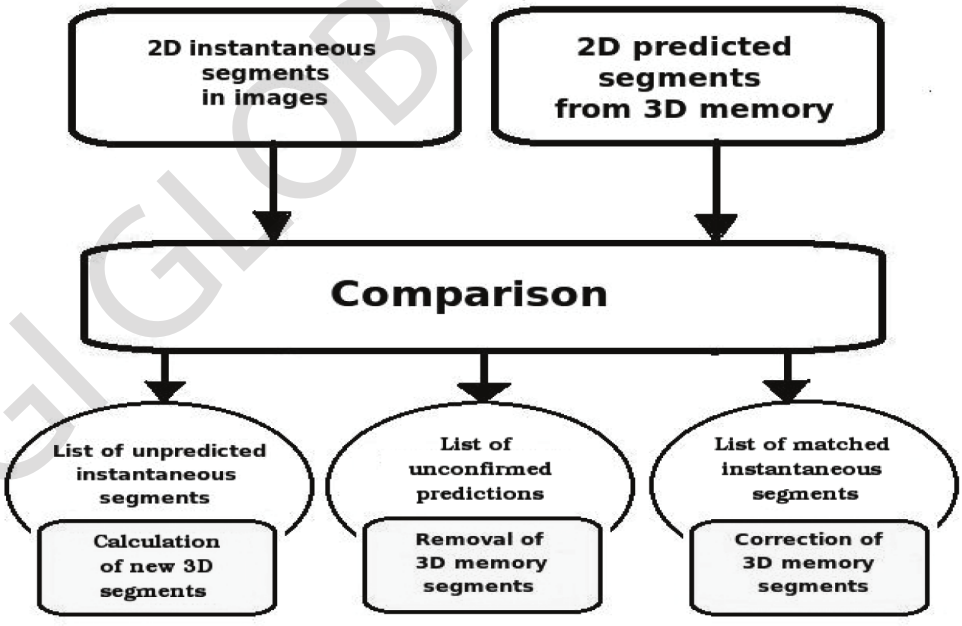
VISUAL ATTENTION

The second building block of the proposed visual perception system is the visual attention. It uses two object attributes: salience and life to decide where to look at at every moment and to forget objects when they disappear from the scene. In addition, this block includes a mechanism to con-

Figure 4. 3D projection on the image plane (left) and matching between predicted and observed segments (right)

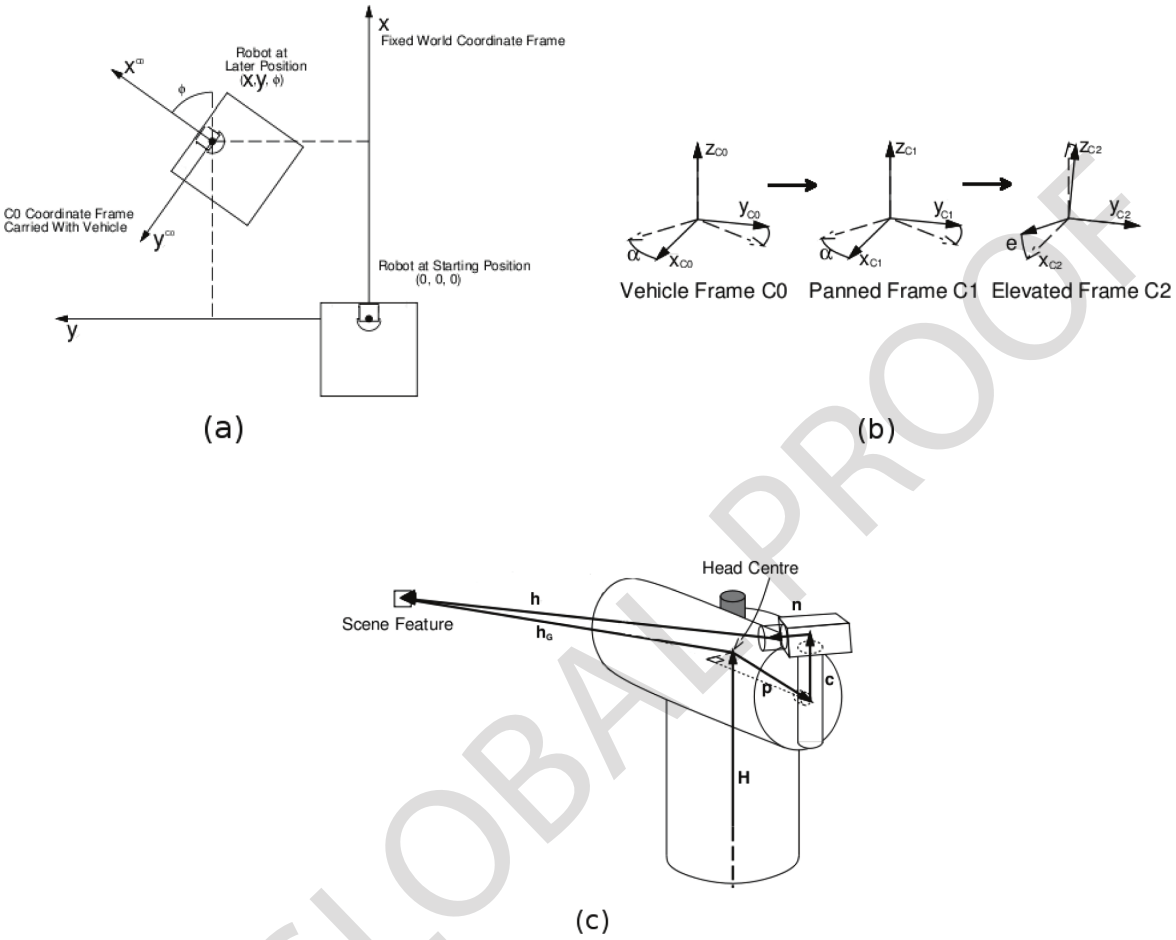


(a)



(b)

Figure 5. 3D coordinate systems used



trol the camera movements for object tracking and exploring of new unknown areas from the scene.

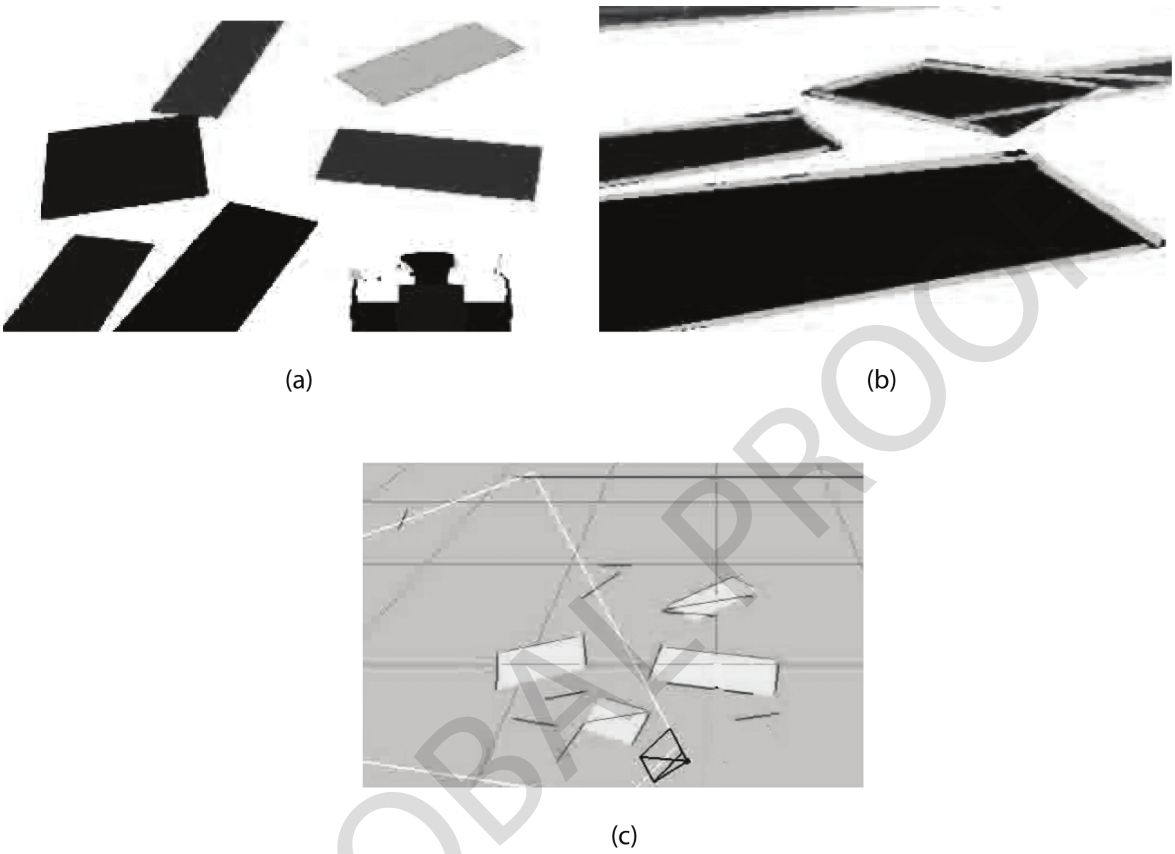
Gaze Control: Salience Dynamics

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, we introduced the dynamics of salience and attention points. 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, following the equation:

$$Salience(t) = \begin{cases} Salience(t-1) \\ if\ \ object\ \ attended \\ Salience(t-1)+1 \\ otherwise \end{cases}$$

The system continuously computes the salience of all attention points a chooses the most salient one to control the gaze, the pan and tilt orders that makes the camera look at it are computed and commanded to the pantilt unit. When a point is visited, its salience is set to 0. A point that the system has not visited recently calls more attention than one which has just been attended. If the salience is low, it will not be visited now. The

Figure 6. Scene situation, instantaneous image, and 3D scene reconstruction



system is thus similar to the behavior of a human eye, as pointed by biology studies (Itti & Koch, 2005) when the eye responds to a stimulus that appears in a position that has been previously treated, the reaction time is usually higher than when the stimulus appears in a new position.

After a while the most salient point 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. We assume that an object will be found near the location where it was previously observed the last time. If the object motion were too fast the reobservations would not match with current object location.

In our 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. If we assigned different rates of growth of salience we could have different priorities for the objects, causing the pan-tilt unit to look more times at objects whose salience grows faster.

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 in the case it moves spatially. For this tracking we use two proportional controllers to command the pan and tilt speeds and thus continually keep that object on the image center. The controller follows next

equations (Figure 8), where: K_p is the P control gain, T_t is the Tilt of the target, T_a is the actual Tilt, P_t is the Pan of the target, P_a is the actual Pan, M_t is the maximum Tilt and M_p is the maximum Pan.

$$v(Pan) = \begin{cases} 0 \quad if \quad \in < 0.3 \\ K_p \cdot (P_t - P_a) \quad if \quad 0.3 \leq \in < M_p \\ M_p \quad if \quad M_p < \in \end{cases}$$

$$v(Tilt) = \begin{cases} 0 \quad if \quad \in < 0.1 \\ K_p \cdot (T_t - T_a) \quad if \quad 0.1 \leq \in < M\_t \\ M_t \quad if \quad M_t < \in \end{cases}$$

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 unknowns areas of the scene around the robot with objects of interest. For that search our 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.

Figure 7. Complex primitives in visual memory: parallelograms with occlusion

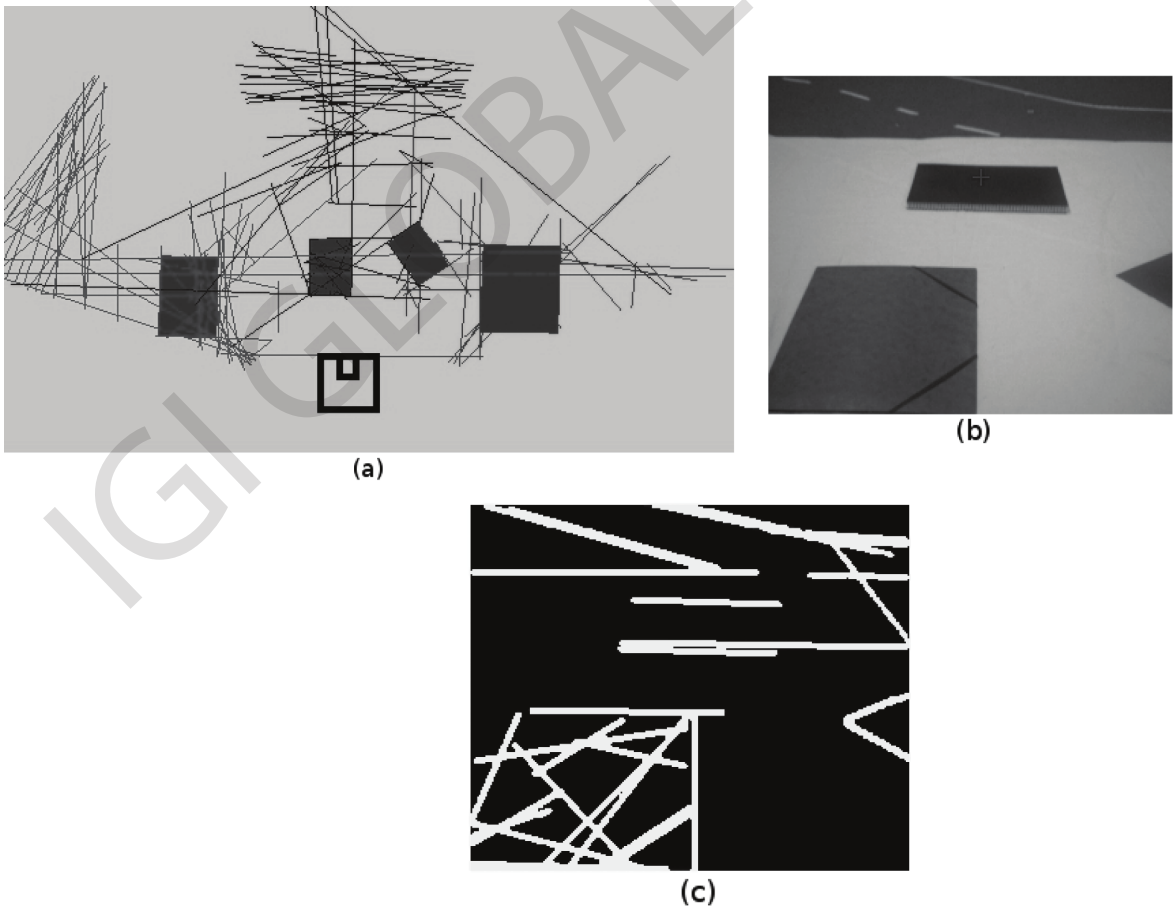
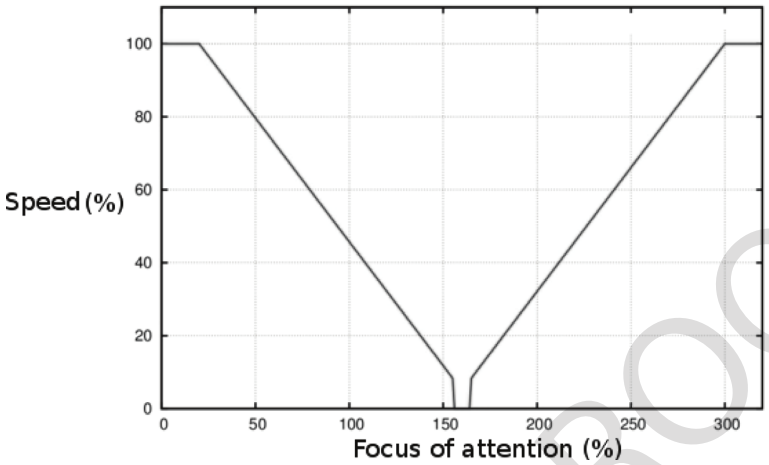


Figure 8. P-controller mechanism



The scanning points can be of two types: random and systematic ones. Random points are distributed uniformly within the pan-tilt range (pan = (-159, + 159), tilt = (-31, +31)). 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.

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 memory in order to maintain coherence between the representation of the scene and the reality. To forget such old elements, we have implemented the life dynamics that follows this equation.

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

Life of unobserved 3D segments or objects decrease over time and every time an object or 3D segment and it is observed in the images (just because the gaze control visits it or 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 is below it that means it has gone and is deleted from visual memory.

Attention Module Operation

The proposed 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 have 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.

It has been implemented following a state-machine 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 we have 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 automata goes to stage 2 and it will analyze whether there are relevant objects in the images or not. After a while it returns to state 0 and starts again.

EVOLUTIONARY VISUAL LOCALIZATION

We have designed a new approach to solve robot self-localization specifically designed to bear 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 evolve over time using genetic operators, such as mutation or crossover. Each individual is a tentative robot position (X, Y, θ) and 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. We have defined two different health functions, one based on instantaneous measures of robot sensors and another one based on the visual memory contents.

Races are set of individuals around a given location,they perform a fine-grain search around it. The algorithm has N 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 races and the algorithm will obtain the real

Figure 9. Basic diagram of evolutionary algorithm

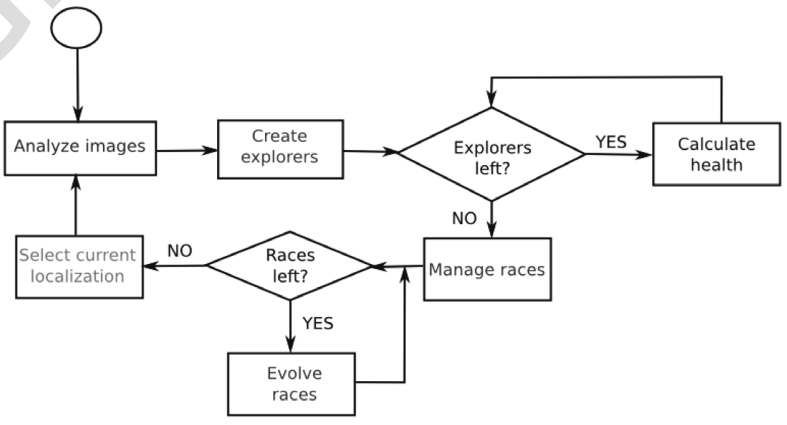


Figure 10. Image analysis before merging (left) and after merging (right)

(a) (b)

robot pose from the best race. On each iteration the algorithm performs several steps:

- Health race calculation using the information obtained after analyzing images.
- **Explorers creation:** We spread randomly new individuals with the aim to find new candidate positions where new races could be created.
- **Races management:** We create, merge or delete races depending on their current state.
- **Races evolution:** We evolve each race by using genetic operators. Besides, if the robot is moving, we update all races taking into account this movement.
- After calculating each race health, we will select one of them to set the current robot pose.

On each iteration of the algorithm there are several steps to be performed to know the current robot pose (Figure 9). They are described in the following sections.

Analyzing Images

When the localization algorithm uses the instantaneous camera images, it first detects lines inside the images following the same technique described in "Local Visual Memory" section. This time the algorithm does not discard segments over the horizon, as it does not require that objects lie on the floor. 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 very close to each other, joining consecutive segments together (Figure 10). Too small segments are ruled out.

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 explained in the next section. We created a grid with different cell sizes and we only save a new point where the lines detected intersect with this grid (Figure 11). The size of this grid cells changes because we want 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 sampling points selected will be the input data to calculate the health of each individual.

Health Calculation from Instantaneous 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 one are, the more likely such location is the correct one.

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 12). It contains the lines the robot would see if it were placed at that location (Figure 10 (right)).

For each sampling points in the observed lines the Euclidean distance d_i in pixels to the closest theoretical line with the same label is computed. After calculating d_i for all points, the Individual's health is computed as the average distance, following the next equation, where N is the number of points and M is set to 50 pixels for a 320x240 image size.

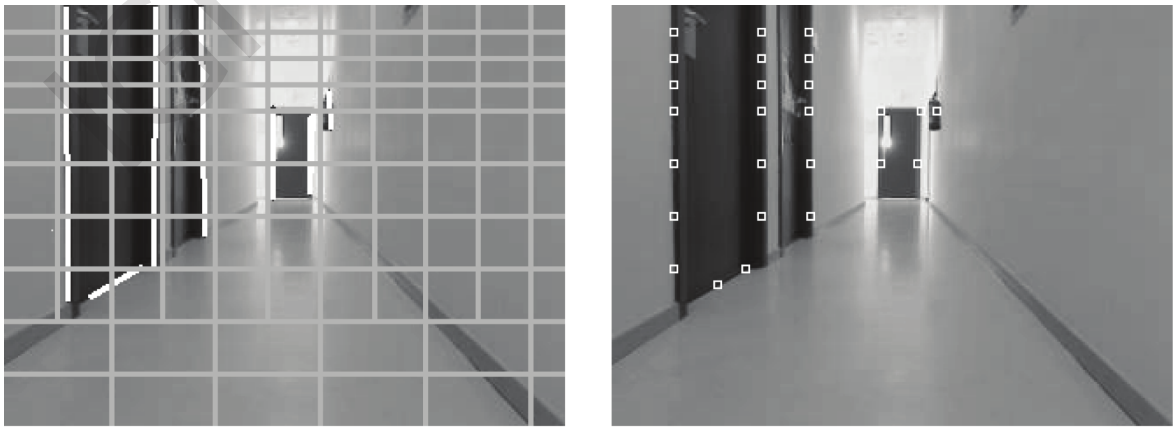
$$H = 1 - \frac{\sum_{i=0}^{N} \frac{d_i}{N}}{M}$$

We will show in "Experiments" section several experiments to analyze the health function behavior in different situations.

Health Calculation with Visual Memory

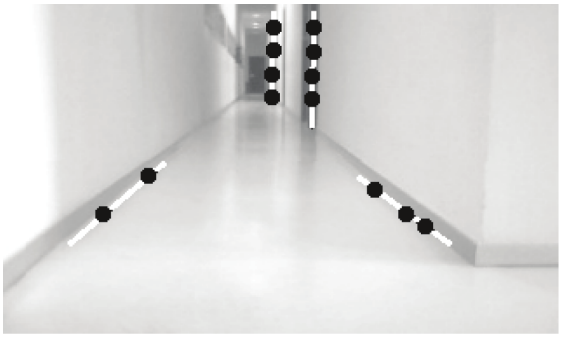
In case of using the visual memory we don't need to analyze each image, just the current visual memory contents from the active_visual_memory component, that is, the set of 3D segments inside, relative to our robot. So we can't compare lines in image as we did before. Besides, we have to take into account that lines may not be detected completely or they may be divided into several small lines. Thus, to calculate the current health of an individual we cover all lines belonging to the visual memory, for each line we get its extremes

Figure 11. Image grid (left) and points selected (right)

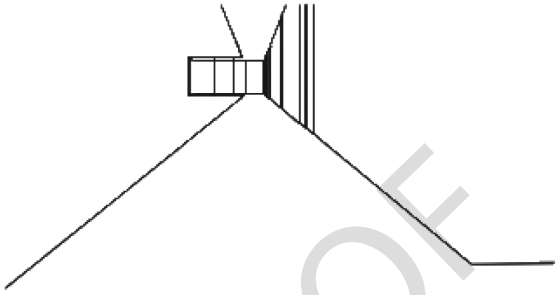


(a) (b)

Figure 12. Lines detected in current image and theoretical image



(a)



(b)

and calculate the Euclidean distances d_{js} and d_{je} to the closest theoretical line with the same label, that is, similar to health function with instantaneous images but in 3D. After calculating d_{js} and d_{je} for each line, we can calculate the health as follows, where N is the number of lines and M is set to 0.5 meters.

$$H = 1 - \frac{\sum_{i=0}^{N} \left(\frac{d_{js} + d_{je}}{2} \right)}{M}$$

Explorer Creation

Explorers are individuals that don't 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 overfitting of the algorithm we chose the random approach.

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

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 increase 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 other 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 has been 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't be deleted or replaced (although it may be merged) until its age reaches 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. It provides some stability to races and avoids the continuous creation and deletion of races.

Our approach has a maximum number or races N 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 became a race, we have to decide when to create new races and when to replace an existing one. If the maximum number of races N 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 don't 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 has been reached, the candidate will replace a victim race when the health of the candidate is greater than that of the victim and the victim's victories parameter falls below 0. If any race can't be replaced, candidates are ignored.

In case two races evolving towards the same localization, we consider that they have led to the same solution, so we merge them. This merging consist of deleting the race with lower victories, or 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 in the real robot pose anymore, the algorithm considers it is wrong and deletes it.

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 N best individuals of each race, arranges them according to their health. They are saved in the next iteration without any change. With crossover the algorithm randomly selects several

Figure 13. (a) Situation; (b) instantaneous image; (c) short-term memory

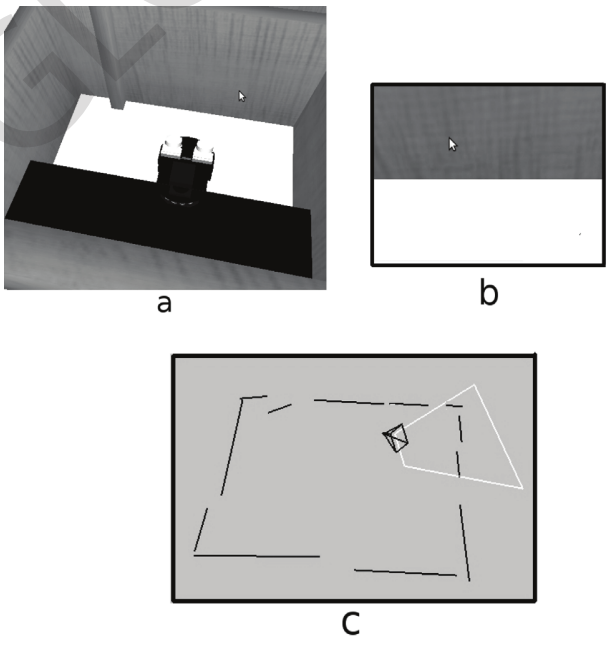
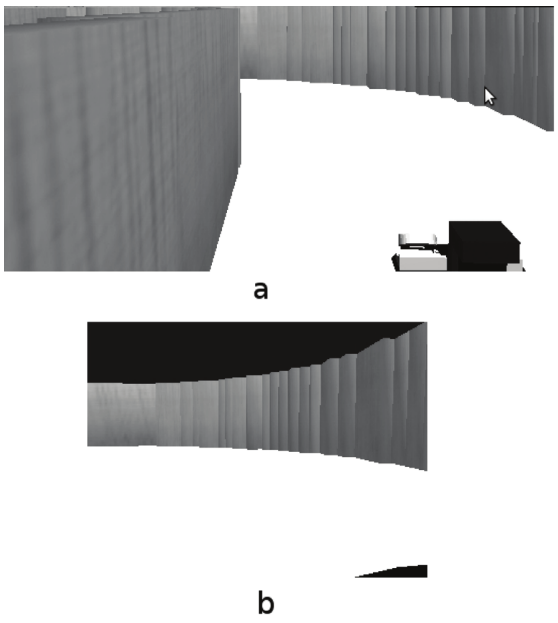


Figure 14. (a) Situation; (b) current onboard image



pairs of individuals 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, to avoid abrupt changes.

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 race selected will be the one with more victories and its pose will determine the calculated robot pose 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 will increase the victories of R_i and will 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 has been the selected race during a lot of iterations. With this behavior, we try to help the races who have been 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).

EXPERIMENTS

To verify our different approaches of visual memory, visual attention and visual localization, we conducted several experiments. Our 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 0.9 as robot simulator. All our experiments are implemented on C++ with Jderobot robotics software platform, which uses ICE as a middleware.

Attentive Visual Memory Experiments

1. **Robot in the middle of a room:** For the first experiment, the robot is in the middle of a room (see Figure 13-a). Then the robot turns around on itself. Figure 13-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 full circle, having stored all the information about their surrounding environment. Thus, we can see in Figure 13-c, how the short-term memory provides more information than an instantaneous image.

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 as shown in Figure 14-a, the robot approaches to a curve area, while navigating through a corridor. If robot used only instantaneous image (Figure 14-b), it would able to see only just some lines in front of itself (Figure 15-a), but with short-term memory it can observe that the path in front of itself is a curve (Figure 15-b). In addition, robot can quickly explore the surrounding environment thanks to the visual attention mechanism, which forces the system to explore unknown areas of the environment.
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.

The initial situation is showed in Figure 16-a. After a few seconds, robot has recovered environment information thanks to the short-term memory and the visual attention system (as showed in Figure 16-b).

Then another robot appears, as showed in Figure 17-a, occluding the field of view of our robot (Figure 17-b), so the robot is unable to see anything. This situation continues for some time while the second robot moves away from our robot (Figure 18-a,b).

This situation is solved by our system because of the persistence of the short-term memory, and the robot can make control decisions taking into account information of areas beyond the robot that is occluding its camera. 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 with the information stored in memory, the system has some persistence before changing the memory contents.

4. **Attentive visual memory on a humanoid robot:** In this experiment we have tested our whole system including visual memory, visual attention algorithm and visual localization on a real robotics platform such as Nao humanoid robot. The robot is in the middle of our department corridor recovering information about the environment around itself. 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 19 some snapshots and the short-term memory built with them.

Figure 15. (a) Information in current field of view; (b) short-term memory

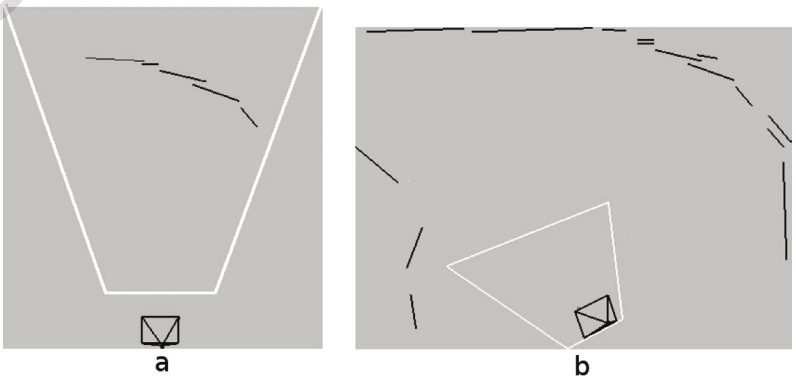
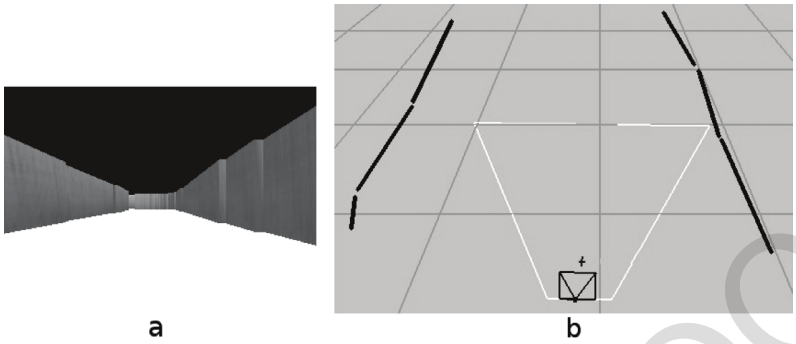


Figure 16. (a) Situation; (b) short-term memory got after a while



Visual Localization Experiments

We have performed several experiments to validate the evolutionary algorithm as well, especially with real robots. The algorithm has several parameters to be configured, such as the maximum number of races (10), the number of exploiters (30), and the percentages of the genetic operators previously explained (20% elitism, 20% crossover and 60% mutation). The maximum number of races has been 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.

1. **Typical execution in humanoid robot:** The first experiment has been performed with a real Nao robot travelling through a corridor (Figure 20). 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 20 shows the calculated positions, the green line the real robot path, and the brown area is the error measured. The average error has been 11,8 cm and 2,1 grades, the algorithm is able to follow the robot movement even when its instantaneous observations don't provide

Figure 17. (a) Situation; (b) field of view

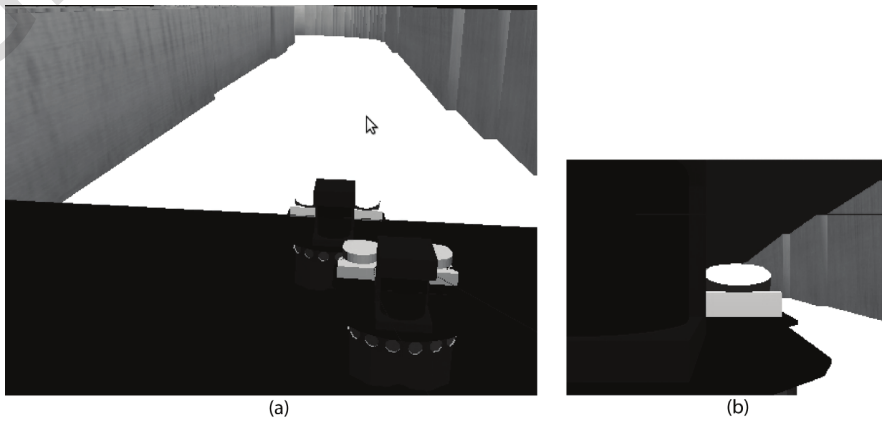
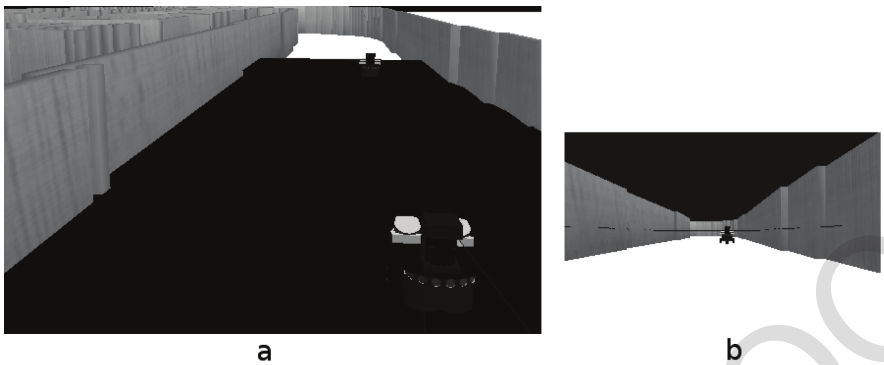


Figure 18. (a) Situation; (b) on board current image



enough information thanks 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.

2. **Dealing with symmetries and kidnappings:** The second experiment (Figure 21) 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 select 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 to the new right location. It 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 kidnap is performed (3.a), this case is similar to the first one, at first it selects a wrong localization (very close to the correct one, 3.b), but after some iterations it changes to the correct

Figure 19. Visual memory with 3D segments coming from four images of robot surroundings

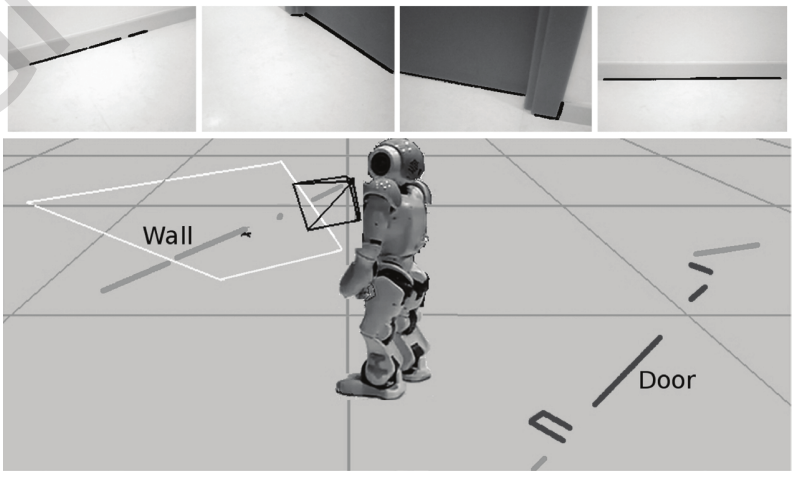
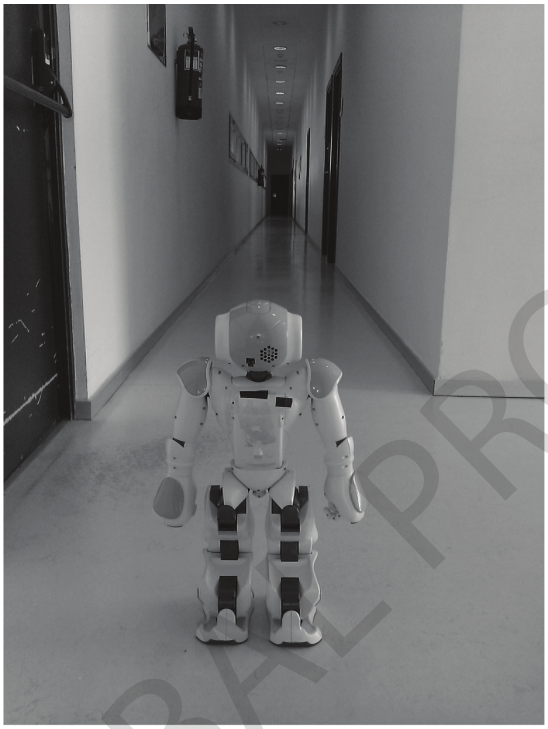
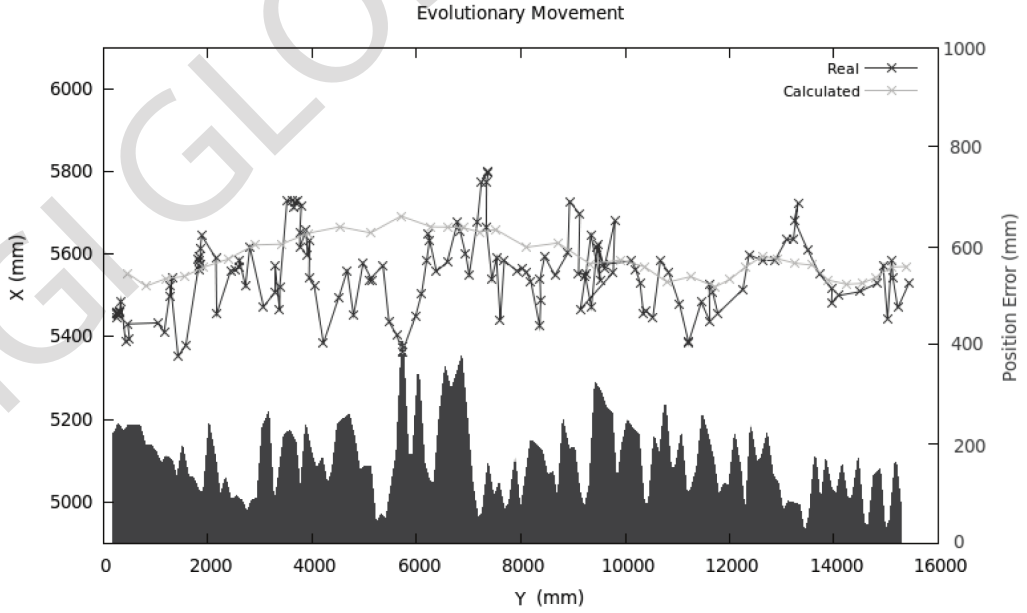


Figure 20. Nao robot travelling a corridor for experiments (left) and estimated localization and position error over time (right)



(a)



(b)

Figure 21. Position error over time (a) and angle error over time (b)

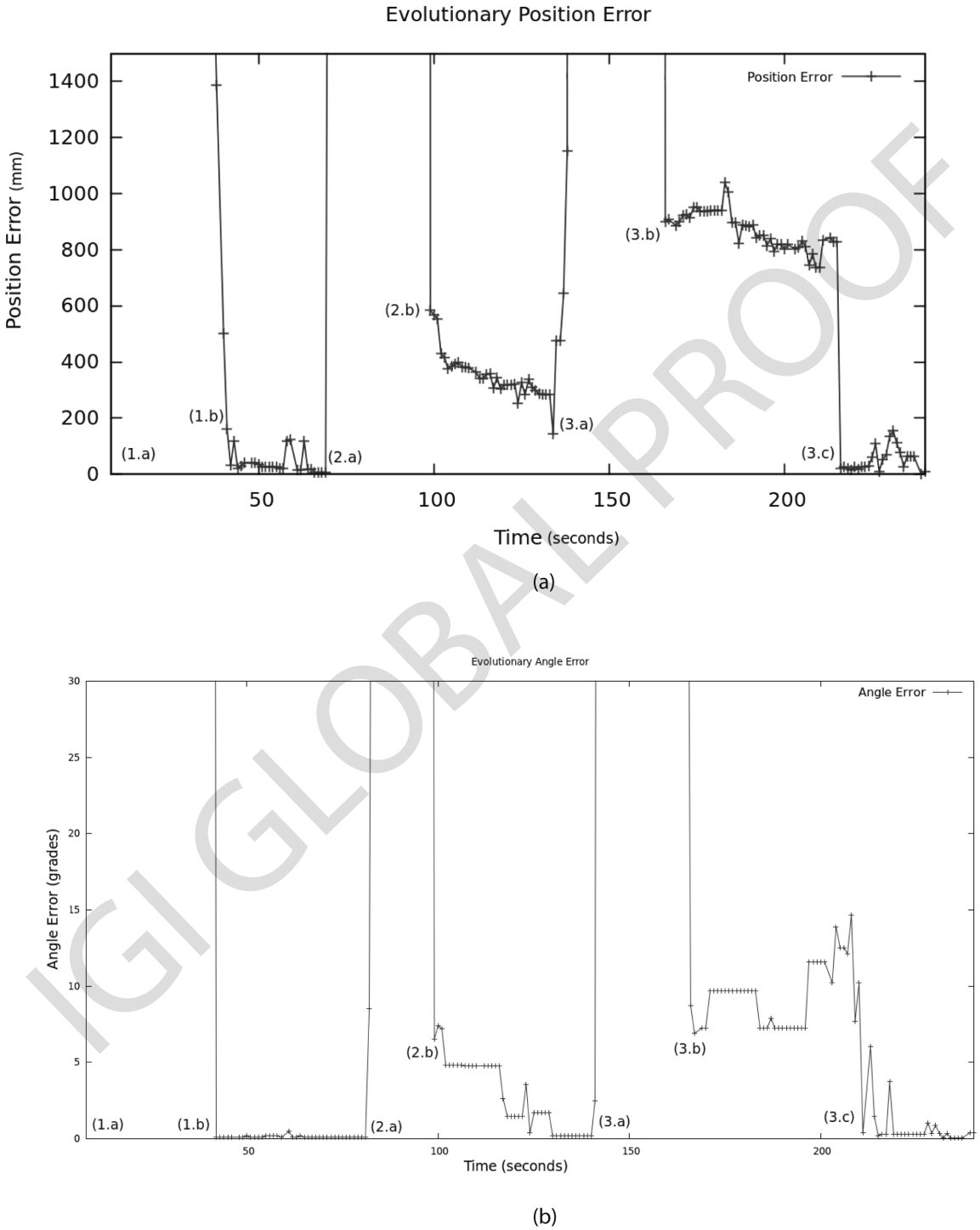
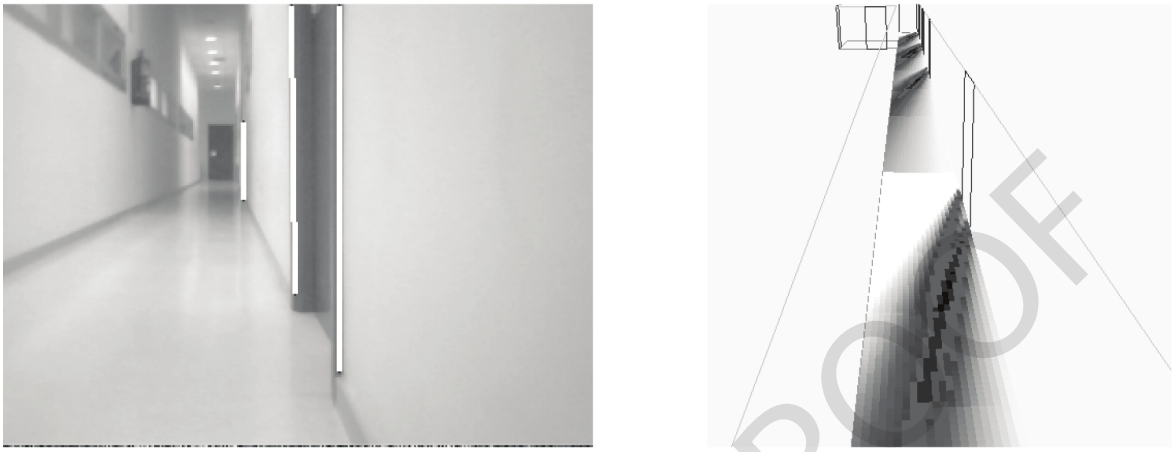


Figure 22. Image obtained (left) and probabilities calculated with θ equals to 0 radians(right)



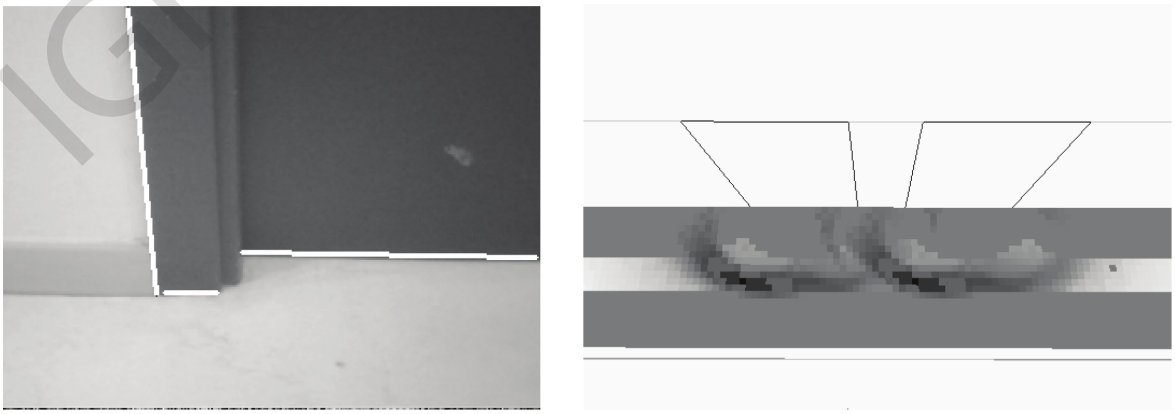
(a) (b)

one (3.c). The average error after selecting the correct race has been 14,9 cm and 3,2 degrees. We have also measured the time spent until the algorithm calculates a new plausible pose after a kidnapping (recovery time). In this experiment was 29,6 secs.

3. **Health function based on instantaneous images:** To validate the health function, we have implemented a debug mechanism to show graphically the value returned by the health function in different positions. In Figure 22 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 probabilities is plausible with the input image.

In case of symmetries we obtain high probabilities in several positions. In Figure 23 we show

Figure 23. Image obtained in front of a door (left) and probabilities calculated for any θ (right)



(a) (b)

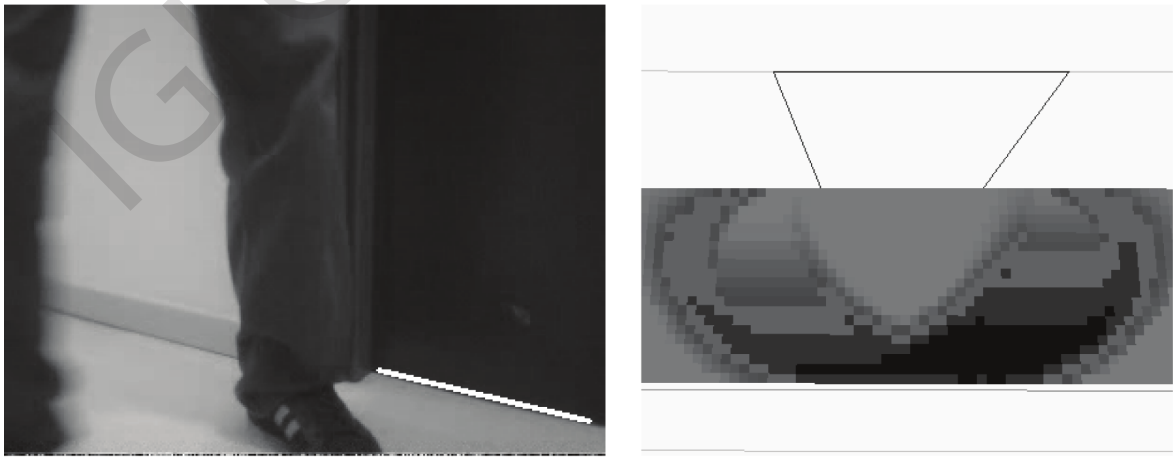
Figure 24. Image obtained (left) and probabilities calculated without occlusions or false positives for any θ (right)

(a) (b)

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 our environment:

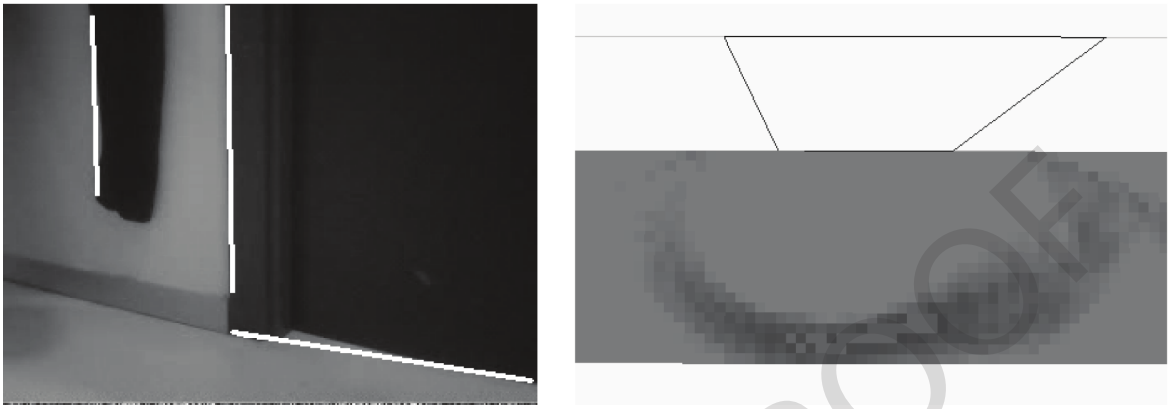
The localization algorithm has been tested in more complex scenarios, with occlusions and false positives. In case of occlusions the algorithm keeps a good behavior, since our approach doesn't penalize if an object is not detected in the image when it should be, and the only objection is a higher probability in some field areas (compare Figure 24 and Figure 25). However, false positives affect negatively to health function and the calculated localization can be totally wrong (compare Figure 24 and Figure 26).

Figure 25. Image obtained (left) and probabilities calculated with occlusions for any θ (right)



(a) (b)

Figure 26. Image obtained (left) and probabilities calculated with false positives for any θ (right)



(a) (b)

4. **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, since we will get more information about the environment, and temporal occlusions won't affect our health function. For instance, Figure 27 shows the health map and position estimation calculated with the visual memory of Figure 19.

CONCLUSION

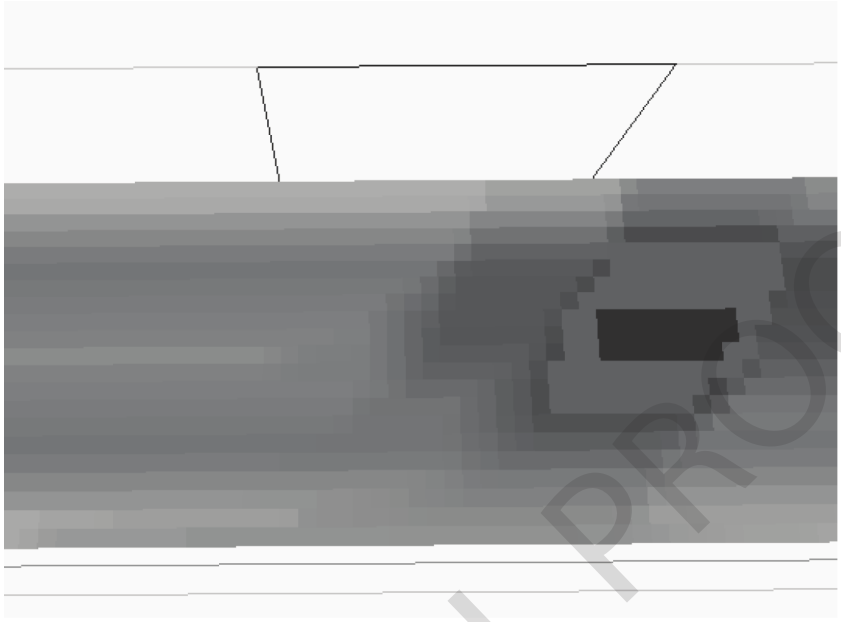
In this chapter a visual perception system for autonomous robots has been presented. It processes the images from a mobile camera and builds a short term local memory with information about the objects around the robot, even if they lie outside the current field of view of the camera. This visual memory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

An overt visual attention mechanism has been created to continuously select where the mobile camera should look at. Using a salience dynamics and choosing the most salient point the system shares the gaze control between the need to reobserve objects on the visual memory and the need explore new areas, providing also Inhibition of Return.

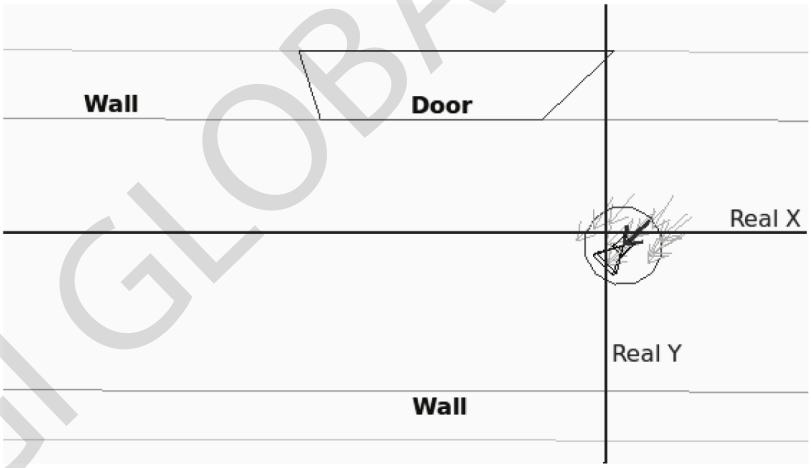
We developed a visual self-localization technique that uses an evolutionary algorithm. It keeps a population of particles to represent tentative robot positions and the particle set evolves as new visual information is gathered or with robot movements. It has been especially designed to bear symmetries, 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 has been validated both on real robots and in simulation. The memory nicely represents the robot surroundings using the images from the mobile camera, whose movement is controlled by our attention mechanism. The memory is dynamic but have some persistence to deal with temporary occlusions. The localization works in real time, provides position errors below 15cm and 5 degrees and is robust

Figure 27. Health values for any θ (left) and estimated position (right) calculated with visual memory



(a)



(b)

enough to recover from kidnappings or estimation errors in symmetric environments.

We are working in extending the system to stereo pairs and dealing with objects at any 3D position, not just the floor. We are also studying how to deal with more abstract objects like tables and chairs into the visual memory. Regarding localization we are working in introducing a monoSLAM EFK for each race of the evolutionary algorithm and improve them to extract localization information from abstract objects, not only 3D points.

REFERENCES

Arbel, T., & Ferrie, F. (2001). Entropy-based gaze planning. *Image and Vision Computing*, *19*(11), 779–786. doi:10.1016/S0262-8856(00)00103-7

Badal, S., Ravela, S., Draper, B., & Hanson, A. (1994). A practical obstacle detection and avoidance system. Proceedings of 2nd IEEE Workshop on Applications of Computer Vision, (pp. 97-104).

Ballard, D. H. (1991). Animate vision. *Artificial Intelligence*, *48*, 57–86. doi:10.1016/0004-3702(91)90080-4

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

Cañas, J. M., Martínez de la Casa, M., & González, T. (2008). Overt visual attention inside JDE control architecture. *International Journal of Intelligent Computing in Medical Sciences and Image Procesing, 2*(2), 93-100. ISSN: 1931-308X

Carrera, G., Angeli, A., & 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,* (pp. 77-82).

Dellaert, F., Burgard, W., Fox, D., & 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*, Vol. 2, (pp. 588-594). Fort Collins, CO: IEEE Computer Society.

Gartshore, R., Aguado, A., & Galambos, C. (2002). Incremental map buildig using an occupancy grid for an autonomous monocular robot. *Proceedings of Seventh International Conference on Control, Automation, Robotics and Vision ICARCV,* (pp. 613–618).

Goldberg, S. B., Maimone, M. W., & Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. *Proceedings of IEEE Aerospace Conference*, (pp. 2025–2036).

Hulse, M., McBride, S., & Lee, M. (2009). Implementing inhibition of return; embodied visual memory for robotic systems. *Proceedings of the 9th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems,* Lund University Cognitive Studies, Vol. 146, (pp. 189–190).

Itti, L., & Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews. Neuroscience*, *2*, 194–203. doi:10.1038/35058500

Itti, L., & Koch, C. (2005). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(11), 1254–1259. doi:10.1109/34.730558

Jensfelt, P., & Kristensen, S. (2001). Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, *17*(5), 748–760. doi:10.1109/70.964673

Mariottini, G. L., & Roumeliotis, S. I. (2011). Active vision-based robot localization and navigation in a visual memory. *Proceedings of International Conference on Robots and Automation.*

Marocco, D., & Floreano, D. (2002). Active vision and feature selection in evolutionary behavioral systems. *Proceedings of International Conference on Simulation of Adaptive Behavior* (SAB-7), (pp. 247-255).

Newcombe, R. A., & 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, (pp. 1498-1505). San Francisco, California (USA), June 2010.

Remazeilles, A., Chaumette, F., & Gros, P. (2006). 3D navigation based on a visual memory. *Proceedings of Robotics and Automation*.

Richard, H., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge University Press.

Solis, A., Nayak, A., Stojmenovic, M., & Zaguia, N. (2009). Robust line extraction based on repeated segment directions on image contours. *Proceedings of the IEEE Symposium on CISDA*.

Srinivasan, V., Thurrowgood, S., & Soccol, D. (2006). An optical system for guidance of terrain following in UAVs. *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance* (AVSS), (pp. 51–56).

Tsotsos, J. K., Culhane, S., Wai, W., Lai, Y., & Davis, N. (1995). Modeling visual attention via selective tuning. *Artificial Intelligence*, *78*, 507–545. doi:10.1016/0004-3702(95)00025-9

Zaharescu, A., Rothenstein, A. L., & Tsotsos, J. K. (2005). Towards a biologically plausible active visual search model. *Proceedings of International Workshop on Attention and Performance in Computational Vision* WAPCV-2004, (pp. 133-147).