Visual memory for robot navigation using JDE architecture *

José M. Cañas, Pedro M. Díaz, Pablo Barrera, Víctor M. Gómez

Dept. Ingeniería Telemática y Tec. Electrónica

Universidad Rey Juan Carlos

28933 Móstoles

jmplaza@gsyc.escet.urjc.es

Abstract

Vision is a very powerful sensor but it must be carefully integrated into the control architecture of the autonomous robots. First, it provides a huge amount of data which must be dealt with in real time. Second, some visual stimuli don't fit into a single image and require some kind of memory to fuse several images along time or from different points of view. In this paper a local navigation behavior is presented which is based on a short term visual memory. It has been designed using JDE architecture and implemented on a real robot. JDE decomposes the vision processing into a dynamic collection of schemas which builds visual stimuli. The stimulus required for safe wandering is a radial depth map, which is built fusing several images from a single camera and allows the identification of corridors.

1 Introduction

The use of cameras in robots is continuously growing. In the last years cameras have become a very cheap sensor (a simple webcam is cheaper than a regular sonar!) and they can potentially provide the robot with much information about its environment. But vision is much more complex than low bandwith sensors like laser rangefinders or sonars. Extracting relevant information from the pixel flow is not easy, neither processing the huge amount of data from the cameras in real time.

The image processing must be carefully inserted in the software architecture of the robots, and it must fit inside the perception approach of that architecture. Beyond simple reactive uses of vision, several researchers have worked on how to organize the visual processing inside autonomous robots [4, 8]. Visual attention and visual short term memory seem to be ingredients of intelligent organization of such processing and the robot architecture should provide mechanisms for them.

Visual attention is a useful tool when dealing with images and only limited computational resources are available. It offers a solution for the processing bottleneck generated by the huge amount of data carried by video streams. For instance, the *covert attention mechanism* [12] focuses in the areas of the images relevant for the task at hand, leaving out the rest.

Visual memory offers some advantages for behavior generation. First, it allows to represent objects in robot's surroundings beyond the limited field of view of the camera. This improves the quality of robot behavior as its control decisions may take more information into account. Some works use omnidirectional vision for that, and they have been succesfully applied to visual localization or soccer behaviors in RoboCup competition. Another proposal uses a regular camera, a short term visual memory and an *overt attention mechanism* [12], which allows for fast sampling of robot surroundings. Second, the visual mem-

*This work has been funded by Spanish Ministerio de Ciencia y Tecnología, under the project DPI2004-07993-C03-01 and Comunidad de Madrid under the project RoboCity 2030: S-0505/DPI/0176

ory allows to perceive complex stimuli, like those that do not lie or do not fit into a single camera snapshot.

In this paper we report a wandering navigation algorithm fully guided by vision. The algorithm uses a short term visual memory which is built from a single moving camera. No attention mechanism has been developed yet, the camera continuously sweeps horizontally. A radial depth map, which is wider than the camera field of view, is built and updated from the image flow. A complex stimuli, corridor, is searched for on that visual depth map.

One architectural approach close to this short term memory is the *Local Perceptual Space* in Saphira [9] where information coming from several distance sensors like sonars is fused altogether, and it allow to perceive complex stimuli like corridors, corners, etc.. In recent years a similar approach, named *visual sonar*, has been employed in the RoboCup domain [1] to enhance the navigation capabilities of the soccer players. They use only vision, as long as it is the main sensor in the Aibo player.

The rest of the paper is organized as follows. Second section describes the foundations of JDE architecture for robot applications as is the framework in which the visual navigation has been designed. Third section details the collection of schemas developed for the visual memory and the wandering navigation. Some experiments have been carried out on a real robot, they are summarized in fourth section. Some brief conclusions end the paper.

2 JDE: an architecture for robot applications

In JDE stance, behavior is considered the close combination of perception and actuation. Both are splitted in small units called *schemas* [11]. Nothing new so far. JDE proposes the schemas can be combined in a dynamic hierarchy to unfold the behavior repertoire of the robot accordingly to current goals and environment situation [7].

2.1 Schemas

JDE follows the Arkin's definition [11]: "a *schema* is the basic unit of behavior from which complex actions can be constructed; it consists of the knowledge of how to act or perceive as well as the computational process by which it is enacted". In JDE, each schema has a time scale and a state associated. They can be switched on and off at will and they accept some modulation parameters.

There are two types of schemas, perceptive and actuation ones. Each *perceptive schema* builds some information piece about the environment or the robot itself, which we will call a *stimulus*, and keeps it updated and grounded. That is its output. It can take as input the value of sensor readings, or even stimuli elaborated by other schemas. Perceptive schemas can be in SLEPT or in ACTIVE state.

Each *actuation schema* takes control decisions in order to achieve or maintain its own goals, taking into account the information gathered by the associated perceptive schemas. The outputs of an actuation schema are tipically commands to actuators, and can also be the activation of other schemas, both perceptive and actuation ones. Such new schemas are regarded as its children, and the parent schema often modulates them. Actuation schemas can be in several states: SLEPT, CHECKING, READY and ACTIVE, closely related to how action selection is solved in JDE. Control schemas have preconditions.

The algorithm running inside the schema contains all the task-knowledge about how to perceive relevant items or how to act, whatever technique be used. JDE architecture only imposes the interface: selective activation (on-off) and parameters for modulation, as long as it affects the way all the pieces are assembled together into the system. Concurrent continuous (iterative) execution is assumed, where each schema actively maps its inputs into its outputs.

2.2 Hierarchy

All awake schemas (CHECKING, READY and ACTIVE) run concurrently, similar to the distri-

bution found in behaviour-based systems. To avoid incoherent behaviour and contradictory commands to actuators JDE proposes hierarchical activation as the skeleton of the collection of schemas. It also claims that such hierarchical organization, in the ethological sense, provides many other advantages for roboticists like bounded complexity for action selection, action-perception coupling and distributed monitoring. All of them without losing the reactivity needed to face dynamic and uncertain environments.

In JDE there is hierarchy as long as one schema can activate other schemas. An actuation schema may command to actuators directly or may awake a set of new child schemas. These children will execute concurrently and they will in conjunction achieve the father's goal while pursuing their own. Actually, that's why the father awoke such schemas, and not others. A continuous competition between all the actuation siblings determines whether each child schema will finally get the ACTIVE state or remains silent in CHECKING or READY state. Only the winner, if any, passes to ACTIVE state and is allowed to send commands to the actuators or spring their own child schemas. The father activates the perceptive schemas that provide the information needed to solve the control competition between its actuation children and the information needed for them to work and take control decisions. This recursive activation of perceptive and actuation schemas conforms a schema hierarchy.

Once the father has awaken their children it keeps itself executing, continuously checking its own preconditions, monitoring the effects of its current kids, modulating them appropriately and keeping them awake, or maybe changing to other children if they can face better the new situation. So the hierarchy is dynamic.

2.3 Perception

Perception has traditionally been treated separatedly from action, but it is an essential part of behaviour generation systems, as it provides *all* the information on which an autonomous robot must make its control decisions (both for Action Selection Mechanism and for the ACTIVE schemas).

The active schemas may change with time, so the perception output is a dynamic collection of stimuli more than a general world model. In JDE perception is task oriented. Actually it is tightly coupled to action, as the parent schema awakes both, the actuation children and at the same time the perceptive schemas which build the relevant information.

Symbols and state are allowed in JDE. The activation of one perceptive schema indicates the internal predisposition to such stimulus, that some computing power is devoted to search for it and update its internal symbols, devoted to perceive it when it appear. An stimulus may appear in reality but if no internal process exists attending to it, the robot will miss it; The event has just not existed for the robot.

Continuous execution of perceptive schemas keeps their symbols grounded.

Hierarchy naturally offers a general attention mechanism, and so JDE perception is selective and situated. Only the data relevant to the current context are searched for, as only such perceptive schemas are ACTIVE. No computational resources are spent building stimuli not needed now, because the control schemas that need them and the associated perceptive schemas are SLEPT, which is the default state. In contrast, in Brooks subsumption architecture all the automatas are always active. Such all-time perception of all potential stimuli does not scale to complex and versatile systems, specially if vision is involved.

This selective perception in JDE is very convenient: when the whole system is going to exhibit a wide range of behaviors, the number of relevant stimuli of the environment which need to be internalized grows. Perception costs computing time, which is always limited. A general attention mechanism is clearly needed. Maybe not if the robot is going to do a single task, but it is essential if a full set of behaviors is going to be integrated into the same system.

Sensor fusion and complex stimuli may be generated in JDE as perceptive schema may activate other perceptive children and fuse

their outputs into a compound stimulus. All perception mechanisms of fusion, fission and fashion mentioned by Arkin [10] find a way inside JDE.

3 Visual memory for local navigation

The goal of this behavior is to wander through an indoor environment without colliding with any obstacle, static or dynamic, and using a single camera as the only sensor of the robot. The camera can be pointed at will using a pan-tilt device. Our solution has been designed as a collection of perceptive and actuation schemas, following JDE. In particular as the hierarchy of schemas at figure 1.

The perceptive `Frontera` schema extracts depth information from the monocular images and updates the *radial depth map* with it, which is part of the visual memory. The perceptive `Pasillo` schema searches in the radial map for long parallel segments which can be assumed to be a corridor. The actuation `Navegador` schema takes into account the obstacles in the radial map and the perceived corridor, if any, and uses a local navigation algorithm to command robot lineal and rotation speeds. The `Padre` schema is just the way to encapsulate the behavior in JDE, as another schema that launches its three children when activated. The `Guipolly` schema is an auxiliary thread in charge of the graphical interface of the application, just for debugging purposes.

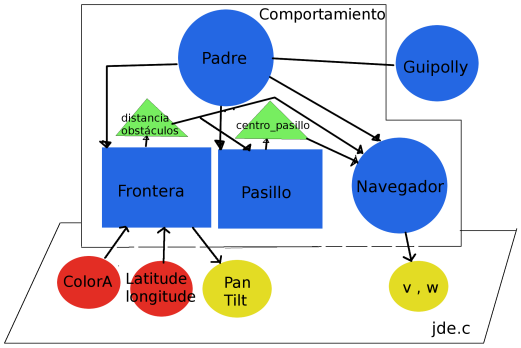

Figure 1: Schema design for the navigation behavior based on visual memory

3.1 Depth from camera images

The computation of depth from a monocular image is based on the ground-plane and the background-texture contraints presented on [3] and also followed by [2] and [1]. The underlying idea, as shown in figure 2, is to assume that the bottom part of the image matches with the floor and to explore each column in ascending pixel order. Wherever the first color discontinuity or edge appears in the column, that pixel is the projection of the 3D point where the closest obstacle to the robot departs from the floor.

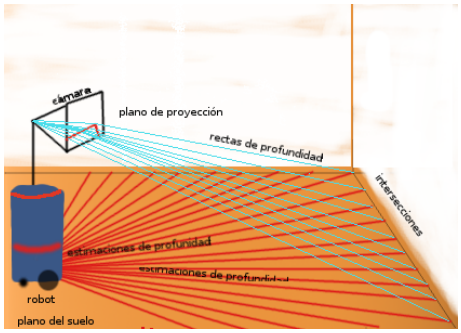

Figure 2: Depth from a monocular image

After doing some experiments with RGB and HSI filters, the edge filter performed better than any color one in our environment. The color image of the camera is displayed at the bottom left of figure 3 and the corresponding edge image at the bottom right. The depth estimation is shown, from bird eye, in the middle white canvas of the same figure. The upper horizontal segment in such canvas and its red points come from the edge image and match the corridor wall in the image, just in front of the robot.

3.2 Local scene composition

Every 3D point computed is inserted in the radial depth map, and closer points in the same orientation are cleared out. The radial depth map is a bounded array of points, each one with its absolute position. Such position is computed from the depth estimation and the location of the robot at the time of the image acquisition. The relative orientation and distance to the robot can be easily computed when needed, just substracting the absolute current location of the robot. Odometry is good enough for current robot location: as a recent odometric estimate was also used when computing the absolute position of the 3D points, absolute errors cancel out.

For every new image, maybe with the camera pointing in different orientation, the depth estimate is performed and the corresponding 3D points inserted in the radial depth map. The absolute coordinates help when fusing the information from different images.

To make the map wider than the field of view of the camera the `Frontera` schema also commands the pantilt unit to make periodic sweeps horizontally. This is no attention at all, as the movement is always the same, in an open loop, but expands the scope of the radial map.

A sample radial map can be shown in the middle of figure 3. Note that the two sides of the corridor are contained in the map, despite that from the current robot position only one of the walls appears in the camera.

3.3 Corridor stimulus

The `Pasillo` schema searches for corridors in the robot surroundings exploring the visual memory, which is composed of depth points. At each iteration this schema finds straight segments in the 3D points collection. It follows the fast segmentation algorithm described in [6], which makes iterative hypothetis about the segments and discards those that don't explain a minimum of consecutive points. The segments found for a sample radial map can be shown as black lines in the middle of figure 3.

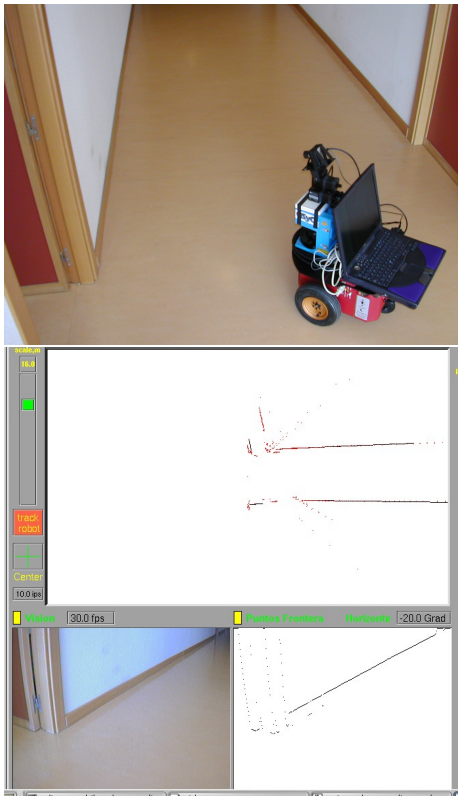

Figure 3: Robot facing to the wall, monocular and edge images, and visual memory

Once the segments have been computed, they are explored searching for parallel pairs. Those pairs of approximately parallel segments and located at a distance between 1 and 3 meters from each other are considered a valid corridor. The locations of the corridors are known and for the closest one to the robot, the further point in the middle of the corridor is computed with geometric operations. Such point will be the target for the local navigation until the next schema iteration, `CentroE` in figure 4. Given the way it is computed, there is free space from the robot towards it.

3.4 Wander navigation

The local navigation has been carried out using the Virtual Force Field (VFF) algorithm

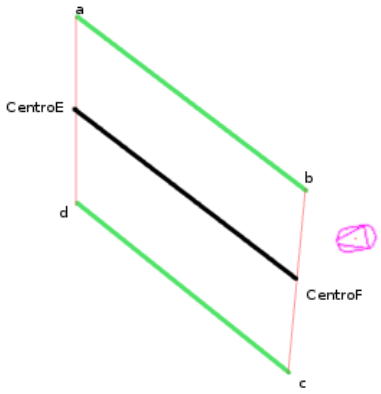

Figure 4: End point of the perceived corridor

[5]. In the VFF, the obstacles generate repulsive forces over the robot, and the navigation target generate attractive forces. For our purposes the obstacles are the depth 3D points computed by `Frontera` schema and stored in the visual short term memory. The navigation target is the central point of the corridor as detected by the `Pasillo` schema. The final force is the simple addition of all repulsive and attractive forces (Figure 5), and so, it offers a good balance between obstacle avoidance and progress towards the goal.

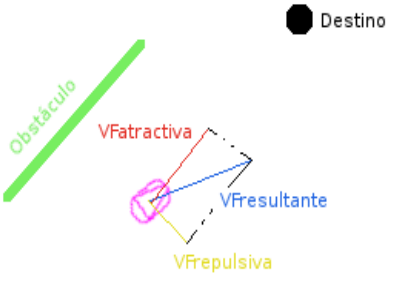

Figure 5: Virtual Force Field algorithm

Once the final force is computed, it must be translated into speed commands: lineal speed (v) and angular speed (w), which are the only orders the robot understand. We have used some simple rules to that.

4 Experiments

The described set of schemas for robot wandering has been programmed and tested under different conditions on a Pioneer robot (that on Figure 3) inside our university building[1]. The robot is endowed with a regular laptop (1.5 GHz Pentium), a pantilt unit (PTU-46-17.5 from Directed Perception) and a single firewire camera on top (DCAM-L from Videre Design). The camera is slightly inclined downwards, looking at the floor, in the experimental setup.

The application software has been developed inside *jde.c* platform, an implementation of JDE architecture in C language. *jde.c* offers the sensor measurements (including camera images) as perceptive variables that the control program reads, and it offers the actuator orders as shared variables that the control program writes.

Several segmentation and navigation tests were performed in the Player/Stage simulator over laser depth data before porting the algorithm to the real depth data from monocular vision.

In the figure 6 the robot is inside a rectangular area with a green floor. The current color image, the last edge image and the radial depth map computed by `Frontera` schema can be seen at the bottom area of such figure.

In the experiments the robot successfully wandered along our university corridors and halls, turned at the corners and avoided unexpected obstacles. It also successfully identified free corridors. The radial map update and the search for corridors are continuously performed in real time (for instance `Pasillo` schema runs at 10 iterations per second), and so, the algorithm quickly reacts to unexpected obstacles and recomputes its pathway.

[1]Some videos of the experiments are available at the web, http://gsyc.escet.urjc.es/jmplaza/research-vision.html

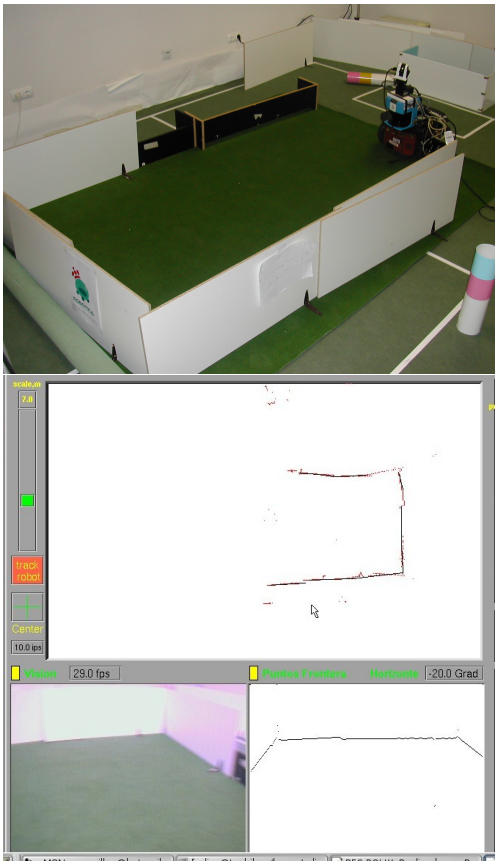

Figure 6: First image incorporated into visual memory

In figure 7 the radial map after incorporating five images is displayed. Note that the application code is the exactly same that in figure 6 but this time the floor is light brown colored. The fake cul-de-sac of the corridor is due to the fact that when no edge is detected in the image column, the upper pixel is considered as edge and that gives a cautious depth computation.

The pantilt movement is composed of a sequence of several stop & go points along the horizontal plane, approximately covering 180 degrees. The camera may get blurry images while the pantilt is moving from one point to another. To avoid processing such frames, the

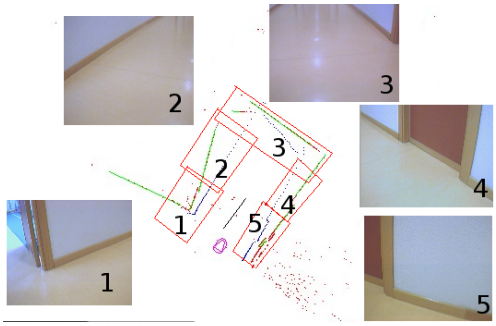

Figure 7: Short term memory after several images

`Frontera` schema waits until the pantilt has stopped at its current sweep position.

5 Conclusions

One solution for the wandering behavior has been described, as a proof of concept of the short term visual memory. The system was implemented and tested on a real robot. Only one camera has been used to unfold the behavior, no distance sensor. The application has been designed as a collection of concurrent schemas inside the JDE architecture. One perceptive schema extracts obstacle distances from the image flow using the *floor hypothesis* and integrates them into a radial depth map. Another perceptive schema identifies corridors on such depth map. They provide information for the actuation schema, which uses VFF algorithm to control the robot.
The visual memory has allowed the identification of complex stimulus that are not included in a single snapshot of the camera, like the radial depth map and the corridor stimulus. It also allows better navigation performance than the pure reactive algorithm. First, no oscillations was observed, which is typical of VFF systems directly mounted on sensor information. Second, the wider scope of the obstacle information gives better control decisions.

There are two future lines we are working on. First, to use this visual memory within an Aibo robot in the RoboCup environment. The green color of the field is a good feature

there to detect the obstacles with the monocular camera. Second, to use some overt attention mechanism to select the next point to look at, in order to insert that area into the visual memory.

References

[1] Lenser, S. and Veloso, M. *Visual sonar: fast obstacle avoidance using monocular vision*, Proceedings of 2003 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp 886–891, 2003.

[2] Ulrich, I. and Nourbakhsh, I., *Appearance-based obstacle detection with monocular color vision*, Proceedings of 2000 AAAI Nat. Conf. on Artificial Intelligenge, pp 866–871, 2000.

[3] Horswill, I., *Visual collision avoidance by segmentation*, Proceedings of the 1994 IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems, pp 902–909, 1994.

[4] Horswill, I., *Visual architecture and cognitive architecture*, Journal of Experimental and Theoretical Artificial Intelligence 9(2-3):277–292, 1997.

[5] Borenstein, J. and Koren, Y., *The vector field histogram fast obstacle avoidance for mobile robots*, IEEE Journal of Robotics and Automation, 7(3):278–288, 1991.

[6] Gómez V.M., Cañas, J.M., San Martín, F. and Matellán V., *Vision based schemas for an autonomous robotic soccer player*, Proceedings of IV Workshop de Agentes Físicos WAF-2003, pp 109-120, Universidad de Alicante (Spain), March 2003.

[7] Cañas, J.M., *Jerarquía Dinámica de Esquemas para la generación de comportamiento autónomo*, PhD Thesis, Universidad Politécnica de Madrid, 2003.

[8] Wasson G., Kortenkamp D., Huber E. *Integrating active perception with an autonomous robot architecture*, Robotics and Autonomous Systems, 29:175-186, 1999

[9] Konolige, K. and Myers, K., *The Saphira architecture for autonomous mobile robots*, In *Artificial Intelligence and mobile robots: case studies of succesful robot systems*, pp 211–242, edited by David Kortenkamp, R.Peter Bonasso and Robin Murphy, The MIT Press, 1998.

[10] Arkin, R., *Behavior Based Robotics*, The MIT Press, 1998.

[11] Arkin, R., *Motor schema-based mobile robot navigation*, Int. Journal of Robotics Research 8(4):92-112, 1989.

[12] Itti, L., Koch, C.; *Computational Modelling of Visual Attention*, Nature Reviews Neuroscience 2:194-203, 2001