14

Biologically-Plausible Reactive Control of Mobile Robots

Zapata René, Lépinay Pascal
Université Montpellier II
France

1. Introduction

In many robotic tasks, a mobile robot has to perform four subtasks constituting a hierarchy from high level to low level actions:

- Path planning
- Localization
- Target pursuit
- Collision avoidance

The last 3 tasks which obviously need perception capabilities and vision (like for natural beings) can provide the robot with very rich information. Unfortunately, artificial vision algorithms present the major disadvantage of being difficult to process in real time (for instance, the collision avoidance process typical time is less than 0.01s). Therefore including real time vision in a robot control loop system requires hardware implementation. But efficient reactive behaviours can also be performed with simpler sensors, like range sensors (Ultrasonic sensors for instance).
Biologically-plausible algorithms spring from the observed natural systems especially through analogies between artificial perception/actions systems and natural ones (Berthoz and Weiss, 2000). Many papers deal with this subject and whole conferences are devoted to it (see for instance SAB proceedings (SAB, 2003)). Neural network theories are an important part of this research and much work has been done on neural aspects of artificial vision. For instance, two recent works are closely correlated with biological experiments (Wurtz and Lourens, 2000) (VanRullen, 2002).
This chapter is divided into 4 parts. The first section describes a reactive control algorithm based on the modelling of the robot/environment interaction. Like Potential fields methods (Kathib 1985) or (Holenstein and Badreddin, 1991), this approach can present the problem of local minima. However, it turns out hat it is quite robust to unknown and dynamic worlds. The second section (section 3) describes two vision-based algorithms: 3D reconstruction and image segmentation, which will be the basis of the implementations proposed in the last section of the chapter. Section 4 presents several simulation experiments illustrating the two previous issues. Section 5 describes the implementation of fast vision-based algorithms on configurable architectures.

Source: Mobile Robots, Moving Intelligence, ISBN: 3-86611-284-X, Edited by Jonas Buchli, pp. 576, ARS/plV, Germany, December 2006

2. A bio-plausible reactive control algorithm

his section describes a bio-plausible reactive control algorithm for obstacle avoidance and target pursuit in an unstructured and dynamic world. This algorithm was first described in (Zapata et al., 1994) for many applications and tested for robots moving in 2 dimensions, flying robots or autonomous submarines and also mobile manipulators (Cacitti, 2000). This algorithm that was initially designed for obstacle avoidance, has two main advantages. First, the environment does not need to be *a priori* known, and second, the controller can take into account other constraints such as target pursuit, altitude maintaining and course control.

2.1 General principle

The reactive control algorithm we use is based on the definition of a protecting and deformable zone surrounding the robot. This DVZ (*Deformable Virtual Zone*) is parameterized by the motion variables of the moving robot and can deform in the presence of distance information in the robot workspace. When an obstacle enters the sensor space, it induces a deformation of the DVZ that will be compensated by the robot motion controller. Therefore, the algorithm is a kind of 2-player game: the first one, i.e. the environment, induces undesired deformations; the second one, i.e. the robot controller, tries to rebuild the DVZ.
Figure 1 illustrates this general principle that will be described in paragraph 2.2.

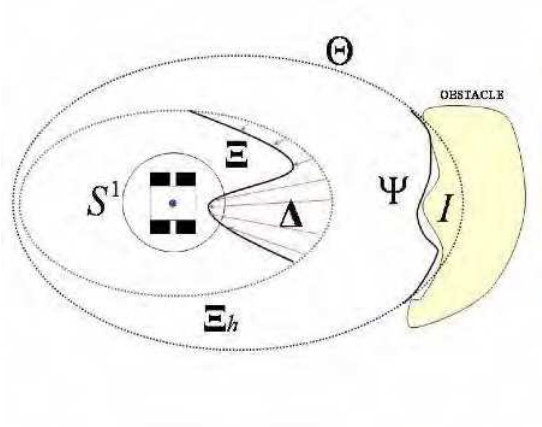


Fig. 1. The DVZ principle in 2D.

2.2 Mathematical basis

The general framework for formalizing this principle is the category *D* of topologically equivalent sets of the (*n-1*)-dimensional unitary sphere $S^{n-1}(0,1)\text{in }\mathbb{R}^n$. An object *A* of this category is related to the unitary sphere through an homeomorphism (imbedding of the sphere):

$$\delta_A : S^{n-1}(0,1) \rightarrow A \subset \mathbb{R}^n \tag{1}$$

The transformations between two objects *A* and *B* of *D* are the deformations obtained by the combination of the two defining homeomorphisms:

$$\delta_B \circ \delta_A^{-1} : A \to B \tag{2}$$

Let $R \subset \mathbb{R}^n$ be a convex rigid body, subset of the *n* dimensional-space $\mathbb{R}^n$. The boundary ∂R of R can also be considered as the result of an imbedding of $S^{n-1}(0,1)$ in $\mathbb{R}^n$. We have $\partial R \in D$ and:

$$\delta_R : S^{n-1}(0,1) \to \partial R \subset \mathbb{R}^n \tag{3}$$

Reciprocally, if *f* is an imbedding of ∂R in $\mathbb{R}^n$ with $E = f(\partial R)$, it is to say that:

$$f : \partial R \to E \subset \mathbb{R}^n \tag{4}$$

then : $E \in D$ by the choice: $\delta_f = f \circ \delta_R$.
Any object $A \in D$ separates $\mathbb{R}^n$ in two connected components, the interior *Int(A)* of *A* and the exterior *Ext(A)* of *A*. Therefore we have $\mathbb{R}^n = Int(A) \oplus Ext(A) \oplus A$. A partial order is induced on *D* by the relation:

$$A \prec B \Leftrightarrow Int(A) \subset Int(B) \tag{5}$$

The rigid body *R* will represent a controlled robot moving among obstacles in $\mathbb{R}^n$. Any *n*-dimensional state vector characterizing the motion of *R* (e.g.translational and rotational velocities) is denoted as a vector:

$$\pi = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix}^T \tag{6}$$

In the following, we will assume that the robot *R* can be controlled by the derivative of this state vector. We note:

$$\phi = \dot{\pi} \tag{7}$$

We define a *DVZ* of R as any imbedding Ξ of ∂R in $\mathbb{R}^n$ such that the relation $\partial R \prec \Xi(\partial R)$ holds. We have $\Xi(\partial R) \subset D$.
We define a *controlled DVZ*, Ξ_h as a DVZ which depends on the state vector characterizing the motion of *R* :

$$\Xi_h = \rho(\pi) \tag{8}$$

Let $P = (\Xi_h, \Xi)$ be a pair of two DVZ of *R* (the first one being a controlled DVZ) and such that $\Xi(\partial R) \prec \Xi_h(\partial R)$. We define the deformation Δ of the DVZ Ξ_h with respect to Ξ as the functional difference of Ξ and Ξ_h:

$$\Delta = \Xi - \Xi_h \tag{9}$$

According to this definition, the deformation Δ is a one-one map that associates the vector $P - P_h$ to the point $M \in \partial R$, where $P = \Xi(M)$ and $P_h = \Xi_h(M)$. It can therefore be considered as a vector field defined on ∂R.
We also assume that the robot can perceive distances in all directions of space and that the set of maximum distances that can be perceived by *R* and the set of actually perceived

distances are two objects of the category *D*, respectively named *Sensor boundary* and *Information boundary* (respectively denoted Θ and Ψ), such that $\Psi \prec \Theta$. The deformation *I* of Θ with respect to Ψ is given by:

$$I = \Psi - \Theta \tag{10}$$

The deformation *I* can also be considered as a vector field on ∂R.
We define an *uncontrolled DVZ*, Ξ, as a DVZ which depends on the Sensor boundary deformation *I*:

$$\Xi = \beta(I) \tag{11}$$

Let $P = (\Xi_h, \Xi)$ be a pair composed of a controlled DVZ and an uncontrolled DVZ, the deformation Δ of the DVZ Ξ_h with respect to Ξ can be written:

$$\Delta = \Xi - \Xi_h = \beta(I) - \rho(\pi) \tag{12}$$

For a given point $M \in \partial R$, the deformation vector $\Delta(M)$ depends on the intrusion of proximity information *I(M),* in the rigid body workspace, and on the controlled DVZ Ξ_h.
By differentiating equations 12 with respect to time, we get:

$$\dot{\Delta} = -\beta'(I)\psi - \rho'(\pi)\phi \tag{13}$$

where f' represents the derivatve of function f .This equation can be rewritten as:

$$\dot{\Delta} = A\phi + B\psi \tag{14}$$

Variations in Δ are controlled by 2 input vectors ϕ (due to the robot controller, tends to minimize deformation of the DVZ) and Ψ (unknown and induced by the environment itself).
In order to control the reactive behaviours of the robot, the desired variation of this deformation is chosen as a function of the real deformation and its derivative:

$$\dot{\Delta}_{des} = -K_{prop}\Delta - K_{der}\dot{\Delta} \tag{15}$$

where K_{prop} and K_{der} are heuristically chosen. The computation of the best control vector $\bar{\phi}$ at time *t* obtained by inverting equation 14 after replacing the deformation derivative by its desired value $\dot{\Delta}_{des}$:

$$\bar{\phi} = A^{+}\left(\dot{\Delta}_{des} - B\bar{\psi}\right) \tag{16}$$

where A^{+} is the pseudo-inverse of the linear function *A* and $\bar{\psi}$ is an estimation of the second control vector Ψ at time *t* obtained at time *t-1:*

$$B\bar{\psi} = \dot{\Delta}_{measured}(t-1) - A\phi(t-1) \tag{17}$$

This control law is illustrated in figure 2.

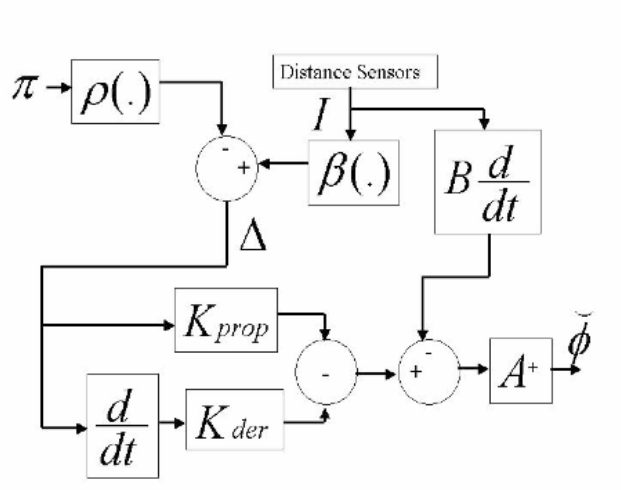


Fig. 2. The DVZ control loop.

Important remarks:

- The control law (Equation 16) tends to minimize the function $\left\| \dot{\Delta}_{des} - \Delta \right\|$ in the Least Squares sense.
- The ∞-dimensional functional equation 14 cannot, of course, be used directly. It is necessary to sample the sensor space in order to obtain an *n*-dimensional definition of the DVZ. This can be done by considering that the information vector has *n* dimensions (as many as the number of distance sensors). Equation 14 keeps its general form but all its entries are now matrices or vectors. For instance, let's consider the simple case of a car-like robot equipped with 32 distance sensors. In this case, the functions I, Δ, $\dot{\Delta}$, $\dot{\Delta}_{des}$, Ξ_h and Ξ are 32-dimensional vectors. The vector π is a 2-dimensional vector and function A is a (32×2)-*dimensional* matrix. The matrices K_{prop} and K_{der} are (32×32) matrices.

2.3 Implementation

Algorithm

The reactive control is implemented as follows:

(i) Measurement of the intrusion of information *I* as an *n*-dimensional vector (as many dimensions as the number of sensors)
(ii) Derivation of the deformation Δ and of its derivative $\dot{\Delta}$
(iii) Estimation of the uncontrolled control vector $\hat{\psi}$
(iv) Computation of the best control vector $\breve{\phi}$

Regarding the target pursuit algorithm in the presence of obstacles, we have the same algorithm except that in this case the deformation vector includes lines of vector components for measuring the deformation between a desired state (robot on target for instance) and the real state (robot at a measured distance from the target): $\begin{bmatrix} \Delta_{\text{avoid}} \\ \Delta_{\text{target}} \end{bmatrix}$

Practical implementation of the function β

In the rest of the paper, we will assume that the function β relating the intrusion of information to the deformed DVZ Ξ_I is defined as follows:

$$\beta(I) = \begin{cases} \Theta + I \text{ if } \Theta + I < \Xi_h \\ 0 \text{ otherwise} \end{cases} \tag{17}$$

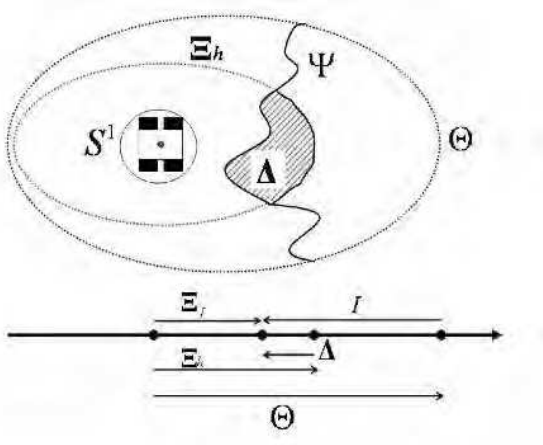


Fig. 3. Computation of function β

where the sign < means the comparison of the sensor information $\Psi = \Theta + I$ and the undeformed DVZ Ξ_h in each direction of measure (see figure 3). This restriction does not change the principle of the DVZ and is just a simplification leading to the confusion of the intrusion *I* and the deformation Δ (their derivatives are now identical)

3. Vision feedback

In this section, we will discuss two vision-based algorithms: 3D reconstruction and image segmentation, which will be the basis of the implementations proposed in the last section of the paper. For these two algorithms, it is obvious that vision can be helpful way for generating the necessary information.

3.1 Vision feedback for collision avoidance

For the collision avoidance scheme, the main input is the distance field in the robot front space. Stereovision allows this 3D reconstruction by measuring the disparity field in two images.

A very good review of advances in stereovision can be found in (Brown et al., 2003). The authors address three basic topics: correspondence methods (local and global), occlusion problems and real-time implementations. This paper lists more than thirteen references in this last field from 1993 to present. The technologies used range from single off-the-shelf processors to custom FPGAs (Faugeras and al, 1993), (Kimura et al., 1999) and DSPs (Beymer and Konolige, 1999).

The vision-based stereo-matching algorithm for distance estimation and collision avoidance involves spatial stereo-matching of *n* characteristic points of two images (left and right) and provides a distance field in the robot front-space. This distance field induces the DVZ. The complete algorithm has 7 steps:

(i) right The right image is divided into regular zones in order to cover the whole front space of the mobile robot. A particular zone of this image called *fovea*, is subdivided into 16 smaller zones in order to increase the perception capabilities of the vision system in a given direction like in biological systems

(ii) A set of interesting points (e.g. high weighted gradient) is chosen in the right image. A subset of characteristic points is derived from this set, containing one point per zone. Each of these points has the strongest gradient in the corresponding zone. Mathematically, this set is characterized by the positions of these characteristic points in the image space (pixels)

(iii) A stereo-matching algorithm, based on the evaluation of a distance between a (11x11) patch around the characteristic points and a (11x11) patch moving in the left image, tries to find the correspondences between these 2 images. A counter correlation is then run to confirm the stereo-pairs. The disparity between the characteristic points provides a distance map (vector *I*) to the robot main computer (figure 4)

(iv) The controlled DVZ $\Xi_h(t) = \rho(\breve{\pi}, t)$ is computed

(v) The complete DVZ $\Xi(t) = \Xi_h(t) + \Delta(t)$ is computed

(vi) The control vector $\breve{\phi}$ is computed

(vii) The DVZ orientation induces re-computing of the position of the fovea in the image for time *t+1*

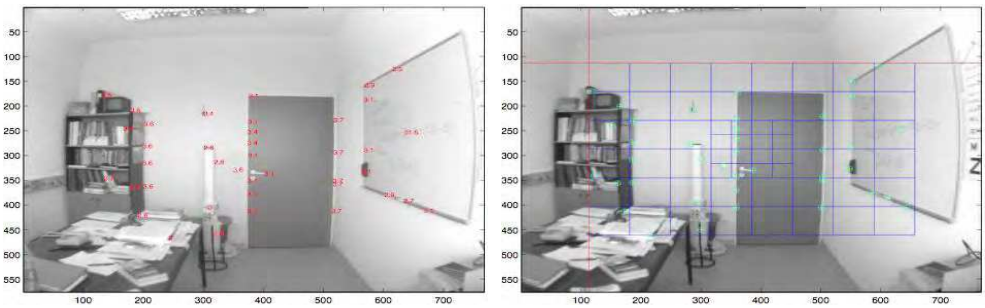


Fig. 4. Left and right images and the distance field on the left image.

3.2 Vision feedback for target pursuit

For the target pursuit scheme, the visual recognition of the target can also involve a visual system. To recognize, or at least to follow, a given pattern in the robot space is a crucial issue in mobile robotics. For instance, in target pursuit tasks, the target has a very special role and its localization, and the measurement of its velocity or its acceleration is essential. Segmentation of a colour image can provide this information. In the other ego-localization example (for path planning purposes), clues in the robot environment can also be detected by colour segmentation. Even for obstacle detection, image segmentation can help in the determination of characteristic points before 3D reconstruction.

We have developed a segmentation algorithm based on two-layer Probabilistic Neural Networks (PNN) and on Mathematical Morphology tools. As summarized here, this algorithm is a 2-step process:

- *Segmentation*: Each neuron of the first layer of the PNN is dedicated to a given user based colour. Its weight *W* is a point in the RGB space. Each *colour-prototype*, is therefore represented by several neurons in the network. When a pixel is read, its distance to *W* is computed and its probability of belonging to the neuron class is computed through a normalized Gaussian function. Then its probability of belonging to a given colour-prototype is computed by a combination of all neural probabilities (as many as the number of neurons of the first layer). The next step carried out by the second PNN layer is the competition of all colour-prototype probabilities. At the end of the segmentation process, the original image has been replaced by a labelled image consisting of as many labels as there are different prototypes.
- *Morpho-Filtering:* Each theme can be transformed in a black and white image (v=1 for a pixel belonging to the prototype, v=0 if not). In order to clean this image, it is then eroded by an erosion process and dilated by a dilatation process. In a second step, the resulting image is labelled in order to detect all objects belonging to the theme.

4. Simulation results

4.1 Collision avoidance and target pursuit

A graphic simulator was developed with MATLAB for experimenting the reactive collision avoidance process based on the DVZ algorithm from either a computational and behavioral point of view. The simulator constitutes an interface that allows us to interactively change the position of the obstacles, the intrinsic parameters of the DVZ, and initial states. It permits as well to visualize the 2-dimensional motion of the robot and the geometry of the DVZ.

The mobile robot presented here is a car-like vehicle equipped with 32 simulated ultrasonic sensors. Next figures show a few results provided by this simulator. In all these cases, the target is virtual: the robot must maintain a constant velocity and a zero-angle for its front wheels

- Left part of figure 5 compares the trajectories between a primary action (Trying to keep [wheel direction=0] and [velocity=5m.s^{-1}]) without collision avoidance versus the same task with collision avoidance of a wall.
- In right part of figure 5 the mobile robot avoids a frontal wall that should have induced a singularity of the DVZ. The noise in sensors avoids this symmetry. To carry out this task, we defined the primary task as before (Keeping [wheel direction=0] and [velocity=4m.s^{-1}]) and turned on the collision avoidance task.
- In left part of figure 6 the mobile robot follows a corridor. To carry out this task, we defined the primary task as before (Keeping [wheel direction=0] and [velocity=3m.s^{-1}]) and turned on the collision avoidance task.
- In right part of figure 6 the mobile robot runs in a complex world up to 4m.s^{-1}.To carry out this task, we defined the primary task as before (Keeping [wheel direction=0] and [velocity=5m.s^{-1}]) and turned on the collision avoidance task.

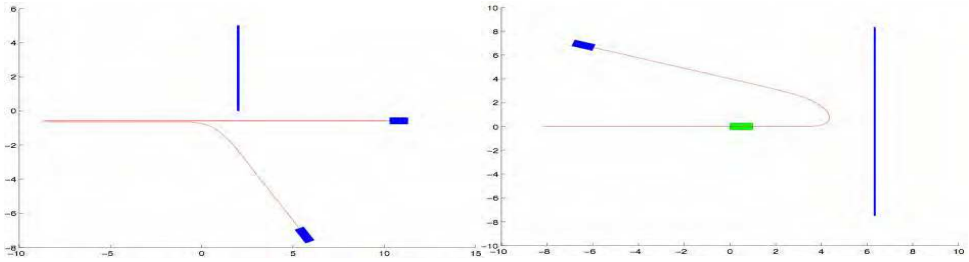


Fig. 5. Avoiding a wall.

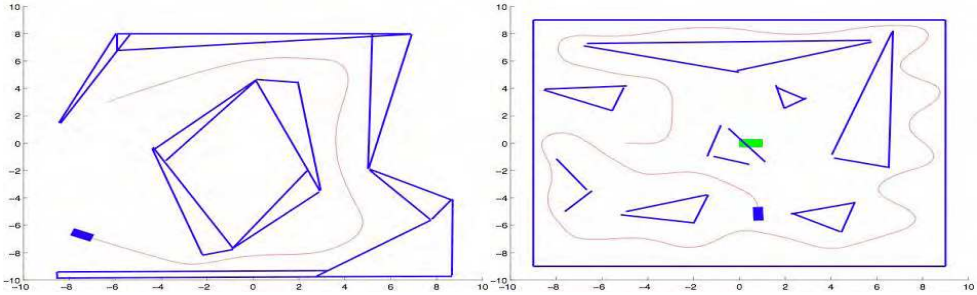
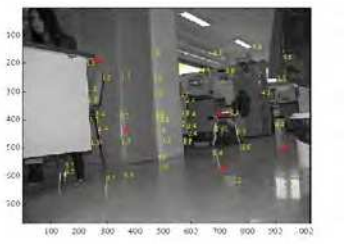


Fig. 6. Moving in a corridor and in a clustered environment.

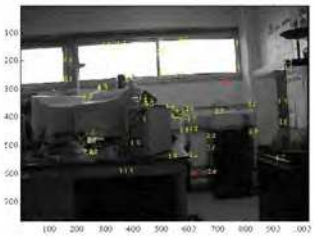
4.2 Vision feedback for collision avoidance

We tested the spatial stereo-matching algorithm on several pairs of images taken in unknown environments (rooms, corridors, outdoor). The algorithm divides each image into 60 zones (44 for global vision +16 for the mobile fovea) and chooses one point of interest per zone (when there is one). We carried out several experiments with different pairs of images and noticed that the number of characteristic points was not constant, due to the existence of uninteresting zones in the image (no significant gradient). For interesting points (with a strong gradient), if the stereo-matching algorithm reveals a good correlation between the left and right images, the distance of the point is computed (with a pin-hole camera model), and therefore its position in the real robot space is known. Otherwise, this distance is set at infinity. A visual check of the results indicated a success percentage from 90 to 100 percent (Figure 7):

- Number of points: 56
- Number of matching: 51
- Percentage of success: 91%



- Number of points: 46
- Number of matching: 48
- Percentage of success: 95.8%



- Number of points : 37
- Number of matching: 36
- Percentage of success : 97%

Fig. 7. Distance fields in gray level images.

4.2 Vision feedback for target pursuit

We also tested the image segmentation algorithm on several colour images. A simulator was developed for designing different PNNs that are run on colour images. The user decides how many colour-prototypes are included in the network and how many prototypes (neurons) for each colour-prototype. This structure allows testing of several classifiers, depending on the chosen prototype. For instance, it is possible to create a PNN to separate dark objects from light objects (2 prototypes), or to build one to detect a red object among others, and so on.

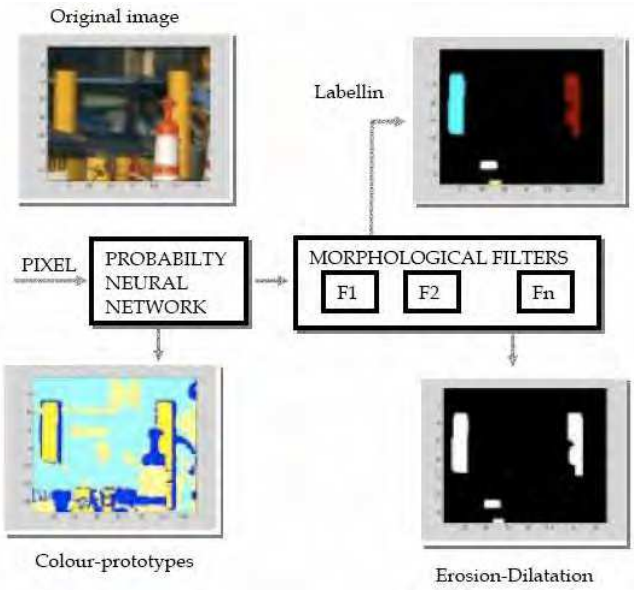


Fig. 8. Segmentation of regions.

Figure 8 examplifies the segmentation process for the detection of yellow segments in an image. Here, the PNN has 8 colour-prototypes (Red, Blue, Green, Yellow, Magenta, Cyan, Dark and Light). At the end of the neural segmentation process, each original pixel has been transformed in a colour-prototype representing number. The whole image now looks like a pseudo-colour image. Then, the morphological process can be applied to any of these prototypes.

5. Implementation

5.1 Collision avoidance and target pursuit

This algorithm was first described in (Zapata et al., 1994) for many applications and tested for robots moving in 2 dimensions, flying robots or autonomous submarines and also mobile manipulators (Cacitti, 2000). This algorithm that was initially designed for obstacle avoidance, has two main advantages. First, the environment does not need to be *a priori* known, and second, the controller can take into account other constraints such as target pursuit (see paragraph 2.2), altitude maintaining, course control and so on. Figure 9 is a omnidirectional robot (equipped with 3 "sweedish" wheels) on which the DVZ algorithm was implemented. This robot has 3 ultrasonic sensors allowing the obstacle detection in 3 directions of space. The robot moves in the *(x,y)* plane and constantly rotates in order to update the proximity information in all directions of space.

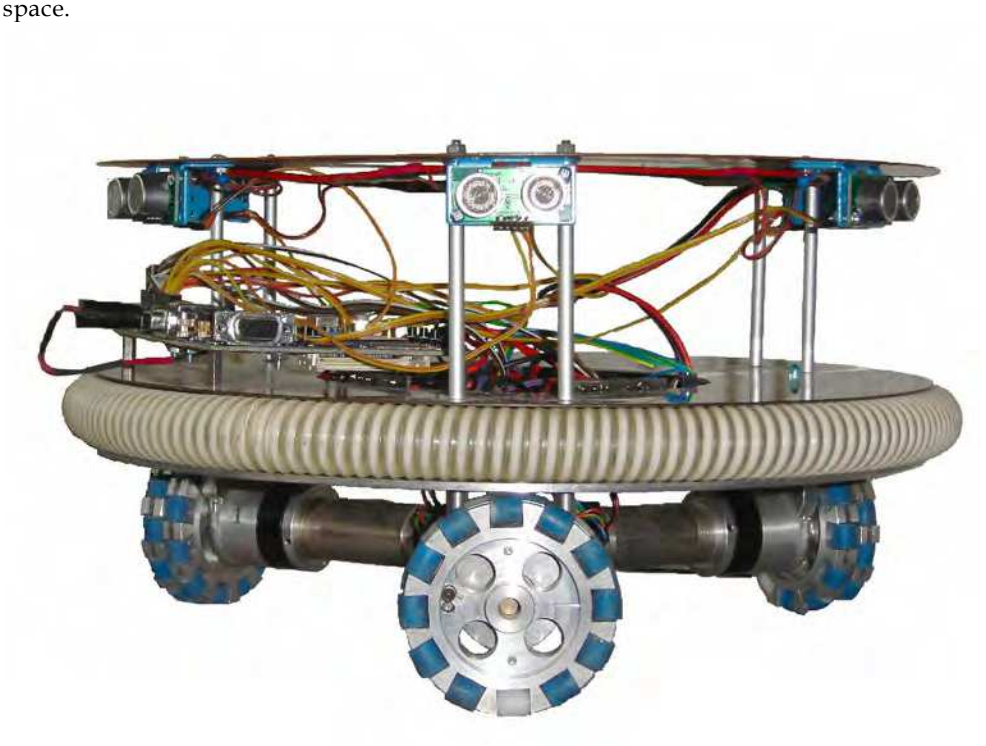


Fig. 9. The RATMOBILE robot.

5.2 Visual information

Images provide high-level sensory information that must be managed to make mobile robots move faster in unpredictable and dynamic worlds. Concurrent design of hardware/software systems (co-design) uses prototyping to evaluate the performance of the final system and to facilitate hardware-software partitioning. Prototype simulation can be achieved in quasi real-time (the clock frequency is generally one order of magnitude smaller than that of the final system on silicon). Signal processing applications (like the image processing algorithm described above) require very long validation digital patterns that lead to a prohibitive time for model simulation. Prototyping a signal processing system on a programmable and configurable platform allows validation of the processing function in real time. Within this framework, the system is first designed at a high abstraction level as a dataflow, with potential links with MATLAB tools.

Many researches have addressed co-design methods. In (Lodha et al.,1998) the authors studied a motion-based collision detection algorithm divided into 2 main subroutines that are allocated to firmware and hardware resources (DSP and FPGA). The algorithm has been tested and compared on a Sun ULTRA1 and Pentium pro133, and targeted to a DSP56002/XILINKS™ FPGAs for real-time implementation.

In (Haldar et al., 2000) the authors designed a real time vision embedded system for motion segmentation purposes, collision detection and object tracking. The hardware is based on a Pentium™ III PC running a Linux light kernel at video rates up to 25 frames per second with medium size images (128×96 and 384×288).

5.3 Hardware and software description

We co-designed vision systems, implemented and tested them on a EPXA10 development board from the Altera™ company. The Excalibur™ development board is built around an EPXA10 device which integrates an ARM922T-based hard processor with 16 Mbytes of 16-bit-wide flash memory, SDRAM controller, UARTs, DMA, Ethernet and PIOs. This device can also host one or several NIOS™ embedded RISC soft processors with 32-bit data paths with the same kinds of interfaces. The device can also be interfaced (UART) with an onboard PC.

This board comes with a development software kit (Quartus II™ and SOPC™ builder) which allows simulation and of software download. It is also possible to use dedicated IPs (Intellectual Property codes) and to develop dedicated home-made IPs. The prototype board is connected through IOs to the different sensors and actuators. The circuit is divided into 2 zones: the processor zone which is hardware-implemented (in the case of the ARM-based board) or software-implemented (in the case of the NIOS-based board). The PLD zone which contains the different home-made IPs dedicated to the control of sensors, actuators and, of course, the vision system. The design involves defining what part of the whole process will be implemented with hardware resources and what part will be implemented with software resources.

There are 3 "programming" levels:

- A low level design (written in VHDL language), part of the "hardware resources", devoted to low level functions such as encoder reading, actuator control, image processing and image analysis.
- A medium level design (written in C language), part of the "software resources", which incorporates functions that are not easy to integrate at a hardware level (e.g. fix point operations).

- A high level design, implemented in C language on the onboard PC to manage communication, supervision and some mathematical issues (e.g. floating point operations).

The development software allows writing and testing of the VHDL code and of the C code is also implemented on the imbedded processors (NIOS and/or ARM).
The vision system involves several home-made IPs (in our case, 3D reconstruction and colour image segmentation) linked to the vision hardware here, two 492×656 colour cameras with LVDS connections to the board.

5.4 Real-time implementation

This paragraph describes several aspects of the real implementation of the vision-based algorithms both on an Excalibur NIOS embedded prototype board and on a Excalibur ARM-based prototype board. The prototyping was only tested here with test images obtained with digital cameras and then downloaded to the embedded processor. We will focus on 4 aspects of this integration:

- The PLD structure for 3D reconstruction
- The VHDL implementation of a vertical gradient
- The VHDL implementation of a set of colour-prototypes
- The morphological dilatation operator

PLD structure for stereovision

The 3D reconstruction algorithm was developed for grey 256-level images. It is composed of 7 main blocks:

- a RAM controller that stores the right and left images
- a zone operator that determines the zone to which the incoming pixel belongs (44 zones for global vision and 16 for the fovea. See \ref{41})
- a gradient operator that computes the threshold of the 256-level image
- an operator that computes the best characteristic point for each of the 60 zones when there is one
- a matching operator that search the left image in order to minimize the SAD (sum of absolute differences) between an 11x11 patch around each characteristic right point and an 11x11 patch moving in the left image
- a sequencing machine that generates all the logical control signals
- a processor (NIOS or ARM) that reads images from a PC, transmits them to the PLD, gets the results and transmits them back to the PC

VHDL implementation of a vertical gradient

VHDL implementation of a gradient operator is a very good example of the difference between a high-level description language (C or MATLAB) and a logical language (VHDL). In this case, the main difficulty is to provide the information needed to compute the convolution of a given image and a gradient mask (in general a 3×3 mask). The $m \times n$ image arrives sequentially and the incoming pixel allows computation of the gradient of the pixel up-and-left from it. The gradient logical function consists of storing 2 lines and 3 pixels, including the incoming pixel, and computing the gradient. These $2n + 3$ pixels are stored in a FIFO-type memory. At each top of the pixel clock, the whole memory is translated and a new gradient is computed.

VHDL implementation of a set of prototypes

Each prototype is a VHDL object involving a maximum of N neurons(subroutines of the VHDL code) and can be configured in C by the NIOS during the programming phase. In this phase, the user chooses the number of prototypes to be programmed, the number of neurons per prototype, and the weight of each neuron. At each top of the programming clock (Progclk), the processor provides the PLD with a validation signal and the weight of the i^{th} neuron to be programmed.

In the application phase, at each incoming pixel arriving at each top of the pixel clock (Pixclk), the theme gives the probability of the pixel belonging to it.

All of these probabilities feed a compete layer that chooses the best prototype. The Gaussian activation function was tabulated in a Look-Up-Table (LUT) whose inputs are a sampling of distances in the RGB space and whose outputs are the corresponding Gaussian probabilities normalized between 0 and 2047 (typically, a power of 2).

Morphological erosion and dilatation

The second step of image segmentation involves filtering using morphological operators. We implemented a sequence of two operators: an erosion followed by a dilatation. The main interest of this sequence is to clean the images, i.e. small objects or protuberances are erased while the size of bigger objects is maintained. In this experiment, we chose a classical 3×3 full matrix as the structural element operating on the image.

Figure 10 shows an original image (left), the pseudo-color image representing the different zones corresponding to the 8 themes previously defined in paragraph \ref{42} (center) and the binary image representing the 6^{th} prototype, in this case the yellow objects (right).

A morphological filter is implemented according to the same principles that underlie the gradient computation. A 3×3 structuring element requires 2 lines and 3 pixels to be stored, including the incoming pixel.

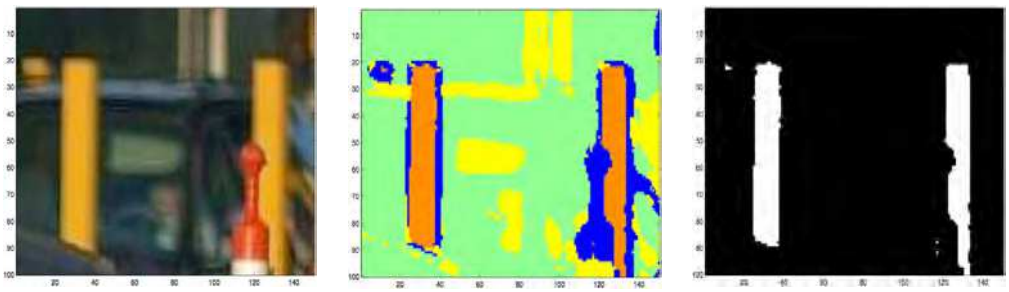


Fig. 10. Colour segmentation and morpho-filtering.

6. Conclusion

This chapter addressed the problem of controlling the reactive behaviours of a mobile robot evolving in unstructured and dynamic environments. We have carried out successful experiments for determining the distance field of a mobile robot using two

cameras. This vision system is coupled with a collision avoidance algorithm based on the DVZ method. In order to implement the previous algorithms, we developed a board including a FPGA programmable circuit from the Altera society. The image storing and the matching of the distance map is made in real time mode. The onboard PC computes the DVZ with the distance information issued from the processing board. This board can be interconnected with all the PC 104 interface board and with all the PC peripherals (screen, keyboard...).The needs for efficient vision-based systems seem obvious. Even if image processing often requires space and time, only high level sensory information provided by vision systems, can provide mobile robots with real autonomous capabilities. Looking for efficiency in vision-based systems can be done either by investigating fast natural vision-systems (natural systems) or by promoting co-designed hardware/software solutions. A mixed approach seems, of course, a good solution.

The bio-plausible reactive control algorithm for obstacle avoidance and target pursuit was tested for robots moving in 2 dimensions, flying robots or autonomous submarines. This algorithm that was initially designed for obstacle avoidance, has two main advantages. First, the environment does not need to be *a priori* known, and second, the controller can take into account other constraints such as target pursuit, altitude maintaining and course control.

7. References

SAB (1999-2003). Simulation of adaptive behaviors, from animals to animats. 21

Berthoz, A. and Weiss, G. (2000). *The Brain's Sense of Movement*. Perspectives in Cognitive Neuroscience, Paris.

Beymer, D. and Konolige, K. (1999). *Real-time tracking of multiple people using continuous detection*. In IEEE Frame Rate Workshop.

Brown, M., Burschka, D., and Hager, G. (2003). *Advances in computational stereo*. IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol.25, N°8.

Cacitti, A. (2000). *Comportamento Reattivo di Manipulatori mobili non-Olonomi basato sul metodo del la Zona Virtuale Deformabile*. Tesi di laurea, Facolta di Ingegneria, Universita degli Studi di Bologna.

Faugeras, O. and al (1993). *Real time correlation-based stereo:algorithm, implementations and applications*. Technical Report 2013, INRIA.

Haldar, V., Vardhan, G., Saxena, A., Banerjee, S., and Balakrishnan, M. (2000). *Design of embedded systems for real-time vision*. In Indian Conference on Computer, Vision, Graphicsand Image Processing (ICVGIP2000), Bangalore,India.

Holenstein,E. ; Badreddin, E, (1991) *Collision Avoidance in a Behaviour-Based Mobile Robot Design*, Proc. IEEE Int. Conf. on Rob. and Aut., Sacramento, California, USA.

Kathib, O. (1985); *Real-Time Obstacle Avoidance for Manipulators and Mobile Robots*, Proc. IEEE Int. Conf. on Rob. and Aut., pp.500-505

Kimura, S., Shinbo, T., Yamaguchi, H., Kawamura, E., and Naka, K. (1999). *A convolver-based real-time stereo machine (sazan).* Computer Vision and Pattern Recognition, Vol.1, pages 457–463.

Lodha, S., Gupta, S., Balakrishnan, M., and Banerjee, S. (1998). *Detection and avoidance: A case study for design space exploration in hw-sw codesign*. In 11th conference on VLSI Design, Chennai, India.

VanRullen, R.and, T. S. (2002). *Surfing a spike wave down the ventral stream*. Vision Research, Vol.42, pages 2593–2615.

Wurtz, R. and Lourens, T. (2000). Corner detection in color images through a multiscale combination of end-stopped cortical cells. Image and Vision Computing, Vol.18 N°6-7, pages 531-541.

Zapata, R., Lépinay, P., and Thompson, P. (1994). *Reactive behaviors of fast mobile robots*. Journal of Robotic Systems, Vol.1 N°8.

Mobile Robotics, Moving Intelligence

Edited by Jonas Buchli

ISBN 3-86611-284-X

Hard cover, 586 pages

Publisher Pro Literatur Verlag, Germany / ARS, Austria

Published online 01, December, 2006

Published in print edition December, 2006

This book covers many aspects of the exciting research in mobile robotics. It deals with different aspects of the control problem, especially also under uncertainty and faults. Mechanical design issues are discussed along with new sensor and actuator concepts. Games like soccer are a good example which comprise many of the aforementioned challenges in a single comprehensive and in the same time entertaining framework. Thus, the book comprises contributions dealing with aspects of the Robotcup competition. The reader will get a feel how the problems cover virtually all engineering disciplines ranging from theoretical research to very application specific work. In addition interesting problems for physics and mathematics arises out of such research. We hope this book will be an inspiring source of knowledge and ideas, stimulating further research in this exciting field. The promises and possible benefits of such efforts are manifold, they range from new transportation systems, intelligent cars to flexible assistants in factories and construction sites, over service robot which assist and support us in daily live, all the way to the possibility for efficient help for impaired and advances in prosthetics.