# NightMare: a simple and efficient vision-based robot

António J. R. Neves , Gustavo A. Corrente and João Figueiredo
DET / IEETA, Universidade de Aveiro
an@ieeta.pt, gcorrente@gmail.com, joao.figueiredo@ieeta.pt

***Abstract*— This paper presents the architecture and algorithms developed for NightMare, a vision-based robot built by the authors for the participation in the 2005 edition of the Micro-Rato contest. This robot has won the special prize for innovation, awarded by the jury, due to the use of vision and a computer running Linux. The architecture of NightMare integrates vision information with a robust navigation algorithm and an odometry system maintaining a relative simple hardware and a small size.**

## I. Introduction

NightMare was designed to be a versatile robot capable of participating in several robot contests, for example, in the Micro-Rato contest [1], [4], in autonomous driving [6] and in the Firefighter Robot contest [5].

The robot has participated in the 2005's edition of the Micro-Rato contest, organized by the University of Aveiro. In this contest, the NightMare robot won the special prize for innovation, due to its architecture, in particular the use of vision and a computer, running Linux, responsible for the main processing. Other interesting features are the developed low-level modules, using a micro-controller Microchip PIC18F258, and the high-level software, in particular the world representation, vision processing and path planning algorithms.

This paper is organized as follows. The next section describes the hardware design. The developed software is presented in Section III. Our first challenge, the Micro-Rato contest, is presented in Section IV. Finally, in Section V we draw some conclusions and present future work.

## II. Hardware design

NightMare is a round robot with a diameter of 28 *cm* and a height of 40 *cm* (see Fig. 1). The main sensors used in the robot are two webcams. The processing is done using a Mini-Itx VIA EPIA Nehemiah M10000 [8], [9]. This computer runs Debian GNU/Linux 3.1 (Sarge) [7], installed on a compact flash with 512 MB, choosing only the essential development tools and a graphical environment.

NightMare acquires the environment information using two cameras, one to see the world around it, OmniVision, and other to see the world in front of the robot, FrontVision (in Fig. 4 it is shown in detail the cameras configuration). To implement the OmniVision system we use a camera with a crown mirror lamp (a simple and cheap solution).

The robot has a low-level structure controlled by a micro-controller Microchip PIC18F258. A global overview of the hardware architecture is shown in Fig. 2.

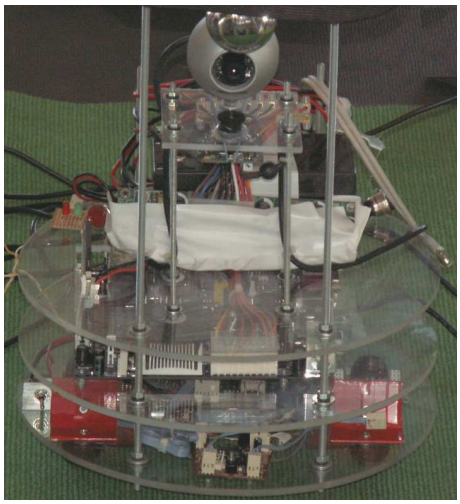

Fig. 1. A view of NightMare. The robot has four levels. On the top the vision system. Under the vision system are the batteries. Next is the Mini-Itx main-board and the power management board. On the lower level are the microcontroler board, the H-bridge, the PS/2 mouse and the motors. The levels are separated with an acrylic circular plate.

Connected to the micro-controller are a PS/2 mouse, two leds, a ground sensor and a module with a dual H-bridge for driving DC motors. The micro-controller is also responsible for the communication with the computer.

The PS/2 mouse was chosen because it has an easy interface and provides three buttons and two position encoders that we use to implement the closed-loop circuit of the motors and the odometry system. Two buttons are used to give the start and stop orders to the robot.

Two leds are used to signal the state of the robot and each led can have three states: off, on and blinking. In the Micro-Rato contest, the two leds have specific functions to mark some reached goals.

A ground sensor detects if the robot is on a black surface or not. Its value is read periodically by the micro-controller.

The robot has two motors controlled separately using PWM (Pulse Width Modulation) with an 100Hz wave. The closed-loop circuit is controlled with a PID (Proportional, Integrator and Derivative) controller. This controller is implemented in software.

The communication with computer is done using a RS232 interface with a bitrate of 115200 bps. The communication is frame-based. The frame structure was defined specifically for this application. The micro-controller sends the buttons and

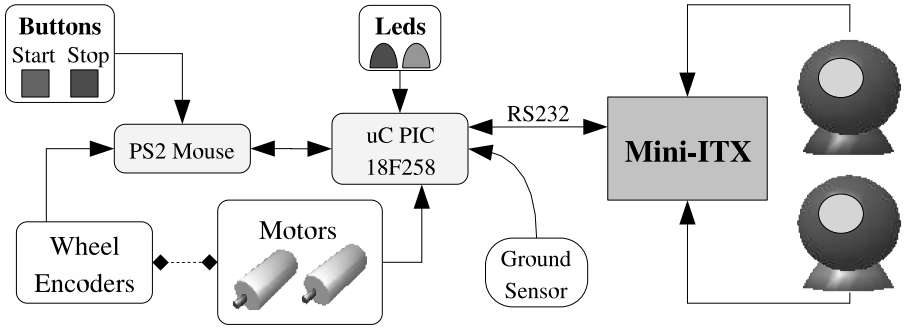


Fig. 2. Representation of the hardware diagram.

encoders information to the computer and receives the power to apply to the motors and the leds status.

## III. Software design

The software that runs in the micro-controller, periodically executes the following tasks: processes the information given by the Mini-itx, reads the information of the PS/2 mouse (odometry and buttons), updates the leds state, sends the new information to the Mini-itx and updates the motors speed. There is also an emergency stop task that stops the motors when the micro-controller doesn't receive any command from the PC for over a second.

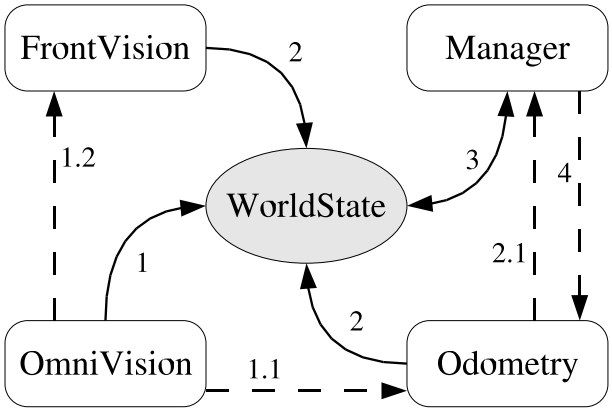


Fig. 3. The Software Diagram.

The software running in the Mini-itx consists in four processes (OmniVision, FrontVision, Odometry and Manager). All processes use shared memory to write/read robot information, denoted WorldState. A global view of the software architecture, including the synchronous messages, is shown in Fig. 3. The access to shared memory (WorldState) is synchronized using message queues. The *clock* used to synchronize the communication is generated by the OmniVision process. After acquiring and processing the image, *OmniVision* writes to WorldState (Fig. 3-1) and sends two messages, one to *FrontVision* (Fig. 3-1.2) and other to *Odometry* (Fig. 3-1.1), informing these processes to write their information in WorldState. After this, Odometry sends a message to Manager, to wake it up (for manager updates some WorldState fields, think and act) (Fig. 3-3). When manager decides the power values to apply to the motors, it sends a message to Odometry (Fig. 3-4), responsible to send this information to the micro-controller.

### A. WorldState

The WorldState represents all the information relevant to the robot decisions and behaviors. This structure contains information of the landmarks and the beacon, such as distance, direction, position and visibility; information of elapsed time; robot data, such as position and orientation. For path planning we have a map that contains information about the environment where the robot is inserted (free, unknown or wall).

### B. Algorithms

The A* [3] algorithm was used for path planning. This algorithm gives the shortest path between two points. The implementation of A* is based on heaps (priority queues). The algorithm uses the world information, that consists in a matrix of points. Each point can be free, unknown or wall. For A* proposal, each type of point has a different passing cost. The points marked as wall are discarded and unknown points have higher costs than free points. Moreover, each point doesn't have a fixed cost but an associated interval, based on the distance. The cost of each point are increased or decreased with an increment depending on the robot distance. Higher distance corresponds to a smaller increment/decrement.

The collision avoidance is made by an algorithm denoted *escapeFilter*, a simplified version of the algorithm presented in [2]. The *escapeFilter* uses the OmniVision information and decides the new robot orientation. This is applied after deciding what to do. The "OmniVision sensors" are divided in three sectors (center, left and right). If the desired direction is occupied, it chooses a better direction. In case of the three sectors are occupied, rotate until finding a free sector. It was also developed an algorithm to follow walls that doesn't use Odometry.

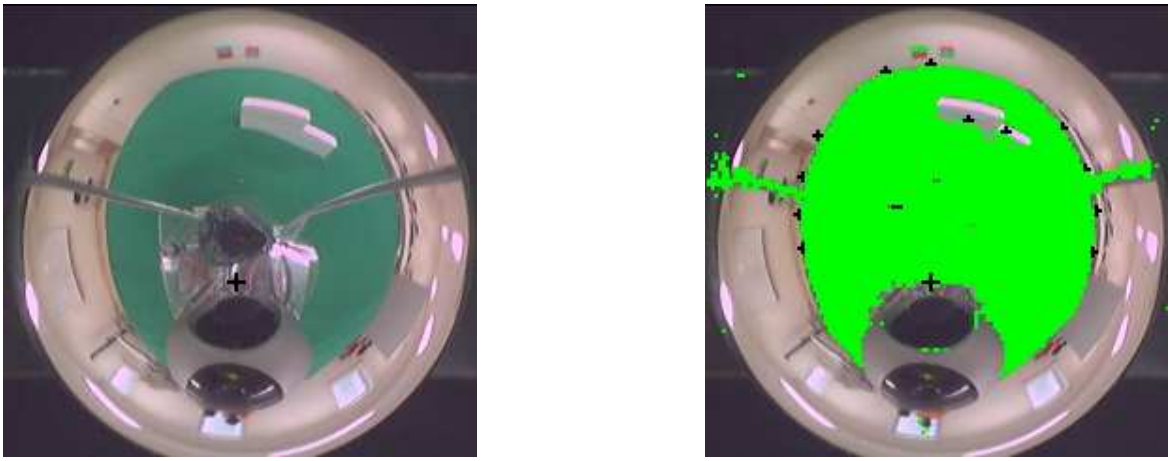

Fig. 5. An image example of OmniVision. On the left, the original image. On the right the processed image. The black crosses denote the obstacles position for each virtual sensor.

Fig. 4. The robot vision system in detail. Here we can see the configuration of the cameras. Note the position of the lamp, acting as a mirror for the OmniVision.

### C. Vision

We use two usb *Logitech QuickCam 4000 Pro* cameras [1] with the pwc[2] module for linux. These cameras can capture video up to 640 x 480 pixels (VGA CMOS) with a frame-rate up to 30 frames per second. The video format is YUV (4:2:0):, the U and V color components are subsampled in the vertical and horizontal directions by a factor of two. With the pwc module it is possible to configure some camera settings such as the resolution, frame-rate, white balance, gain, compression, shutter speed, etc.

The OmniVision and FrontVision systems are further described below in the context of the Micro-Rato contest, which was the first purpose of the robot. In Section IV we describe the contest with more detail.

Image processing in the Micro-Rato contest is simplified, since the objects are color coded. The ground is green, the walls are white, the beacon and other marks have different and known colors. We explore this fact defining color-ranges for each object, that we can calibrate when needed. To perform color calibration, and find some settings for the camera, we developed some tools.

Since the objects are color coded, all processing is made using UV information. This approach decreases the processing time since the UV image is four times smaller than the Y image.

### D. OmniVision

The OmniVision system is used to detect the walls and other obstacles that can exist in the labyrinth. To perform this task, we define some virtual sensors. These sensors are radial search lines, spaced by a constant angle. We search for color transitions, between ground color and other color. Each search line has one well defined angle relatively to the robot. When we find one transition, we know that for that angle there is an obstacle at some calculated distance.

The real distance to the objects is calculated applying a transformation of pixel coords to distance, obtained in the image. The transformation was determined measuring some real distances and using polynomial regression.

With the method described above, we can have a good perception of the free space and the presence of obstacles around the robot.

To ignore the own body, the center of the image, the software uses a non-search mask consisting in a set of points that don't be searched.

Fig. 5 shows an image acquired by OmniVision system. The omniVision camera works with a frame-rate of 15 fps. This frame-rate is used to synchronize all the existing processes (see Fig. 3).

[1] `http://www.logitech.com`

[2] This module can be obtained from `http://www.saillard.org/linux/pwc/`

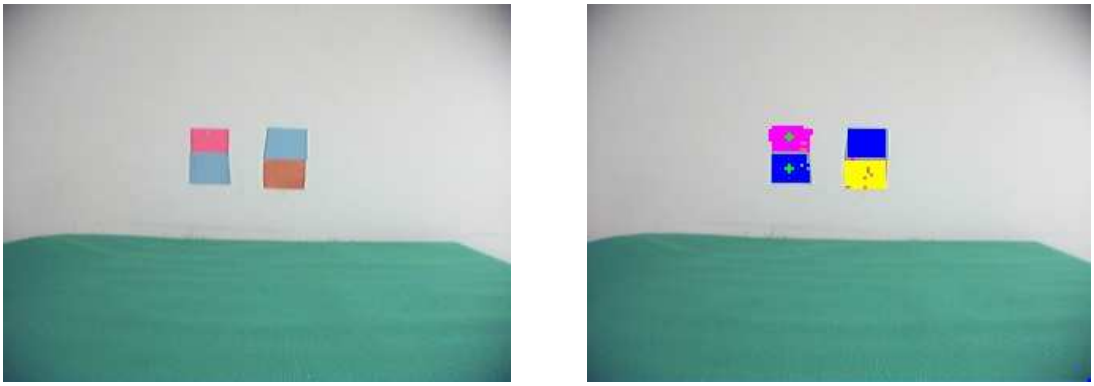

Fig. 6. An image example of FrontVision. On the left the original image. Ont the right the processed image.

Fig. 7. One moment of the competition.

### E. FrontVision

The FrontVision system is used to search for the beacon and the marks existing in each corner of the labyrinth. We developed an efficient algorithm to search objects scanning the image using a radial-based approach. When the position of an object is unknown, we start the search in the center of the image. If the object was detected in the last frame acquired, we start the new search in that position.

We search for all objects (Beacon and LandMarks) in every frame received. This search is done with a frame-rate of 5 fps. At the end of the processing, we update the position of each object, if we have seen them.

The objects are validated following certain rules based on the physical properties of them like the relative color position, number of pixels of the object, etc.

Fig. 6 shows an image acquired by FrontVision camera.

### F. Odometry

The Odometry process is implemented on the PC as a process that handles the communication with the micro-controller board. It receives the messages from the micro-controller and places the relevant information on the WorldState. It also computes the robot position with the information it receives on those messages from the encoders and places it on the WorldState. Lastly the process checks the WorldState for new led and speed commands to send to the micro-controller.

### G. Manager

Manager is responsible for the robot behaviors, decisions and path planning. Behaviors are organized in three parts: STATUS, OBJECTIVE and STATE. STATUS gives the ”state of the contest” (stop, pause, run, finish). OBJECTIVE is the main goal, in the Microrato contest four main goals are used: beacon, home, return, calibrate. The auxiliary goals are supported by STATE. This STATE is organized in as follows: wander (if never seen beacon, we must explore the labyrinth), followPath, followWall.

## IV. Micro-Rato 2005 contest

The first challenge of the NightMare robot was the 2005's edition of Micro-Rato contest [4], organized by the University of Aveiro. In this contest, the robot's task is to go from a start area to a goal area in the shortest time possible and return to the starting point. The robot must do the trajectory with the minimum number of collisions with the environment or with other robots.

The goal area (**Beacon**) is identified by an object with a fixed dimension and two colors. The ground in the goal area has a 60 *cm* diameter circle made with a dark material of low infrared reflection coefficient.

The competition area is a square of $7.5 \times 7.5$ *m* with a green ground. Inside of this area, white objects (**Walls**) are placed forming a labyrinth. The minimum distance among every two walls is 50 *cm*. The competition area have also in each corner one cylinder with two colors (**LandMarks**).

## V. Conclusion

In this paper we presented a simple and efficient vision-based robot. Using vision as sensor we have a lot of advantages, facilitating the use of this robot in various contexts and environments. The robot has participated in the 2005's edition of the Micro-Rato contest and won the special prize for innovation. The project is in its the beginning, a lot of work needs to be done in hardware and software. Next year we will try to participate in more robot contests, using our vision capabilities and our robot design.

## VI. Acknowledgments

The authors would like to thank Armando J. Pinho for providing some source code related to robot vision and his help on the reviewing of this paper.

## References

[1] Almeida, L., Fonseca P., Azevedo, J. L., Cunha, B., *The Micro-Rato Contest: Mobile Robotics for All*, Proc. of the Portuguese Control Conference, CONTROLO 2000, Guimaraes (2000).

[2] Javier Minguez, Luis Montano, *Nearness Diagram (ND) Navigation: Collision avoidance in trouble*, IEEE Trans. on Robotics and Automation, vol.20, no.1, february 2004.

[3] Russel, S., Norvig P.: *Artificial Intelligence: A Modern Approach*, 2nd Edition, 2003.

[4] Micro-Rato contest homepage, `http://microrato.ua.pt/`

[5] Fireman Robot contest homepage, `http://www.estg.ipg.pt/robobombeiro/`

[6] ROBOTICA 2005 homepage, `http://robotics.dem.uc.pt/`

[7] Debian homepage, `http://www.debian.org/`

[8] VIA mini-itx specifications homepage, `http://www.viaembedded.com/`

[9] Mini-itx homepage, `http://mini-itx.com/`