Vision Sensors on the Webots Simulator

Yuri López de Meneses and Olivier Michel

Microprocessor Lab (LAMI)
Swiss Federal Institute of Technology (EPFL)
Lausanne, Switzerland.
Tel: +41 21 693 39 08 Fax: +41 21 693 52 63
lopezdem@di.epfl.ch, Olivier.Michel@epfl.ch

Abstract This paper presents the implementation of the panoramic, linear, artificial retina EDI within the frame of the Webots mobile-robotics simulator. The structure and functioning of the simulation are presented. Realistic sensor readings and software compatibility are the two key concerns of the simulator. A realistic sensor output is obtained by modelling the physical processes underlying the sensors. Interface compatibility between the real and simulated retina has been made possible through the use of the Khepera API. The simulator is validated by running a control algorithm developed on a real robot equipped with such a vision sensor and obtaining the same behavior.

1 Introduction

Simulators have always been used in mobile robotics [6] [2] [3] [10] . They allow researchers to develop complex algorithms without having to suffer from unreliable and sometimes expensive hardware. Recently the possibility of working with simple and yet reliable mobile robots, such as the Khepera [7] and Nomad [8], has allowed a greater part of the mobile-robotics research community to switch from simulators to real platforms.

However, some areas still benefit from the use of mobile-robotics simulators. For instance, multiple-partner projects in which the research tasks are distributed among several partners while having a single robotic platform on which to test. In these situations, robot simulators are a practical solution to flexibly test the robot in different environments, but they should be realistic enough to allow the developed algorithms to be used on the real robot. Another area that benefits from robotic simulators is Evolutionary Robotics and Robot Learning. Experiments based on the adaptation capabilities that evolution provides (Genetic Algorithms) or simple learning by an individual robot (Reinforcement Learning) can run for several days or weeks. That can easily be above the mean time between failures (MTBF) of the hardware and it can be very arduous to supervise by a human if his intervention is needed. Therefore, many works on learning and adaptation have been developed on simulated robots [10],[2].

J.-C. Heudin (Ed.): Virtual Worlds 98, LNAI 1434, pp. 264–273, 1998.

Mobile-robotics simulators can be roughly classified into two classes which we have defined as processor-based simulators and sensor-based simulators. In the former, the robots usually move in a discrete workspace where the surrounding obstacles occupy cells in the grid [9]. The robot receives inputs from neighboring cells, and these inputs are often of symbolic nature, such as "a person is in the front cell". The processing capabilities of the mobile robot are well simulated, but not the sensory ones. These simulators are very fast and they afford to develop complex algorithms. However, the algorithms are difficult to test on real robots. On the other hand, sensor-based simulators try to mimic the response of the sensors to the surrounding environment [8]. This requires a geometric and physical model of the environment, as well as adequate modelling of the physical processes on which the sensors are based. The simulated robot moves in continuous space, interpreting the sensor signals and deciding the actions to be taken based on this information.

The Webots simulator [4] is a recent example of such sensor-based simulators for mobile robotics. It uses an extension of VRML (Virtual Reality Modelling Language) to store a description of the robot and its environment, and a rendered image of the scene can be obtained by using an OpenGL-compatible graphics library. This in turn allows to simulate vision sensors, which have always been difficult to introduce in robotic simulators. This paper presents the simulation of a particular vision sensor, the EDI artificial retina, within the Webots simulator. Section 2 introduces the artificial retina, and section 3 discusses its functional simulation. Section 4 deals with an experiment comparing the behavior of a real and a simulated Khepera robot equipped with such a retina. The concluding remarks discuss the project, its applications and future work.

2 The Panoramic, Linear, Artificial Retina EDI

The EDI[1] panoramic, linear, artificial retina is an analog-VLSI vision chip designed by Oliver Landolt, at CSEM (Neuchâtel, Switzerland), intended to evaluate the capabilities of such bioinspired sensors. The Microprocessor Laboratory (LAMI) of the EPFL is currently working on the application of such sensors to mobile robotics, and most particularly, on a Khepera robot.

The EDI retina has a linear array of 150 photodiodes lying on an arc that covers 240 degrees of view. But as its name suggests, the EDI is more than a simple vision sensor capable of transducing light. As in most animal retinas, some simple signal processing takes place in the vicinity of the photodetectors, still in the analog domain. This local processing has the effect of filtering the information, reducing its bandwidth, and providing the next stage with a more relevant information. In EDI's case, 3 resistive grids [11] can be combined to

[1] EDI is a shorthand for ED084V2A, the CSEM's device number.

apply a lowpass, a bandpass or a highpass filter on the image. An odd-impulse-response filter can also be applied, producing the derivative of the image. The filter output can be accessed by a microprocessor thanks to the internal A/D converter present on the retina. In this way, a 64-gray-level image can be used by the robot for any further processing.

The EDI retina has been mounted on a Khepera turret known as Panoramic turret, together with the necessary ancillary devices and bias circuits. The turret has been included as a plug-in module in the Webots simulator. The following section discusses the simulation of the Panoramic turret and the EDI retina.

Figure1. The Khepera Panoramic turret incorporates an EDI artificial, linear retina.

3 The Panoramic Turret on the Webots Simulator

The simulation of the EDI retina follows its functional blocks, which comprise the optics, the photodetectors, the filters and the digital conversion. Each block will be briefly discussed in the subsections that follow. A final subsection will cover the software interface and its compatibility with the real retina.

3.1 Optics

The first step of the simulation is to calculate the incident light intensity on each photodetector. We will consider the pixel values of the OpenGL rendering engine as the light intensity falling on the pixel surface. The real EDI retina has a spheric mirror and a lens that focus the image on the photosensors, which lie on the chip as seen in figure 2. Ideally, the incident light that reaches the silicon surface should be the same as the incident light that hits a ring on the spheric mirror. Taking this hypothesis, we will try to obtain the incident light on this ring.

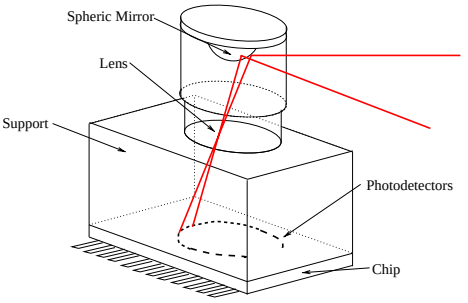

Figure2. The optics of the EDI retina

The OpenGL rendering engine produces the image of the scene as it would be seen by a flat screen camera. However, the EDI retina is spherical, as it sees an image on 240 degrees in azimuth and 11 degrees in elevation. To solve this problem, the 240-degree view is obtained from two 120-degree (azimuth) by 11 degrees (elevation) projections on a flat surface, each one projecting on a 75 x 7 pixel array, as shown in figure 3.

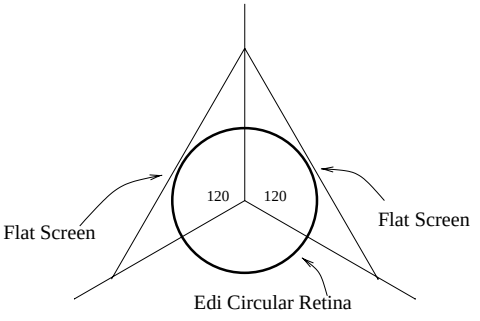

Figure3. Decomposition of a 240-degrees, cylindrical view into 2 flat views.

Next, both arrays are warped by using the transformations as it can be inferred from figure 4. The flat-screen coordinates x,y are transformed into the cylindrical coordinates ϕ and θ as a function of the focal distance f, which is equal to the radius of the cylinder:

$$x = f \cdot \tan(\theta) \Longrightarrow \theta = \arctan(\tfrac{x}{f})$$

$$y = \sqrt{x^2 + f^2} \cdot \tan(\phi) = f \cdot \tan(\phi) \cdot \sqrt{1 + \tan^2(\theta)} \Longrightarrow \phi = \arctan(\tfrac{y}{\sqrt{x^2+f^2}})$$

Eventually the vertical pixels are averaged to obtain two 75 x 1 pixel vectors, and the two vectors are concatenated to produce a 150 pixel vector.

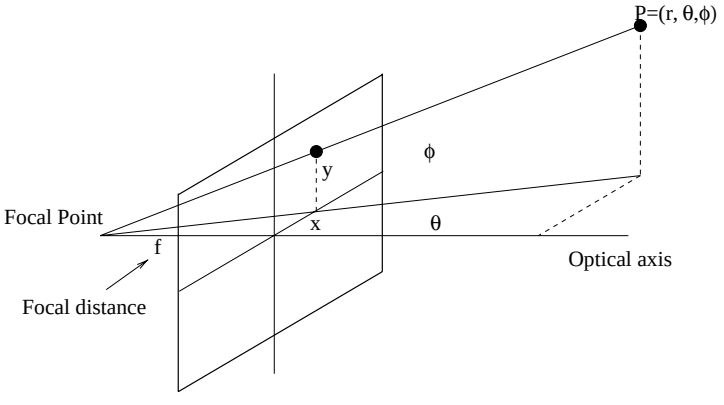

Figure4. Projection of a world in spherical coordinates on to a flat screen

3.2 Photodetectors and Normalization Circuit

The image obtained in the previous subsection is considered as a measure of the incident light intensity on each of the 150 photodetectors. The RGB image is converted to gray scale by using the RGB to Y (luminance) transformation of the PAL TV standard. This transformation gives a certain weight to each color channel. A better modelization could be achieved if the spectral response of the photodetectors were used to generate the luminance. The luminance thus generated is considered as the photocurrent on each photodiode. Next, a normalization circuit divides each pixel's photocurrent by the average current of the whole array. This affords to obtain an illumination-invariant image, that is, an image that will not be perturbed by changes in the global illumination settings of the scene. The normalized current is stored in the model's `analog[ ]` array.

3.3 Filtering Layer

The filtering layer simulates the filtering capabilities of the artificial retina. These are implemented with VLSI resistive diffusion networks [11] that can be combined to enhance the image and to extract some characteristic features from it. A simple resistive diffusion network has a spatial-lowpass characteristic that can be used to reduce noise in the image and detect uniform regions. Other filters are generated by combining such networks. For instance, the highpass or edge-enhancing filter is obtained by subtracting from the original image its lowpass

version. The odd impulse-response filter is generated by subtracting the output from two unidirectional diffusion networks. It produces the derivative of the input image, and it can be used to detect sharp edges in the scene.

1-D resistive diffusion networks behave as low-pass filters of exponential impulse response in the spatial and spatial-frequency domains:

$$h(n) = e^{\frac{-|n|}{\lambda}} \overset{\mathcal{F}}{\Longleftrightarrow} H(\omega) = \frac{\lambda}{\pi} \cdot \frac{1}{1+(\lambda \cdot \omega)^2}$$

Resistive diffusion networks are parallel, locally-connected systems, that perform computations on a real-time basis. This makes it difficult to recreate on a single processor. However, since the system is made out of resistors, that is, linear devices, it can be shown that the output is the convolution of the input signal with a kernel or impulse response of the forms that can be seen in figure 5.

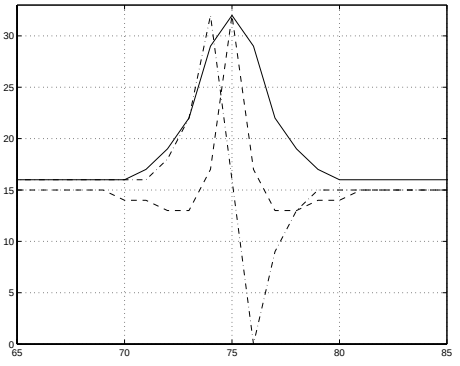

Figure5. Impulse responses of the lowpass (-), highpass (--) and odd (-.) filters.

The result of convolving the `analog[ ]` array with the resistive network impulse response is stored in the `filtered[ ]` array.

3.4 Digital Conversion

The EDI retina has a 6-bit successive-approximation A/D converter, that digitizes the output current from the resistive grid into 64 gray levels. Since the analog filters can produce negative values, the higher bit is used as a sign bit. The conversion, defined by the formula $digital[] = 31.5 \cdot (1 + \frac{filtered[\,]}{I_{bda}})$ depends on the variable I_{bda}, which represents a bias current on the EDI chip. In the simulator this variable has been set to a value that makes the output barely saturate when there is a single maximally iluminated photodetector.

3.5 Software Interface: The Khepera API

The benefits of such a sensor-based simulator would be lost if the algorithms developed on a Webots simulated robot cannot be easily applied on a real robot. To that end, we have developed a common interface for the real and simulated Panoramic turret and Khepera robot, in the form of an Application Program Interface: the Khepera API [5]. Such a common interface allows the user to use the same C source code, without having to change a single comma, on both platforms. Two different libraries, one for the real robot and one for the virtual one are provided with the Webots simulator.

The Khepera API contains several functions to access the Panoramic turret. Some functions allow the user to enable or disable a given filter and other functions are used to read the output of the desired filter. The configuration functions allow the user to set the filter spatial cutoff-frequency, the gray-scale resolution and the region of interest.

4 Experiments: A Robot Regatta

To analyze the validity of the Webots simulator and the Panoramic turret, we have run on simulation a control algorithm that was already tried on the real Khepera. To do so, first we had to rewrite the old code using Khepera API and test it on the real robot. Then we applied the resulting C code to the Webots simulator, and observed whether the same behavior was obtained.

The algorithm combined two different behaviors, obtaining a new, emergent behavior. The first behavior is based on Braitenberg's obstacle avoidance behavior [1], using the 8 IR sensors of the Khepera robot. The IR proximity-sensor values control the motor command through a feed-forward neural network as it can be seen in figure 6. Two bias neurons ensure a forward motion of the robot in the absence of any IR-sensor stimuli. The second behavior is light seeking by using the Panoramic turret. The odd impulse-response filter is used to obtain a derivative of the image, from which the light-intensity local maxima are detected. The robot steers towards the maximum (i.e. light source) that is closer to the center pixel by commanding the motors with a signal proportional to the pixel distance between the light and the center pixel. This motor command is added to the IR network output, producing a smooth transition from obstacle avoidance behavior to light following behavior.

The real and simulated robots are placed on a square arena on which 3 light sources or "buoys" are set forming a triangle (figure 7). The light buoys lie at the height of the retina and are therefore always visible, except if hidden by another buoy. The Khepera will move towards the buoy that lies ahead, and once it reaches it, the IR sensors make it turn away from the light. However, the retina still tries to steer the robot towards the buoy and the combination of

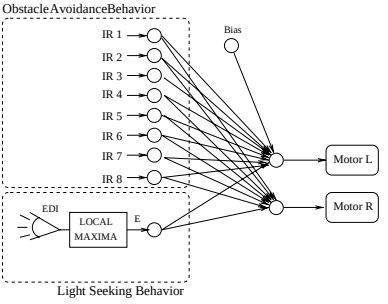

Figure6. The neural network used in the regatta experiment

both commands makes the robot turn around the buoy until it sees a new buoy lying ahead. It will thus move from buoy to buoy as in a regatta. In case there is only one light buoy on the arena, the robot will steer towards it and then turn endlessly around it.

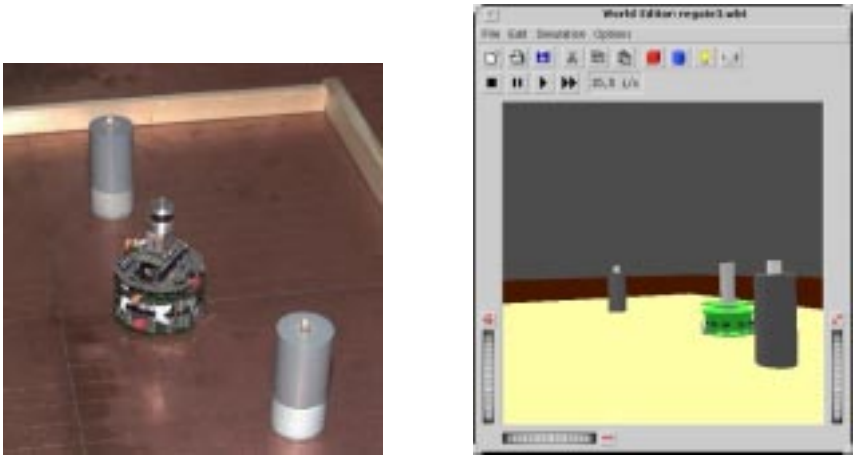

Figure7. The real (a) and simulated (b) Khepera regatta.

The control algorithm does not contain a description of the steps to be taken when a buoy is encountered, nor a definition of what a buoy is. It is a simple combination of two simple behaviors, obstacle avoidance and light-seeking, and yet we can observe the emergence of a new, unforeseen behavior that makes the robot navigate from buoy to buoy. The emergence of this new behavior depends on the balance between the algorithm parameters, i.e. the network weights.

In the simulated part of the experiment, the light buoys are simulated as cylinders with a bright object on top of them (figure 7b). The same C source-code program is downloaded on the simulated Khepera, and the same behavior can be observed: the Khepera moves from buoy to buoy without running into them and turns around the light buoy when there is only one on the arena.

The Panoramic turret simulation is validated by the fact that not a single parameter (i.e. the IR-network weights and the angle-to-speed constant of the light-seeking behavior) of the real controller had to be changed. However, a simulated behavior closer to the real one is obtained if the IR-network gain is doubled. This occurs because the real buoys are covered with IR-reflecting film to allow the robot to sense them form a higher distance, since the cylindric shape and the material from which the buoys are made produce a small IR cross-section. The fact that a realistic behavior is obtained when reflecting against a flat surface seems to support this hypothesis. Since IR-reflecting coatings are not simulated in the Webots simulator, the IR sensor sensitivity or the IR network gain have to be doubled to allow the simulated robot to detect the buoys at the same distance as in the real experiment. This evidence indicates that the Webots simulator models realistically the IR-sensor responses.

5 Conclusion

The paper describes the modelling of a vision sensor within the frame of the Webots mobile-robotics simulator. Great care has been given to model the physical processes underlying the sensor to obtain a realistic sensor reading. The sensor presented in this paper is a very particular one, since the EDI artificial retina is a linear, panoramic (i.e. spheric) sensor with signal processing capabilities. However, the same principles can be applied to simulate other vision sensors and standard cameras. The Khepera's commercially available K213 vision turret is currently being developed in the Webots simulator along the same line.

Our second concern in developing the simulated retina and turret was software compatibility. We see simulators as a tool for the time-consuming development of complex algorithms and for sharing data and results among different research teams. In both cases it is highly desirable to be able to test the algorithm on a real robot, and to easily do so, the code should be compatible. For the Webots simulator, an application program interface (API) has been defined, allowing to interface the real and simulated Khepera robot and its turrets with the same C source code.

A simple experiment has been used to validate the simulated EDI and the Webots simulator. A demonstration program that had previously been run on a real Khepera robot has been tested on the Webots simulator. We were able to reproduce the same behavior without having to change a single parameter of the

robot controller. A more thorough validation would require evolving the same behavior on a real and a simulated robot.

The Webots simulator and the Panoramic turret are currently being used in a landmark-navigation experiment. The simulator is used to partially train the robot and eventually the evolved controller will be transferred to the real Khepera for a final, shorter learning stage. The success of such experiment will further prove the adequacy of the simulator.

Acknowledgements

We wish to thank Eric Fragnière and Olivier Landolt for the helpful discussions on analog VLSI circuits.

References

[1] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge, MA, 1984.

[2] L.M. Gambardella and C. Versino. Robot Motion Planning Integrating Planning Strategies and Learning Methods. In *Proceedings of 2nd International Conference on AI Planning Systems*, 1994.

[3] M. Mataric. Designing and Understanding Adaptive Group Behavior. *Adaptive Behavior*, 4(1):51–80, 1995.

[4] O. Michel. Webots, an Open Mobile-robots Simulator for Research and Education. `http://www.cyberbotics.com/`.

[5] Olivier Michel. *Khepera API Reference Manual.* LAMI-EPFL, Lausanne, Switzerland, 1998.

[6] J.R. Millán. Reinforcement Learning of Goal-directed Obstacle-avoiding Reaction Strategies in an Autonomous Mobile Robot. *Robotics and Autonomous Systems*, (15):275–299, 1995.

[7] F. Mondada, E. Franzi, and P. Ienne. Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proceedings of the Third International Symposium on Experimental Robotics 1993*, pages 501–513. Springer Verlag, 1994.

[8] Nomadic Technologies, Inc. *Nomad User's Manual*, 1996.

[9] P. Toombs. *Reinforcement Learning of Visually Guided Spatial Goal Directed Movement.* PhD thesis, Psychology Department, University of Stirling., 1997.

[10] C. Versino and L.M. Gambardella. Ibots. Learning Real Team Solutions. In *Proceedings of the 1996 Workshop on Learning, Interaction and Organizations in Multiagent Environments*, 1996.

[11] E. Vittoz. Pseudo-resistive Networks and Their Applications to Analog Computation. In *Proceedings of 6th Conference on Microelectronics for Neural Networks and Fuzzy Systems*, Dresden, Germany, September 1997.