A ROBOTIC PLATFORM FOR AUTONOMY STUDIES

Sergio Ribeiro Augusto and Ademar Ferreira
Departamento de Engenharia de Telecomunicações e Controle, Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto 3/380, 05508-900 Sao Paulo, Brazil

Keywords: Mobile robotics, supervised learning, radial basis function networks, teleoperation.

Abstract: This paper describes a mobile robotic platform and a software framework for applications and development of robotic experiments integrating teleoperation and autonomy. An application using supervised learning is developed in which the agent is trained by teleoperation. This allows the agent to learn the perception to action mapping from the teleoperator in real time, such that the task can be repeated in an autonomous way, with some generalization. A radial basis function network (RBF) trained by a sequential learning algorithm is used to learn the mapping. Experimental results are shown.

1 INTRODUCTION

In robotics navigation problems, including learning or not, navigation techniques must be tested in real robots to be useful (DORIGO, 1996). This is due to the uncertainties involved, non uniformity of sensors measurements and real time requirements. To deal with these severe characteristics, this paper proposes a mobile robotics platform developed in a modular and hierarchical way, to be used in real time autonomy studies. The objective is to create a flexible development environment for studies in which teleoperation can be easily integrated with autonomous operation. The idea is to join teleoperation with supervised learning in a way that innate or prior knowledge can be acquired, or that an agent can be taught to realize specific navigation tasks. Such possibility allows a robotic agent to learn with its own operation. Kaelbling (1996) points out that without prior knowledge an agent can not learn with effectiveness. Unsupervised learning techniques, as for example reinforcement learning, have a long convergence time and do not provide operational agents from the beginning. Therefore, it is important to mix such methods with supervised ones (Ye et al., 2003; Er and Deng, 2005).

Although miniature like robots, as for instance the Khepera (Mondada et al., 1993), have been used in studies and papers related to autonomous robotics, as in Er and Deng (2005), it is more realistic to perform the same experiments using larger robots due to the dynamic effects associated, which places them closer to real service robots. For this reason, we decide to build a mobile robotic platform with dynamic characteristics that could be applied in a flexible way to navigation and learning experiments. In this sense, the platform allows sensory-motor data to be stored and recovered during or after operation, and new sensors to be added and configured according to the application.

Differently from Ye et al. (2003) and Er and Deng (2005), in our work the supervised learning takes place in a real environment, not in a simulated one, and in real time. The objective is to teach the agent to perform simple navigation tasks using ultrasound sensors.

In order to have incremental learning with some generalization, a radial basis function neural network (RBF) is developed. We adapted the resource allocation algorithm proposed in Platt (1991) for the function interpolation field, to obtain supervised learning in real time, while the robot is teleoperated. In this aspect, our work is also different from Reignier et al. (1997), where the supervised learning is off line, implemented in a GAL ("Grow and Learn") network, with results verified in simulation.

This paper is organized as follows. Section 2 describes the platform and the software framework developed. Section 3 introduces the supervised learning application. Section 4 presents some experimental results that we got until now. Finally, conclusions are drawn in Section 5.

Ribeiro Augusto S. and Ferreira A. (2007).
A ROBOTIC PLATFORM FOR AUTONOMY STUDIES.
In *Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics*, pages 390-395
DOI: 10.5220/0001622603900395

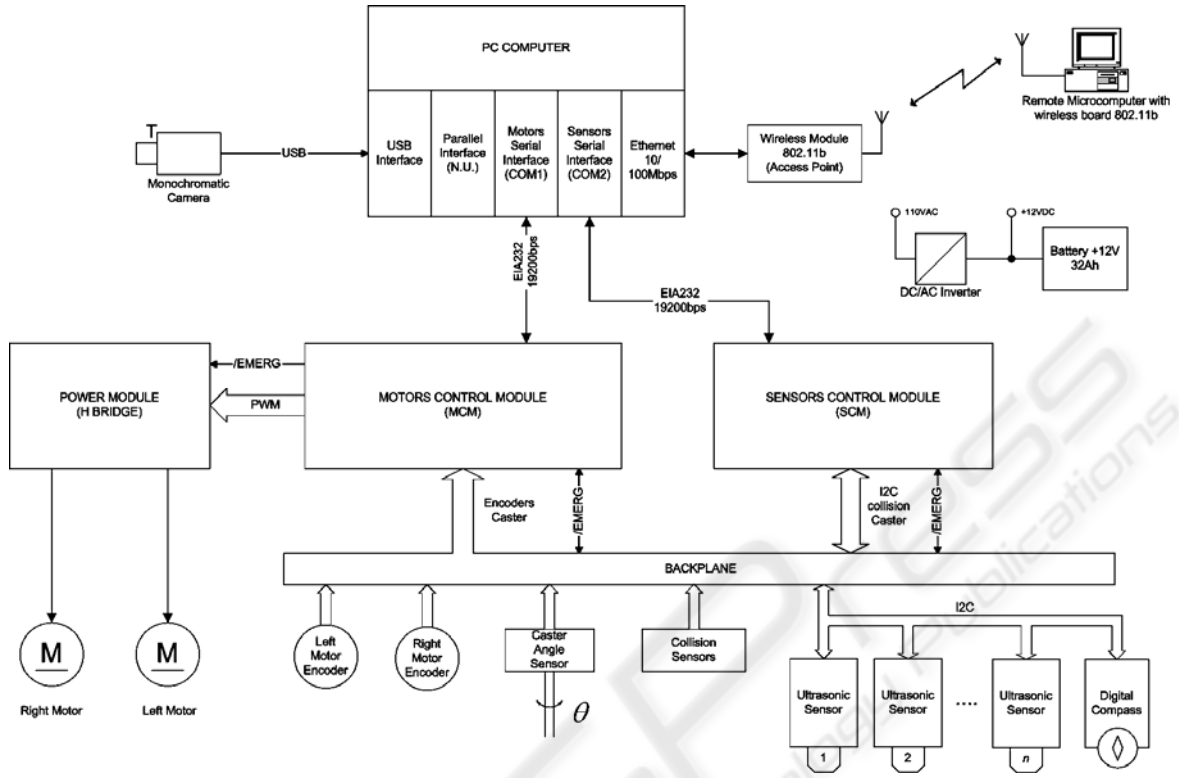

Figure 1: Block diagram of the robotic system.

2 THE ROBOTIC PLATFORM

The robotic platform in its present version is designed for indoor experiments. It measures 50 cm (diameter) by 80 cm (height) and weights 40 kg. In the sequence we describe some aspects of its hardware and software architectures.

2.1 Hardware Architecture

A block diagram of the robotic system is shown in Figure 1. The control of the robot locomotion is accomplished by two motor wheels, powered independently by two DC motors, using differential steering (Dudek and Jenkin, 2000) and a caster wheel. The platform has an image module, and seven ultrasound sensors distributed in its frontal side. The sensors are allocated in a way that objects on the floor can be detected. A digital compass and an angular sensor connected to the caster allow sensorial integration techniques to be exploited to assist in the navigation. Two incremental encoders are used for odometry and velocity control. Collision sensors protect the robot lower perimeter. Other sensors can be added to the platform using a synchronous serial interface available in the system, so that the user can configure it to different types of studies and experiments.

The hardware architecture is arranged in hierarchical processing modules, each one responsible for some of the tasks involved in the mobile robot control. There are four main modules: the Management Module, the Motors Control Module (MCM), the Power Module and the Sensors Control Module (SCM). The Management Module is responsible for the coordination of the robotic unit. Currently a PC on board computer is used for this function. It runs the software framework, described in the section 2.2, for developing of user's applications. The Power Module is responsible for the steering of the motor wheels under control of the MCM. The Motors Control Module implements two PID (proportional plus integral plus derivative) controllers in parallel, allowing independent velocity control of each wheel. The SCM module permits acquiring data from the diverse sensors on the robot. It has a synchronous serial interface (I2C) for

sensors expansion and it can also provide for emergency stopping in case of collision.

The platform also has wireless TCP/IP communication resources, allowing remote monitoring, data exchange and teleoperation. The energy system gives the robot at least one hour of navigation's autonomy.

2.2 Software Architecture

The software environment provides the robot with autonomous navigation as well as teleoperation. The software architecture is divided in two main applications: The On Board Management Software (OBMS), running in the Management Module of the mobile platform, and the Remote Control and Supervision Software (RCSS), executing in a remote microcomputer. The communication between them is made using the Client-Server paradigm (Andrews, 2000) and through the wireless network available in the system.

A block diagram of the OBMS is shown in Figure 2. The architecture is arranged in four main levels: the Communication Level, the Management and Supervision Level, the Execution Level and the Software Interface.

The Communication Level implements a TCP/IP server that is responsible for receiving commands from the remote microcomputer and sending data back to it. Simultaneous connections are possible and data can be exchanged with more than one remote microcomputer if desired. The Management and Supervision Level deals with the commands received at the TCP/IP server, interpreting and executing them. This level also performs the management of the mobile unit concerning its operation mode, autonomous or teleoperated, which is controlled through commands sent by the RCSS. The effective control of the robot is made in the Execution Level, which implements the operation modes. This level is easily adapted to the application required using a library of functions available to the user. Each operation mode has a template which the user can modify or adapt to his own necessities. In the application described in this work, the learning algorithm is added to the teleoperation mode and the learned neural network is recovered and executed in the autonomous mode. The Software Interface isolates the hardware aspects of the robot creating an application program interface (API). This permits that hardware modifications can be made without any change in the other levels of the architecture, supplying modularity.

The software framework has a multithreaded architecture (Andrews, 2000), which is adequate to implement real time applications.

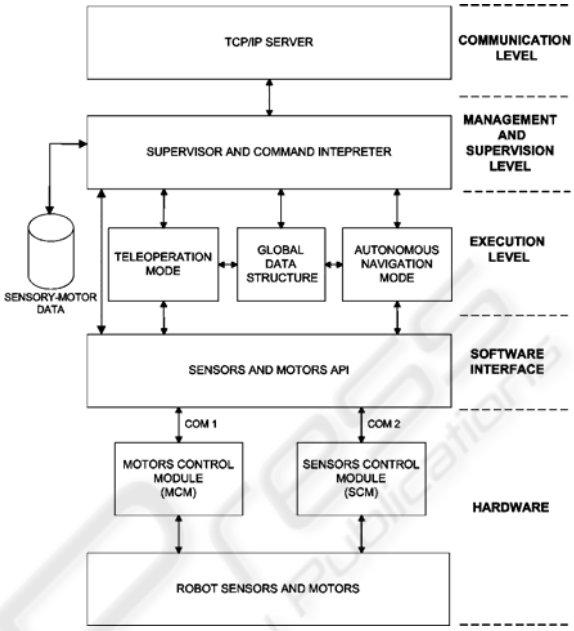

Figure 2: Block diagram of the OBMS.

Sensory-motor data are stored in a data base for analysis and utilization. Sensory-motor coordination aspects (Pfeifer and Scheier, 1997) can then be exploited in the training and learning of autonomous agents. A global data structure allows data exchange among the several software modules in execution.

The RCSS has the main objective of informing the OBMS concerning the operation mode requested by the user. In the teleoperation mode, the robot is controlled through a joystick connected to the remote microcomputer. A TCP/IP client in the RCSS communicates with the OBMS allowing messages and commands exchange.

3 APPLICATION: LEARNING EXPERIMENTS

Using the facilities of the platform, a supervised learning application, assisted by teleoperation, was developed. An RBF neural network was trained in a sequential way appropriate to real time applications. The network starts with no computational units and grows by allocating units (hidden units), or centers, based on the "novelty" of an observation. The novelty is characterized by two joint criterions: the

distance criterion and the prediction error criterion. The former is based on the distance between the input pattern observed and the network units. The latter uses the errors among the desired outputs and the network ones due to the input pattern. The network forms a compact representation and it has a quick learning. Learning patterns do not have to be repeated. The units only respond to a local region of the space of the input values, making easy the incremental learning. If a new pattern is presented to the network and the joint criterion is satisfied, a new unit is allocated. Else, the network parameters are update using the LMS (Least Mean Square) algorithm. Instead of using LMS, an algorithm based on the extended Kalman filter (EKF) has been proposed in the literature (Kadirkamanathan and Niranjan, 1993) to speed up the convergence of the network. Because of the computational complexity involved in the EKF, requiring longer processing time, we decided to use LMS in our real time application. The experimental results in section 4 show that our choice was sufficient for the navigation tests that were realized, allowing the training of the robot to the tasks proposed. It is not our objective in this work to minimize the number of teleoperations for learning, so the speed of convergence of the network is not our main approach.

In our proposal, teleoperations are used for training an RBF network that has the seven ultrasound sensors of the platform as input pattern and the angular velocities of the two motor wheels as outputs. The network parameters are updated in real time during the teleoperation and stored in the end of the training. A new teleoperation can be made with the network starting with the stored parameters. The autonomous mode implements the learned network in such a way that the robot can repeat the task with some generalization. This means that the robot produces coherent outputs for similar inputs, although not equal to those encountered during training. If the performance is not good, the platform allows that new teleoperations can be made, starting from the parameters that have already been learned.

The learning algorithm, adapting from Platt (1991), Kadirkamanathan and Niranjan (1993), is described mathematically as follows, where: $\mathbf{x}(n)$ is the input pattern at the instant n; $y_j(n)$ is the angular velocity desired for each wheel (j=1,2); $s_j(n)$ are the network outputs; $\mathbf{u}_k$ is the unit k of the network; w_{jk} is the weight connecting the unit k to the output j; $\varepsilon(n)$ is the value of the distance threshold in the iteration n that inserts a new unit; ε_{max} and ε_{min} are respectively the maximum and minimum values of $\varepsilon(n)$; e_{min} is the threshold to the network prediction error; k_d is the overlap factor to the network units; γ ($0 < \gamma < 1$) is a decay constant and $\mathbf{u}_{nr}$ is the nearest center to the input $\mathbf{x}(n)$**.** ε_{max} and ε_{min} represent the scale of resolution in the input space, respectively the largest and the smallest scale of interest**.**

The network outputs are written as:

$$s_j(n) = \sum_{k=0}^{m} w_{jk}(n-1)\Phi_k(\mathbf{x}(n)) \qquad (1)$$

where m is the number of units or centers of the network, $\Phi_k(\mathbf{x}(n)) = 1$ for $k = 0$ and

$$\Phi_k(\mathbf{x}(n)) = \exp\left(\frac{-\|\mathbf{x}(n)-\mathbf{u}_k(n-1)\|^2}{\sigma_k^2}\right) \qquad (2)$$

for $k \neq 0$.

Algorithm:

In the first iteration ($n = 0$):
$\varepsilon(n) = \varepsilon_{max}$, $w_{j0}(n) = y_j(n)$ ($k = 0$)
For each observation ($\mathbf{x}(n)$; $y_1(n)$, $y_2(n)$)
{
$e_j(n) = y_j(n) - s_j(n)$
If $|e_j(n)|$ ($\forall j \in \{1,2\}$) > e_{min} and $\|\mathbf{x}_n$-$\mathbf{u}_{nr}\| > \varepsilon(n)$
Allocate a new unit:
$\mathbf{u}_{m+1} = \mathbf{x}(n)$
$w_{j(m+1)} = e_j(n)$
If it is the first unit
$\sigma_{m+1} = k_d\, \varepsilon(n)$
Else
$\sigma_{m+1} = k_d\, \|\mathbf{x}(n)$-$\mathbf{u}_{nr}\|$
Else execute LMS
$\varepsilon(n) = \max\{\, \varepsilon_{max}\gamma^n, \varepsilon_{min}\}$
}

The updating of the network parameters in accordance with the LMS algorithm is given by equations (3) and (4). The time index n was omitted for clarity. Equation (3) is the correction term for the component i of each center k. The dimension of each unit is d, the same of the input pattern (i = 1, 2, …, d). The correction to the weights is given by equation (4). In both equations η is the learning rate.

$$\Delta u_{ki} = \frac{2\eta}{\sigma_k^2}(x_i - u_{ki})\Phi_k(\mathbf{x})\sum_{j=1}^{2} w_{jk}\left(y_j - s_j\right) \qquad (3)$$

$$\Delta w_{jk} = \eta\left(y_j - s_j\right)\Phi_k(\mathbf{x}) \quad (4)$$

4 EXPERIMENTAL RESULTS

Experiments were performed to verify the robot ability to learn the perception to action mapping from the teleoperator in real time, checking if the mobile unit can repeat the navigation task autonomously. The parameters values used in the tests were: $\varepsilon_{min} = 0.03$, $\varepsilon_{max} = 0.5$, $\gamma = 0.9$, $\eta = 0.3$ and $k_d = 0.5$. The input and output data used to train the network were normalized in the range [-1, 1]. Figure 3 shows the results of teleoperating the robot in a corridor with a turn to the right.

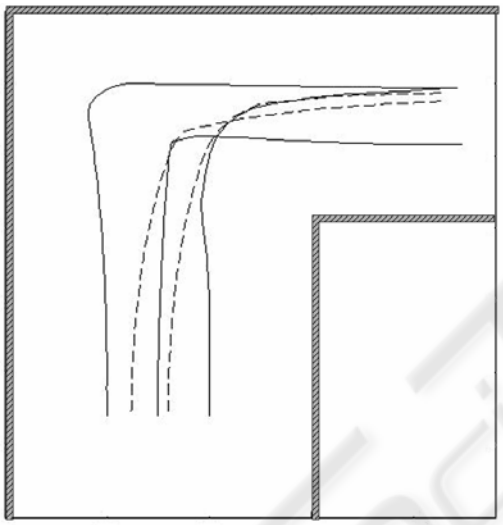

Figure 3: Navigating in a corridor with a turn to the right.

The training consisted of three teleoperations, each one starting at a different initial position. In the figure, the solid lines represent the trajectories executed during the training and the dashed ones are the routes performed autonomously by the robot. The trajectory is the path traversed by the medium point between the two motor wheels. It should be remembered that the unit has a diameter of 50 cm. The training resulted in a network with 121 units.

The idea of the experiments was to demonstrate the capacity of the method in acquiring reactive movements such as wall-following and obstacle avoidance. In Figure 4 the robot goes around an obstacle. Four teleoperations were realized for training, resulting in a network with 119 centers. After the learning phase the autonomous mode was activated and the agent executed the task with success.

The agent was also trained to make a path in the shape of an 8 around two obstacles. Figure 5 shows the test environment.

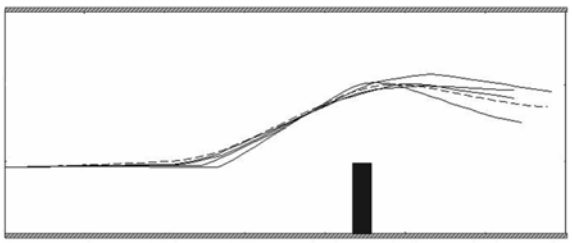

Figure 4: Robot avoiding an obstacle.

Figure 5: Robot in the test environment.

A few teleoperations were realized to train the robot in this navigation task. In the autonomous mode, it was observed that the robot continued surrounding one of the obstacles instead of completing the 8 when arriving at the trajectory cross point. We then added the value of the digital compass available in the robotic platform to the network input pattern, in a way that the agent could infer the direction of the movement during the learning. Such alteration allowed the robot to complete the task autonomously. The initial training allocated 262 units in the network. Sometimes the robot did not complete the navigation task by itself with success. In those cases the autonomous mode was finished and the agent was teleoperated, completing the route. The incremental and local learning characteristics of the neural network allowed new units to be added to the network encompassing such situations. In the end a neural network with 935 units resulted.

In Figure 6 we have some training trajectories (solid lines) and some paths realized by the robot when operating in the autonomous mode (dash and dash-dot lines).

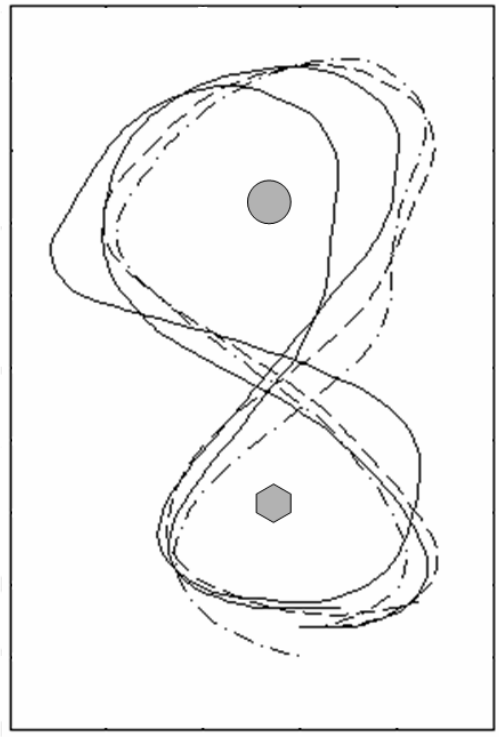

Figure 6: Robot making a path in the shape of an 8.

5 CONCLUSIONS

The mobile robotic platform proposed showed to be efficient to the realization of teleoperated as well as autonomous experiments and studies. The navigation results obtained with the sequential and local learning algorithm used are promising. The results exhibited some generalization, although no specific experiment has yet been made to verify that more systematically. The technique can be applied, for instance, to get prior learning in reactive robotic applications, speeding up real time learning.

As future work, real time pruning techniques should be developed and added to the algorithm to minimize the number of units in the neural network.

REFERENCES

Andrews, G. R. (2000). *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley. New York.

Dorigo, M. (1996). Introduction to the Special Issue on Learning Autonomous Robots. *IEEE Trans. on Systems, Man, and Cybernetics*, Part-B, vol. 26, no. 3, pp. 361-364.

Dudek G. and Jenkin, M. (2000). *Computational Principles of Mobile Robotics*, Cambridge University Press. Cambridge.

Er, M. J. and Deng, C. (2005). Obstacle Avoidance of a Mobile Robot Using Hybrid Learning Approach. *IEEE Trans. on Industrial Electronics*, vol. 52, no. 3, pp. 898-905.

Kadirkamanathan, V. and Niranjan, M. (1993). A Function Estimation Approach to Sequential Learning with Neural Networks. *Neural Computation*, vol. 5, pp. 954–975.

Kaelbling, L. P. (1996). On Reinforcement Learning for Robots. In *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, Osaka, Japan, pp. 1319-1320.

Mondada, R., Franzi, E. and Ienne, P. (1993). Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms. In *Proc. of the Third International Symposium on Experimental Robots*, Kyoto, Japan, pp. 501-513.

Pfeifer, R. and Scheier, C. (1997). Sensory-motor coordination: The metaphor and beyond. *Robotics and Autonomous Systems*, vol. 19, pp. 244-257.

Platt, J. C. (1991). A Resource-Allocating Network for Function Interpolation. *Neural Computation*, vol. 3, no. 2, pp. 213-225.

Reignier, P., Hanser, V. and Crowley, J. L. (1997). Incremental Supervised Learning for Mobile Robot Reactive Control. *Robotics and Autonomous Systems*, vol. 20, pp. 157-178.

Ye C., Yung, N. H. C. and Wang, D. (2003). A Fuzzy Controller with Supervised Learning Assisted Reinforcement Learning Algorithm for Obstacle Avoidance. *IEEE Trans. on Systems, Man, and Cybernetics*, Part-B, vol. 33, no.1, pp. 17-27.