

HOW INTRODUCING ARTIFICIAL INTELLIGENT BEHAVIOURS IN EDUCATIONAL ROBOTICS

Marco Comite¹, Michele Moro²

Department of Information Engineering - University of Padova
Via Gradenigo 6/A-B - I 35131 PADOVA - Italy
(1) *marco@comite.it* – (2) *mike@dei.unipd.it*

Abstract

This paper presents a didactical experience using an Artificial Neural Network on a LEGO Mindstorms NXT robot. Artificial Intelligence is an advanced topic with a broad variety of application fields and robotics is one of the most promising both for practical effectiveness and future developments. Robotics particularly attracts pupils but they get even more interested if the robot can show a certain degree of automatic learning. Educational robotics is a new subject where teachers and researchers are involved in providing methodological and practical frameworks to develop robotic-enhanced project-based activities to support in a new way the teaching/learning of scientific and non-scientific disciplines at different levels. The presented experience deals with a simplified version of the Optical Recognition of Characters using standard sensors and firmware on the NXT and the NXC programming language. The user gives a command to the robot on a small stripe of white paper in form of a handwritten sequence of black/empty squares. Each command learnt during a training phase by an Hopfield Neural Network implemented in the robot program, is associated with a simple motion of the robot. This work was made in the framework of European TERECOP (Teacher Education on Robotics-Enhanced Constructivist Pedagogical Methods) project aimed to define a curriculum for teacher training on educational robotics.

Keywords

Artificial Intelligence, Neural Network, Hopfield Network, LEGO Mindstorms NXT, NXC, constructionism, TERECOP

1. Introduction

Educational robotics is becoming an effective teaching/learning tool to transfer scientific and even humanistic knowledge and concepts to students from the kindergarten to the university [1][2][3]. This is due several aspects. First robotics is intrinsically a multidisciplinary subject, therefore one can create activities not limited to obvious subjects (e.g., mechanics, computer science and programming languages) but can design activities involving arts and humanities, as well. Second, several not-so-expensive robots or robotics kits are on the market now. Last but not least, the research community has proposed (and experimented in several successful cases) novel methodological approaches for the use of robots in education [4][5][6]. Constructivism/constructionism is one of most promising: coming from a well-established pedagogical tradition, with evident links with robotics and artificial intelligence (see the Logo Turtle experience), it has been adopted by the TERECOP project as the fundamental methodological framework to develop a teacher training curriculum and several examples with significant educative value [7][8][9].

It is a common experience that pupils are attracted by (and get emotionally involved with) robots that exhibit reactive behaviours. In most examples, the robot is programmed by hand to react in a pre-determined way to a given situation. However, we realized pupils get even more interested if the robot can show a certain degree of automatic learning. This is because learning is recognized as a human (or animal) high-level intelligent feature. Most automatic learning algorithm rely on Artificial Intelligence (AI) techniques.

Actually AI is an advanced topic with a broad variety of application fields. Among others robotics is one of the most promising both for practical effectiveness and future developments. Moreover robotics is having a new golden age for an increasing interest of the general public, particularly attracted by the

apparent 'intelligent' and reactive behaviour of robots, and the convinced expectation that in less than 20 years robots will be the new basic interfaces between men and computing resources [10]. Speaking about education purposes, AI can be introduced in educational robotics mainly for two reasons: one is to make the student acquainted with AI programming techniques, studying one or more of the several approaches proposed in literature. This is usually reserved to undergraduate university students who have a specific interest and suitable mathematical background to investigate such complex techniques and trying to apply them to new problems. The second reason regards mainly primary and secondary students: it is the aim through educational robotics to give them the basic idea of what intelligence is, or at least in what extension intelligent behaviours can be distinguished from simpler relations of action/reaction. Other issues that are implied by AI are very general concepts like learning, knowledge, reasoning, behavioural strategy etc. that are worth to be investigated and discussed in the classroom.

Typical AI-oriented problem solving is divided into two phases: training and recall. The first phase could be interpreted as a knowledge augmentation whereas the second is the application of the previous knowledge to only partly known or unknown inputs to provide an acceptable response. Regardless of the specific approach adopted, AI algorithms may often cause a heavy computational load that implies the necessity to perform most of the computation on a supporting PC; even when 'trained' algorithms can be run on a very limited hardware platform, sometimes the more demanding training phase must be performed on the PC, or some specific extension or a different operating system are required on the target system to overcome unacceptable limitations.

Our approach was to propose an experience using the same robotic platform adopted by the TERECoP project, i.e. the LEGO® Mindstorms™ NXT [11], performing all the computations, including the training phase, within the robot equipped with its standard firmware. In fact our initial purposes were to demonstrate that AI robot controlling is possible on such kind of limited hardware/firmware targets and at the same time to choose a problem that could resemble a well recognizable animal-human potential.

We decided to use an Artificial Neural Network (ANN) as our AI model, even if other models have been effectively used in educational robotics [12][13][14]. Among the different ANN documented in literature, our choice would have been made considering both the complexity of the problem and the limitations of our robotic platform. The most significant limitation with the standard firmware is that integer arithmetic is supported but floating point is not. This is why many recent experiences with AI and NXT are realized on the basis of leJOS [15][16], which implements a Java ME virtual machine on the target hardware with a larger support of general classes.

Optical Character Recognition (OCR) is a typical hard problem that is affordable with an AI approach. Some kinds of ANNs manifest the necessary robustness to cope with the unavoidable noise in printing (or hand-writing) and the ability to recognize a class of actually different shapes which can be nonetheless associated with the same character. Moreover the human interpretation of printed commands is for sure recognizable as an intelligent behaviour, and therefore it is interesting to 'simulate' such a behaviour in a simple robot. The NXT robot comes with very simple sensors and the only candidate usable for our experiment was the light sensor. Taking into account its known characteristics, as explain in detail section 3, we decided to code each command with a specific sequence of empty/full little hand-written squares on a narrow white paper stripe. The imprecision in making such kind of 'bar codes' gives similar problems as those present in OCR (in this sense, our code can be seen a simplified, 1-D version of 2-D character/words representation). The ANN model chosen was the Hopfield Network [17] which appeared suitable both for the problem level of complexity and the hardware/firmware possibilities.

The reminder of the paper is organised as follows. In section 2, after a brief overview of the hardware/software used in the experiment, the general problem of the handwritten command recognition and the choice made by the authors to simplify the problem to an affordable one are described. Section 3 recalls some features of the Neural Network paradigm and, analysing the restrictions of the NXT firmware support, motivates the choice of the Hopfield model instead of a more general back-propagation one, and describes the implementation. The final section 4 discusses the obtained results and how the experience can be used as a general approach for similar AI-oriented future experiences.

2. An Optical Command Recognition on a LEGO Mindstorms NXT

The LEGO Mindstorms NXT Educational kit includes a 'robot brain' (the brick), some sensors (light, touch, sound, sonar), and some motors. The brick incorporates an ARM7 32 bit processor with 256 KB of flash memory and 64 KB of RAM, USB and Bluetooth connections with a PC and other NXTs, four input ports for sensors, three output ports for servo-motors, an 100x64 pixels LCD display and a loudspeaker. The processor is driven by a firmware implementing a rather powerful virtual machine able to pilot the motors, to sample analog and digital data from the input ports, and to manipulate fundamental data types and structure (boolean and integer variables, both elementary and in multi-dimensional arrays, strings, simple mutexes for synchronization) but, as already mentioned, it does not support floating point.

The LEGO producer provides an iconic-oriented graphical programming environment called NXT-G that permits to easily design and compile robotic programs including conditionals and loops. The compiled program is a bytecode for the NXT VM which must be downloaded to the robot through the USB or Bluetooth connection. When the complexity of the program overcomes a certain threshold, other tools are advisable. Among others we considered the free C-like language called NXC (Not eXactly C) [18] whose compiler produces a bytecode compatible with what generated by NXT-G and thus it can be used with the native NXT firmware. We were aware that the absence of the floating point support, either by a dedicated hardware or by a suitable library incorporated in the standard firmware, would have limited the possible choices to a small number of ANN types and obviously influenced the precision of all the calculations performed in the network.

Considering the general OCR problem, the 2-D shape representing the character to be recognized is usually pre-elaborated to partly reduce noise and the obtained matrix (or string) of pixels is the input of a recognizing algorithm, whereas the character code is the output. Due to the limited reproducibility of the same character and the variety of layouts the same distinguishable character may assume, particularly when hand-written, a training phase is required during which the algorithm, an ANN or any other type, must make an adaptation induced by the inputs that will assure the quality of the classification of the read characters. Therefore an OCR system is a typical automated procedure with a training phase; moreover it is rather known because available in consumer scanner and PDA software. It is a state-of-the-art function with good performance, at least for printed characters. For hand-written text, the current recognizing systems require that the user writes either separated single characters or single words following a pre-defined structure to obtain an acceptable reliability (80-90% of accuracy); the general problem of hand-written text recognition is still a research issue.

Following these remarks, we could consider OCR as an interesting case of machine learning with a practical application. But it must be necessarily simplified to permit the use of the standard NXT light sensor. In fact this sensor can emit a red light and measure a light intensity, i.e. a gray scale, around its sensitive element within a pre-established range of values, either raw (unscaled) between 0 and 1023 or in the scaled percentage (Fig. 1), and with a limited spatial precision.

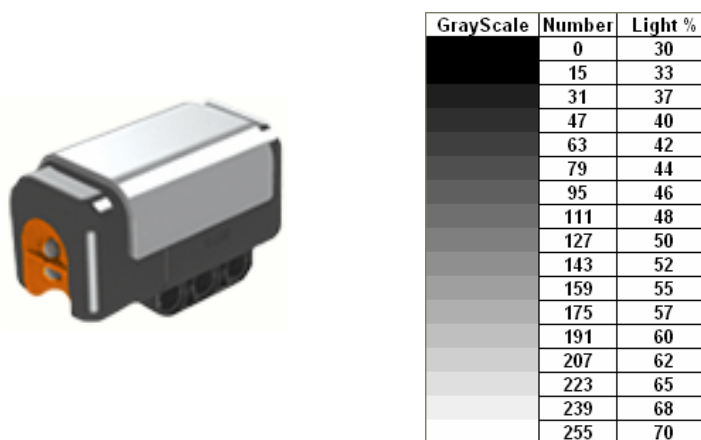


Fig. 1 – The NXT light sensor and an experimental table of values read by the sensor placed 4 mm over a laser printed sheet with the sequence of gray levels on the left of the table (Number is the common value of the three RGB components)

Therefore we defined a 1-D structure constituted by a sequence of 'fillable' squares (Fig. 2): such a sequence is the layout of a simplified 'character' representation that will be recognized by our ANN and associated with one specific robot action. The user is required to fill a subset of squares with a dark pen accordingly to the chosen command. This Morse-like command code must be prepared on a narrow white paper stripe so that it can be easily dragged by the robotic system with a fixed speed under the light sensor. What we expected is that the ANN would have cope with the imprecision in the square filling (some squares can be filled overflowing either of the two borders, or only partly filled, see Fig. 3) provided the acquisition system can sample the black/white sequence with a sufficient spatial frequency.

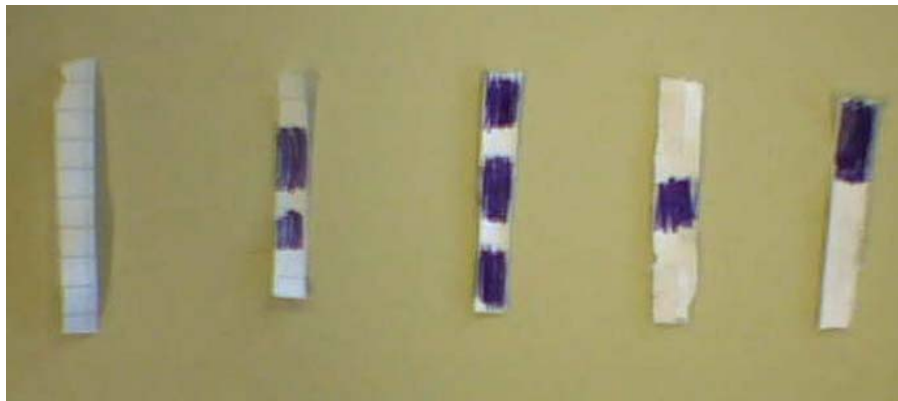


Fig. 2 – The command stripes

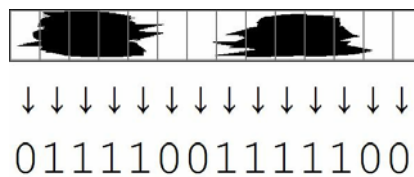


Fig. 3 – Stripe acquisition

3. The ANN chosen and its implementation on the NXT

An ANN is the artificial counterpart of what we have in our brain and nervous systems. Structurally it may be thought as an interconnection of computational nodes called 'neurons': each neuron may have several inputs and usually one output. The output value, which can be binary or numeric, is in general a non linear function (called 'transfer function') of the weighted sum of the input values; such weights can change during an adaptation or training phase and represent the acquired knowledge of the problem domain (Fig. 4). The way how the neurons are connected one another and the algorithm used during the training phase are specific of the various kinds of ANNs. After a 'correct' training phase the ANN is able to give reasonable responses even in case of not already seen inputs, provided the training phase had produced the storing of 'generalization rules' that can recognize clusters of inputs with some kind of similarities: in this case it produces a single output representative of the cluster. This potential of clusterization is the expected effect of the training procedure.

Current literature shows that one of the best choices for a precise handwriting recognition is the so called 'back-propagation' network. This kind of ANN is made of one input layer, one or more hidden layers and one output layer; there is a full forward connection between each node of a layer (apart the last one) and each node of the following layer. The name of this type of ANN recalls the fact that during the training phase the output values backward feed the updating of the connections' weights. In the supervised form, i.e. when during the training phase you can present the correct outputs corresponding to the training input patterns, the updating procedure tends (or should tend) to minimize an error function and the training can stop when such kind of stability is reached. This would be also

our case because the training is performed knowing the correct response when a certain 'character' input is presented to the network.

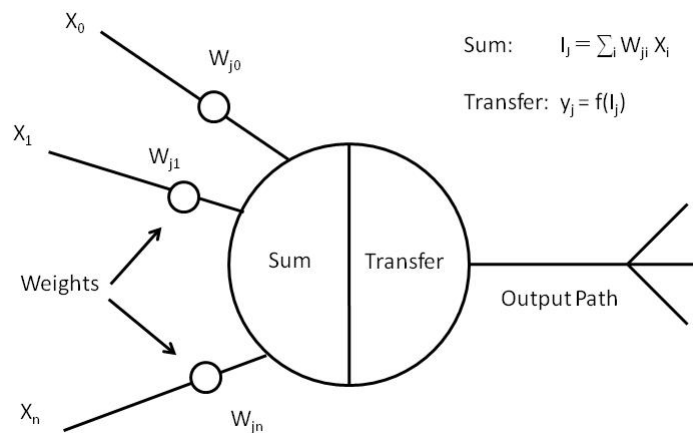


Fig. 4 – The artificial neuron general model

Unfortunately the well known and diffusely used back-propagation networks were not suitable for our experience mainly for the absolute necessity to use floating point calculations. Moreover the 1-D simplification of the input sequence allows the adoption of simpler ANN architectures without relevant risks to obtain unsatisfactory results. After a brief analysis, we chose the Hopfield Network (HN) considering the following properties: it can use only integer numbers, its structure is rather simple and therefore its behaviour is more simply understandable, and its computational requirements more limited.

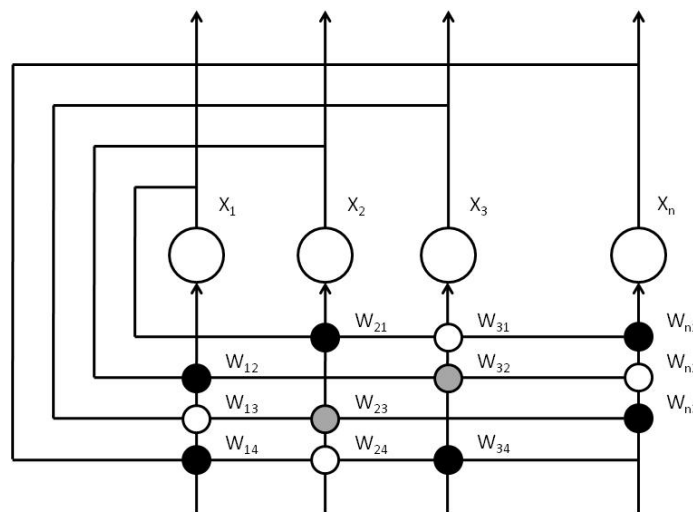


Fig. 5 – The typical connection matrix of a Hopfield Network

In HN all neurons are interconnected each other except themselves, therefore it can be considered formed by only one layer and the transfer function is a step function (Fig. 5). Each connection between the output node j and the input of node i has a weight w_{ij} expressing the strength of the influence of the output of the node j towards node i . Weights are necessarily symmetric (i.e. $w_{ii}=0$, $w_{ij}=w_{ji}$). Any neuron has an activation binary value (its output) which is on when the associated input sum is over a given threshold. When the network is learning a new pattern, its weight matrix must be updated: when the output of a node i is the same of the input j , the connection w_{ij} will be strengthened, otherwise it will be weakened. Because the current values of the weights represent the maintaining of the acquired knowledge, during the recall phase the network makes a sort of similarity check to establish the class of the given pattern. If it seems belonging to a known class, the network returns its representative pattern; otherwise it declares the pattern unknown and the user can decide to improve the knowledge of the network training it with this new pattern.



Fig. 6 – The robot of the experiment

Because of the demonstrative purposes of the experience, the robot is built as a simple pivoting vehicle (two wheels with one motor each) (Fig. 6) connected through its ports to a motorized reading extension where you can put the paper stripe to be analyzed (Fig. 7). We defined the following operative procedure. The program is usually waiting for a triggering signal by the user (he/she has to push on of the button of the brick), then it powers the motor of the acquiring extension on to drag and read the stripe. A certain number of samples from the light sensor is acquired and such samples are given as inputs to the network. When the pattern is similar to one of the already known and coded patterns, the robot executes the associated action, otherwise it considers the pattern as a completely new one and asks the user to provide the corresponding new action. The current version includes a set of five possible simple actions (forward, backward, turn right or left for a while, or sound a beep) but this set can be easily extended. To signal possible error conditions, the user has five seconds either to confirm the correctness of the executed action or to make a correction through the action menu.

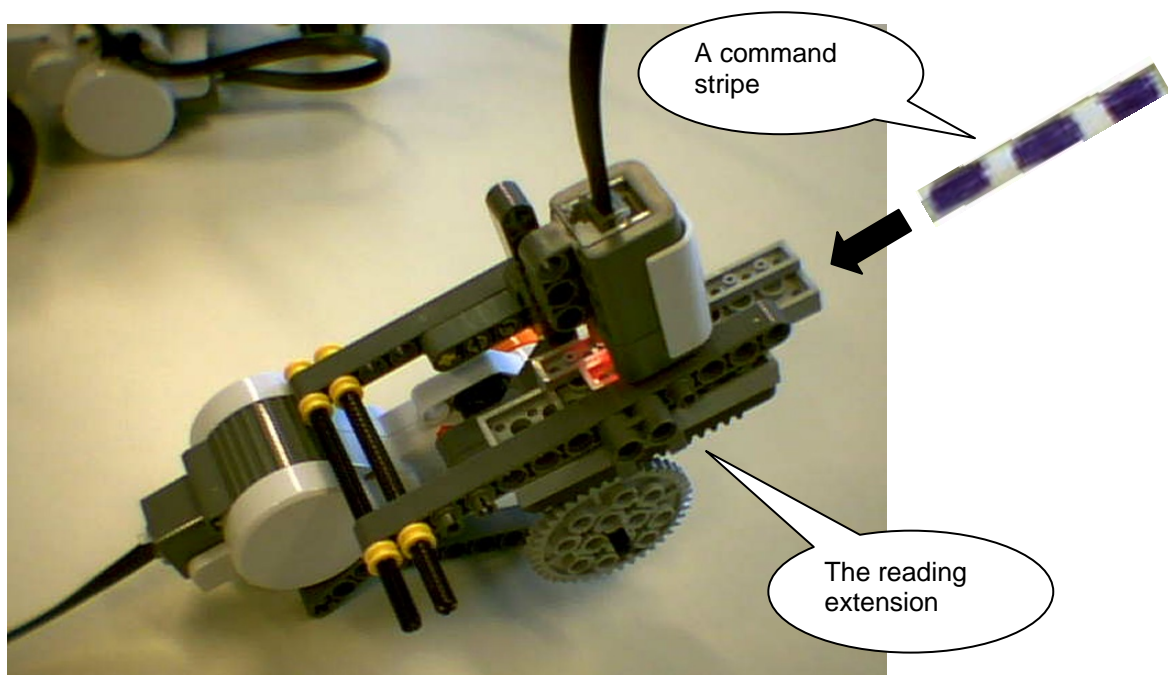


Fig. 7 – The acquisition system

We have calibrated the number of read samples (currently 20), which gives also the size of the network, in order to perform the weight updating calculations within an acceptable time. Limiting to 4-5 squares the structure of the hand-written code, the size of the input pattern of samples gives a sufficient margin to guarantee good performance for the network without renouncing to a certain variety of allowed commands. Though the robot memory is initially empty, to facilitate the practical use of the experiment, new acquired patterns are also stored in a file (the NXT firmware supports a very simple form of file system) together their associated actions so that the network can be re-learned every time the program is restarted without reading again the already read stripes.

The NXC code is a modified porting of a free implementation of an HN in C language, designed and realised by Karsten Kutza, able to recognize simple 2-D images [19]. Main adaptations were, as already noted, the 1-D layout of the input samples, and some language differences and limitations of NXC with respect to C.

4. Discussion and conclusions

One of the main objectives of the TERECoP project is to develop a framework for teacher education courses in order to enable teachers to implement the robotics-enhanced constructivist learning in school classrooms. Specific attention is devoted by the project partnership to investigate both methodological and practical issues and to develop critical examples for using in a constructivist way with teachers mainly of secondary level. Though practical emphasis was given to NXT-G implemented examples to promote the transition between robotics as a learning object to robotics as a learning tool, we are developing several demonstrative examples covering different target disciplines, different ages of students, different approaches. So some experiences require programming tools more flexible than NXT-G, particularly when you must perform many calculations involving several variables and/or not elementary data structures. This was the case of the experience presented in this paper which may be used at two different levels.

For younger practitioners it represents a way to introduce discussions about human learning and machine learning, about the real meaning of knowledge and how to recognize intelligent behaviors. In this case the teacher can simply present a general description of the experiment, follow the instructions to build the robot (or an equivalent one) and try to teach the robot some commands. Pupils are asked to prepare some handwritten paper stripes, explaining why sometimes the robot asks the user to input a new choice about the action to be executed. In case the teacher has a sufficient programming skill, he/she can adapt the code to other kind of inputs modifying some configuration parameters but maintaining substantially unaltered the network structure.

On another level, students able to program in NXC can analyze the code and then make further modifications to adopt the network to different scenarios. This is also a way to investigate how implementing a machine learning and to have a deeper awareness of the representation of knowledge which is one of the most critical aspects of AI.

On the one hand future works could verify the applicability of other AI methods with the same standard NXT firmware (e.g. genetic algorithms), on the other hand some advanced experiences in the literature, developed with Java and leJOS on the NXT platform and therefore usually suitable for older students, could suggest different affordable approaches with significant educative value even for the primary/secondary levels of school. This could also give the basis for introducing more elaborated robot architecture, e.g. 'animaloids' and humanoids, when they will be available at consumer level.

5. Acknowledgments

This paper was partly based on work done in the frame of the project "Teacher Educations on Robotics-Enhanced Constructivist Pedagogical methods" (TERECoP) funded by The European programme Socrates/Comenius/Action 2.1, Agreement N° 128959-CP-1-2006-GR-COMENIUS-C21 2006-2518/001-001 S02.

References

- [1] Portsmore, M. (1999). ROBOLAB: Inuitive robotic programming software to support life long learning. *APPLE Learning Technology Review*, pp. 26–39.
- [2] Maja, J.M. (2004). Robotics Education for All Ages, *Proceedings, AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education*, Palo Alto, CA, Mar 22-24.
- [3] De Michele, M.S., Demo, G.B., Siega, S. (2008). A Piedmont SchoolNet for a K-12 Mini-Robots Programming Project: Experiences in Primary Schools, *Proceedings of the SIMPAR 2008 Intern. TERECoP Workshop "Teaching with robotics: didactic approaches and experiences"*, Venice, (Italy), Nov. 3rd, pp. 90-99.
- [4] Resnick, M. (2002). Rethinking Learning in the Digital Age. In *The Global Information Technology Report: Readiness for the Networked World*, edited by G. Kirkman. Oxford University Press.
- [5] Carbonaro, M., Rex, M. and Chambers J. (2004). Using LEGO robotics in a project-based learning environment. *The Interactive Multimedia Electronic Journal of Computer-Enhanced Learning* 6(1) (<http://imej.wfu.edu/articles/2004/1/02/index.asp>).
- [6] Resnick, M., Martin, F., Sargent, R., and Silverman, B. (1996). Programmable Bricks: Toys to Think With. *IBM Systems Journal* 35, 3, 443-452.
- [7] Alimisis D. et al. (2007). Robotics & Constructivism in Education: the TERECoP project, *Proceedings of the 11th European Logo Conference*, <http://www.eurologo2007.org/>, 19 - 24 August, Bratislava, Slovakia.
- [8] Frangou, S., et al. (2008). Representative examples of implementing educational robotics in school based on the constructivist approach, *Proceedings of the SIMPAR 2008 Intern. TERECoP Workshop "Teaching with robotics: didactic approaches and experiences"*, Venice, (Italy), Nov. 3rd, pp. 54-65.
- [9] Arlegui J., Menegatti E., Moro M., Pina A. (2008). Robotics, Computer Science curricula and Interdisciplinary activities, *Proceedings of the SIMPAR 2008 Intern. TERECoP Workshop "Teaching with robotics: didactic approaches and experiences"*, Venice, (Italy), Nov. 3rd, pp. 10-21.
- [10] Gates, B. (2007) A Robot in Every Home, *Scientific American*, January 2007.
- [11] Lego Mindstorms web site, <http://mindstorms.lego.com>
- [12] Schrum, J.B. (2004). Study and comparison of genetic algorithms when applied to lego mindstorms robots, *The Journal of Computing Sciences in Colleges* 19(4): 353-353.
- [13] Vamplew, P.W. (2004). Lego Mindstorms Robots as a Platform for Teaching Reinforcement Learning, *AISAT2004: International Conference on Artificial Intelligence in Science and Technology*, Hobart, Australia, 21-25 November.
- [14] KLASSNER, F. (2002). A case study of LEGO Mindstorms suitability for artificial intelligence and robotics courses at the college level. *SIGCSE'02*, February 27- March 3, Covington, Kentucky, USA. p 8-12.
- [15] leJOS web site, <http://lejos.sourceforge.net/>
- [16] Vlist, B.v.d., et al. (2008). Teaching machine learning to design students. In Z. Pan, X. Zhang, A. El Rhalibi (Eds.), *Technologies for E-learning and digital entertainment: third international conference, Edutainment 2008*, Nanjing, China, June 25-27, pp. 206-217, Berlin: Springer.
- [17] Russel, S., Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*, 2nd edition, Prentice Hall.
- [18] NXC web site, <http://bricxcc.sourceforge.net/nbc/>
- [19] A C-language implementation of an Hopfield network, <http://www.neural-networks-at-your-fingertips.com/>