A Case Study on Goal Oriented Obstacle Avoidance

Pasquale Caianiello and Domenico Presutti
DISIM
Università dell'Aquila
Italy
Email: pasquale.caianiello@univaq.it, domenico.presutti@gmail.com

***Abstract*—We report on several test experiments with a mobile agent equipped with an artificial neural net control to achieve a basic route direction goal reflex in a 2-dimensional environment with obstacles. A real assembled *4tronix Initio* robot kit agent is reproduced with its sensor and motor characteristics in a virtual environment for experimenting and comparing the behavior of its artificial neural net control with two different learning approaches: A standard supervised error back propagation training with examples, and an unsupervised reinforcement learning with environmental feedback.**

I. Introduction

Obstacle avoidance is a basic task for mobile agents. Commercial and research applications address the problem by using state of the art adaptive techniques and mathematical modeling. In this work we confronted with the task of using a simple neural net architecture to equip a real *4tronix Initio* [22] robot kit agent with a basic obstacle avoidance control. Its neural net would take inputs from a pre-processing of the sensor information of the robot, and provide an output to control its motor actuators. The preliminary aim of our project, as reported in this paper in its start-up phase, is to construct the virtual counterpart of the *4tronix Initio* robot agent, that will let us perform fast and reliable test experiments of possible control strategies at the reflex proactive level with real time response.

As a result we identified a simple artificial neural net architecture and a pair of training strategies that would let the agent show simple adaptive/reactive capabilities in avoiding obstacles, while achieving a given target position goal. The agent's control neural net model is deliberately kept at a reflex level state, the perceptron basic input/output transduction, with no high level onthologies or primitives for describing the environment, no environment model or map acquisition capabilities, and no planning ability of any sort. The environment only exists for the pattern that it induces on the agent sensors at a given position and orientation. The agent will have to react by issuing control settings to its motor components, taking into account its given goal.

II. Tools and Methods

A. The agent

The agent hardware is a *4tronix Initio* [22] robot kit controlled by a Raspberry Pi B+ [25] computer that emulates the artificial neural net and performs low level sensor acquisitions and issues motor control primitives to a PiRoCon v.2 motor controller. The agent is equipped with a pan-tilt HC-SR04 ultrasonic sensor that acquires a panoramic of the environment, two infrared front mounted close range proximity sensors and a GY-80 multi-chip module that integrates a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis digital compass used for determining the agent absolute orientation.

B. The virtual agent and environment

The virtual environment is constructed as a bit matrix of dimension $m \times n$. Each bit represents a virtual position for the agent. A bit is set to 1 if the position is blocked, there is an obstacle, and the position cannot be occupied by the agent. The *goal* area of the environment is a circle identified by a center position and a boundary radius. The virtual agent emulates the real hardware in performing its basic control commands to the hardware controllers. The basic functioning cycle has three steps: Sample sensors, Compute orientation and velocity, Run for a time unit.

C. The artificial neural net control module

The artificial neural net (ANN) is emulated through the use of the Neuroph [24] Java framework used to implement a 3-layer fully connected feed-forward net with 31 input units, 62 hidden nodes, and 10 output units as depicted in fig 1. All artificial neurons in the net are sigmoid. The 31 input units collect the infrared proximity information, the distances in 9 predefined pan positions measured by the ultrasonic sensor (normalized in the unity interval), the distance from the agent to the center of the goal area, and the computed orientation angle with respect to the given direction goal, and represented into 18 binary direction intervals of 20° to comprise the whole 360° range. The output units consist of 9 binary orientation directions (the front 180°) and one real in the unity interval to represent the distance to be covered. So the control protocol will interpret the ANN output by setting the agent to the orientation given, and let it run for the given distance.

The experiments are described when the ANN is trained according to two different protocols: Supervised error Back Propagation (BP), environment reinforcement learning (RL).

1) The Supervised error back propagation protocol: The BP protocol was experimented with training sets (TS) constructed in two different ways, the first (BPp) by sampling the control choices of a human pilot, the second (BPh) by recording the behavior choices of a heuristic evaluation function in a wide range of enumeration of input sensor patterns. The use of BPh allowed the easy construction of much larger virtual TS, while TS construction with BPp required synchronized reading

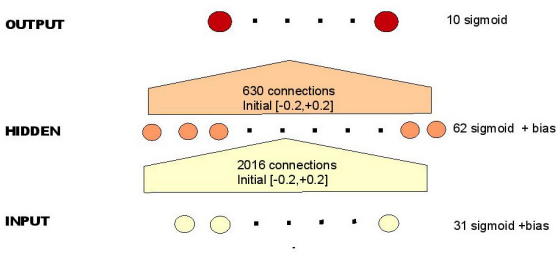

Fig. 1. The perceptron architecture for proactive reflex input/output transduction

the agent sensors and sampling the human pilot, who on the other hand was allowed to comprise higher level cognitive faculties, and was allowed to look at the environment map while making his decision.

To preserve generalization capabilities and avoid overfitting of the ANN, the training process was stopped at predetermined network mean square error limit values. For this purpose, each training sets were split in two subsets of training subsets and test subsets, containing respectively 90% and 10% of the original TS samples. Optimal limit network error values were determined by training the network on training subsets and testing network response on the test subsets: when the error on the test subsets became stationary or increasing, training was paused and finally completed by using the full TS and the minimum network error limit.

After the training process, the trained ANN's were tested on the agent control system and some critical aspects emerged as of the dimensional insufficiency of the TS obtained with BPp when compared to the dimension of the inputs state configurations. It did, in fact, bring about insufficient network output response polarization on new input patterns and a random-like behavior of the agent in specific configurations. On the other hand the ANN's trained with TS constructed with BPh was often trapped in stationary or cyclic behavior in sub-optimal positions with respect to the navigation goal. In consideration of the complementary critical aspects described for the BPp and BPh cases, a third instance of the ANN was trained with an incremental training process that combined both pilot driving and heuristic evaluation TS. In this case the learning process consisted in two training phases. In the first phase, the BPh TS was used for training the ANN for a small number of training epochs in order to give the network base response capabilities in covering a wide range of input configurations. In the second phase, the BPp TS was used until training completion. As the incremental trained network was finally tested on the robot, the critical behaviors were relieved.

All the test results reported in the following are obtained by running the net configuration obtained at the end of the training process as described, with a sample environment problem.

2) The Reinforcement Learning protocol: The same net architecture has been used with an unsupervised reinforcement training protocol Q-learning [15] with reward/reinforce function taking into account distance from goal, runs length, and route declination from goal direction. Reward was corrected by the Q function [15] to take into account future effects of actions

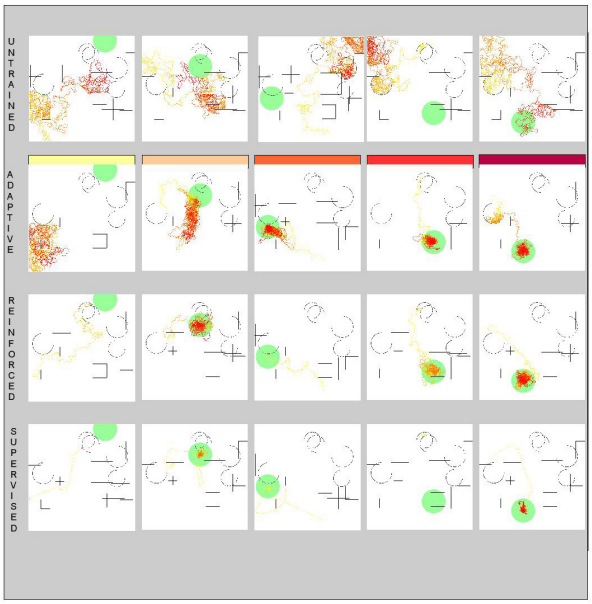

Fig. 2. Simulation Results, Environment complexity 3. Each snapshots records the trajectory of the corresponding row agent. Trajectory position points have a time scale color starting from yellow and going to darker red as time passes. Goal area is green. See text.

and to loosen excessive local/opportunistic behavior. The net was trained on a collection of problem samples with random selection of obstacles configuration, starting position and goal area to obtain the trained net that was used in the comparative experiments where its behavior is sampled at different levels of maturation.

III. Experimental Results

The experimentation was performed after training the ANN both with the BP protocol and with the Q-learning protocol leading to two net configurations named SUPERVISED (SU) and REINFORCED(RE) respectively.

The RE network was trained for 4000 learning sessions of 100 base cycles. For each session a random start and target is assigned in a randomly generated environment. Main learning parameters are set by an empirical optimization process to the following values: Learning rate= 0.036, Future actions discount factor g= 0.24, and Stochastic action selector temperature T=4. A high temperature T value is necessary during the learning process to maximize reinforcements. The T value is subsequently lowered on tests to 0.4 value to appreciate neural network response and control system behavior.

On the other hand, SU is trained for one learning epoch on the BPh TS, containing 1.152.000 training samples, and resulting in a 0.14 mean square error after training. The following 2076 training epochs are performed with the BPp TS, with 400 training samples. Mean square error at the end of the training process is 0.07. Both BPh and BPp training sets are generated on several base generation sessions of 50 iterations each. Network weights are initalized with random values in the [-0.02, 0.02] interval.

After training the nets are tested on a battery of several tests

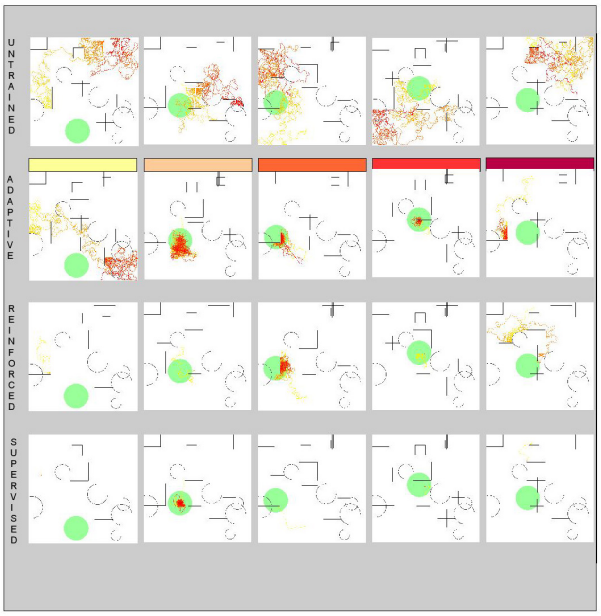

Fig. 3. Simulation Results, Environment complexity 7. Each snapshots records the trajectory of the corresponding row agent. Trajectory position points have a time scale color starting from yellow and going to darker red as time passes. Goal area is green. See text.

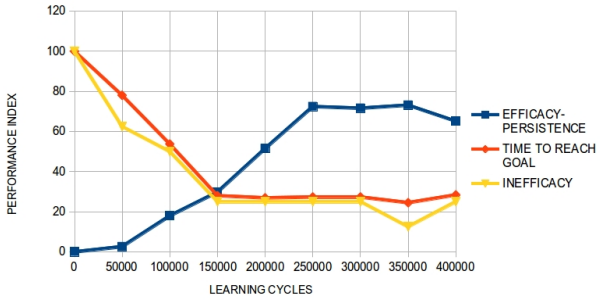

Fig. 4. Performance of RE while training

on the same environment, organized in groups of increasing environment complexity.

Their behavior, while trying to achieve the target area goal, is sampled as reported in fig. 2 and in fig.3. The whole experiment ranges over 8 batteries of 10 random problems in the same environment, for 10 different random choices of start/target positions. In the figures we show the outcome of just two test batteries, where each row collects an order preserving under-sampling of five out of the ten snapshots in the battery, each representing the trajectory of the agent behavior in the same environment. Snapshots of RE and SU behavior in test problems are in the bottom two rows. The top two rows records the agent behavior when controlled by an UNTRAINED (UN), randomly selected net configuration, and when controlled by a Q-learning ADAPTIVE (AD) net. AD is always in learning phase, it is randomly initialized at the first test in the battery, and retains its net configuration through subsequent tests in the battery. As AD gets trained while testing, it is expected to converge to the one of RE.

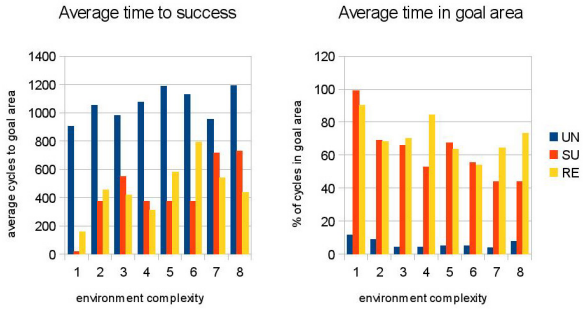

Fig. 5. Comparative statistics of reflex effectiveness in reaching the goal

Each snapshots records the trajectory of the corresponding row agent. Trajectory position points have a time scale color starting from yellow and going to darker red as time passes. Test problems in a row are presented in the same order as they were performed. The order is irrelevant for UN, RE, and SU, but AD's behavior changes (and improves) while solving a problem. For subsequent tests in the battery, AD's behavior change gives an idea of how a Q-learning net evolves to a finally trained RE from an UN.

Figure 5 reports a statistics over the tests performed in a single test time of 1600 iterations. The first index indicates *ineffectiveness* on task achievement, obtained by measuring the time taken for the agent to reach the goal area. High values of ineffectiveness are generally associated to wandering behavior or stationary dead ends encountered during navigation. The second index computes effectiveness and persistence, by measuring time percentage spent inside the goal area after the area is reached. Low values of the index are generally associated with excessive random behavior or fortuitous goal area achievements. The tests are performed on increasing environment complexity levels, and then growing time needed to success. UN network shows negative performances on all tests conditions, while SU network shows the best performances especially into low complexity environments, with a low number of obstacles, moving straight to the goal area in a few number of cycles, basically focusing on target. RE shows best performances particularly in high complexity environments, proving better explorations capabilities and abilities to overcome stationary configurations.

Figure 4 reports performance statistics indexes over increasing learning time of AD neural network from 0 to 400.000 learning cycles, increasing by a 50.000 interval. The progressive trend is evident and demonstrates the adaptive capabilities of the reinforcement learning protocol. Positive performance indexes show a clear increasing trend, while negative performance indexes show a decreasing trend. Performance charts show acceleration between 100.000 and 200.000 learning cycles, with inflection points in this interval, and a final stabilization after 250.000 learning iterations.

A.

IV. Conclusion

We presented simulations of the behavior of a mobile agent equipped with a neural net reflex-like control in avoiding ob-

stacles and achieving a given target position goal. At this stage of the project we use no high level onthologies or primitives for describing the environment, no environment model or map acquisition capabilities, and no planning abilities. We implemented the artificial neural net control with two different learning approaches: a standard supervised error back propagation training with examples, and an unsupervised reinforcement learning with environmental feedback. We constructed both a real robot agent and a virtual agent-environment simulation system, in order to perform fast and reliable test experiments. The virtual environment let us perform advanced integrated training and test sessions with progressive complexity levels and random configurations, leading to a high grade of generalization for the neural net control. We collected statistical data on several test experiments and compared the performance of the two learning approaches. The analysis of the control system critical aspects and capabilities, as observed in the simulations, favored fixing and improving data presentation in the training protocol.

REFERENCES

[1] Anvar A.M., Anvar A.P. (2011). *AUV Robots Real-time Control Navigation System Using Multi-layer Neural Networks Management*, 19th International Congress on Modelling and Simulation, Perth, Australia.

[2] Awad H.A., Al-Zorkany M.A. (2007). *Mobile Robot Navigation Using Local Model Networks*, World Academy of Science, Engineering and Technology.

[3] Bing-Qiang Huang, Guang-Yi Cao, Min Guo (2005). *Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance*, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou.

[4] Chen C., Li H.X., Dong D., (2008). *Hybrid Control for Robot Navigation - A Hierarchical Q-Learning Algorithm*, Robotics and Automation Magazine, IEEE, 15(2), 37-47.

[5] Floreano, D., Mondada, F. (1994). *Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot*, Proceedings of the third international conference on Simulation of adaptive behavior: From Animals to Animats 3 (No. LIS-CONF-1994-003, pp. 421-430), MIT Press.

[6] Floreano D., Mondada F. (1996). *Evolution of homing navigation in a real mobile robot*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 26(3), 396-407.

[7] Janglova D. (2004). *Neural Networks in Mobile Robot Motion*, International Journal of Advanced Robotic System, Institute of Informatics SAS, vol. 1, no. 1, pp. 15-22.

[8] Glasius, R., Komoda, A., Gielen, S.C. (1995). *Neural network dynamics for path planning and obstacle avoidance*, Neural Networks, 8(1), 125-133.

[9] Medina-Santiago A., et. al. (2014). *Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors*, Instituto Tecnolgico de Tuxtla Gutirrez, Chiapas, Mxico. pp. 104-110

[10] Michels J., Saxena, A., Ng, A. Y. (2005). *High speed obstacle avoidance using monocular vision and reinforcement learning*, Proceedings of the 22nd international conference on Machine learning (pp. 593-600), ACM.

[11] Milln J. (1995). *Reinforcement Learning of Goal-Directed Obstacle-Avoiding Reaction Strategies in an Autonomous Mobile Robot*, Robotics and Autonomous Systems, Volume 15, Issue 4. pp. 275299.

[12] Na, Y.K., Oh, S.Y., (2003). *Hybrid control for autonomous mobile robot navigation using neural network based behavior modules and environment classification*, Autonomous Robots, 15(2), 193-206.

[13] Pomerleau D.A. (1991). *Efficient Training of Artificial Neural Networks for Autonomous Navigation*, in Neural Computation3: 1. pp. 88-97.

[14] Rogers T.T. , McClelland J.L, (2014). *Parallel Distributed Processing at 25: Further Explorations in the Microstructure of Cognition*, Cognitive Science 38, 10241077.

[15] Rummery G.A., Niranjan M. (1994). *On-Line Q-Learning Using Connectionist Systems*, Cambridge University.

[16] Tsankova D.D. (2010). *Neural Networks Based Navigation and Control of a Mobile Robot in a Partially Known Environment*, Mobile Robots Navigation, Alejandra Barrera (Ed.), ISBN: 978-953-307-076-6, InTech.

[17] Ulrich I., Borenstein J., (2000). *VFH*: Local Obstacle Avoidance with Look-Ahead Verification*, International Conference on Robotics and Automation, San Francisco, CA, 28, 2000, pp. 2505-2511

[18] Yang G.S., Chen E.K., An C.W., (2004). *Mobile robot navigation using neural Q-learning*, Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on (Vol. 1, pp. 48-52), IEEE.

[19] Yang S.X., Luo C. (2004). *A neural network approach to complete coverage path planning*, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(1), 718-724.

[20] Yang S.X., Meng M. (2000). *An efficient neural network approach to dynamic robot motion planning*, Neural Networks, 13(2), 143-148.

[21] Floreano D., Mattiussi C. (2002). *Manuale sulle reti neurali*, Il Mulino, Bologna.

[22] 4tronix website, $http://4tronix.co.uk/$

[23] *HC-SR04 Ultrasonic Ranging Module*, Iteadstudio, $http://wiki.iteadstudio.com/Ultrasonic\_Ranging\_Module\_HC-SR04$

[24] *Neuroph Framework*, Neuroph website, $http://neuroph.sourceforge.net/$

[25] RaspberryPi website, $http://www.raspberrypi.org$