INTELLIGENT ADAPTIVE LEARNING IN A CHANGING ENVIRONMENT

Guillaume Valentis[1] and Quentin Berthelot[2]

[1,2]Embedded Systems, ECE Paris Engineering School, France
valentis@ece.fr and berthelo@ece.fr

ABSTRACT

In order to develop ever more intelligent and autonomous systems, it is necessary to make them self-learning, since it is impossible to include in their program everything they may encounter during their life-cycle. In this research work, we aim at answering the following: if a system's environment is modified, how could the system respond to it quickly and appropriately enough? We achieve it by using reinforcement learning to allow the system to rate its decisions, then by developing adaptive learning algorithms for gain and loss rewards. The algorithms include probabilities' analysis providing to the system ability to adapt its knowledge through time and to respond to a changing environment. Simulations are made for a robot finding its exit in a labyrinth. Results show that reinforcement and adaptive learnings can have many useful applications by offering to a system a reliable possibility of evolution within complex environments in specific situations.

KEYWORDS

Reinforcement Learning, Neural Network, Autonomous Systems, Adaptive Learning, Changing Environment

1. INTRODUCTION

Present project concerns artificial intelligence in autonomous systems such as robots, and more precisely intelligence based on experience usually called reinforcement learning [1-5]. After development of high level mechanical capabilities for these systems with corresponding stability for trajectory following and robustness for handling adverse environment effects [6], embodiment of further cognitive capabilities became evident in a next step for higher efficiency with much larger adaptive response based on strong and redundant sensory background [7,8]. Within this extended setup, considerable attention is nowadays paid to giving autonomous systems access to "decisional" level for mastering themselves their own action [9,10]. Expectably, the most available results are obtained by usually expensive and time consuming processes often disqualifying real time response [11]. So here attention is focused on research of much shorter access to decision based on immediately available and global observations. The approach followed here is, contrary to some approaches which somehow antagonize environment change, to take instead advantage of this change when time elapses, implying that the system must adapt its knowledge. So the system should be learning during its whole life cycle because of environment changes happening around it. On the other hand, for all man-made autonomous and intelligent systems, one still cannot predict all situations they will encounter during their life cycle. Therefore, one must grant them some freedom to make their own decisions, and one possibility is that they may learn based on their gain consecutive to each particular decision

David C. Wyld et al. (Eds) : COSIT, DMIN, SIGL, CYBI, NMCT, AIAPP - 2014
pp. 217–225, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.4921

instead of other possible ones [12]. Rating the decision taken with higher or lower reward will enable the system to conclude on the decision having been good, average or bad, and thus to change its experience-based knowledge according to the actual gain provided by the environment (either the environment or an operator – human or machine). Therefore, the system would learn from its actions and correct its decisions from its mistakes [3,4,13].

The autonomous system must perceive the environment around it in order to make a decision before taking any action [3]. To illustrate the problem, the example of a robot in a labyrinth is considered here, where the overall path pattern stays the same, but an intersection can become an exit or a dead-end and conversely as time evolves.

2. THE MODEL

A system can learn in an unknown environment, thanks to reinforcement learning. Indeed, this consists in learning the pattern from experience: it has to face situations with rewards provided by the environment [3,14]. Typically the basic conceptual representation of such a situation is a tree-like structure where the system has to choose a branch to follow at each branching point for continuing its evolution. In a totally unknown environment with branching, the probability of taking a path instead of another one at an intersection is the same the first time the system encounters it. Thus, for m possible paths there is a probability $1/m$ to take each of them (it is a random neutral situation). Let the system choose a path and predict a gain upon taking this path. Its reward is the difference between final real gains and expected one. Depending on this reward, the probability to take again the same path will increase or decrease [15]. Here comes the interesting part: indeed the system is left learning for n times, n representing the number of times it would have made a complete journey in the environment (from the start to one end). After these n times, if the reward is consequently modified, in usual approach the system would take time to change its knowledge (if it is able to do it). Therefore the first statement made here is that no path can achieve a probability value equal to 0, and the least possible probability is fixed at a threshold value $P_{th} = 0.05$. It means that the system is allowed to take this decision 0.05 of the occurrences, giving it the possibility to explore this path and see if something new happened (if the reward has changed) or not since the previous visit. Therefore the maximum probability becomes:

$$P_{max} = 1 - 0.05 * (m - 1) \quad (1)$$

Then it is assumed that when a positive reward occurs, the gain curve must grow faster than the current probability to take this path. Indeed, taking again the example of a labyrinth explored by a robot, the robot has a 0.05 rate to go left where there is a dead-end, and has a 0.95 probability to go straight where it finds an exit. At some point it is decided to exchange the exit and the dead-end. When the robot takes the left path (0.05) with this very low rate one can predict that situation is worse than the straight path where there exists a 0.95 rate to take it. But once the path has been travelled the robot arrives at an exit, and the reward is positive. Therefore the probability has to be adjusted quickly to this change, but after certain limit the adjustment must be slower in order to stabilize the system. Thus one gets the following gain algorithm:

$$P_i = P_i + (1 - (m - 1) * 0.05 - P_i) * G * \varepsilon \quad (2)$$

Where Pi represents the probability at the previous intersection to take this path, m the number of different possibilities, G represents the reward (here only a positive reward can occur) and ε a learning coefficient (the smaller is the coefficient, the finer the learning curve is). The response time depends on ε, which has to be between 0.95 and 0.05 otherwise the program fails. Indeed, if $\varepsilon > 0.95$ or $\varepsilon < 0.05$, the gain can be superior to the maximum or inferior to the minimum

probability and make the system fail. We also need to consider the maximum gain factor as follows:

$$\varepsilon < P_{\max} / G_{\max} \quad (3)$$

With these elements, probability curves can be calculated for various values of ε. For $n=10$ runs with a gain $G=1$ one gets:

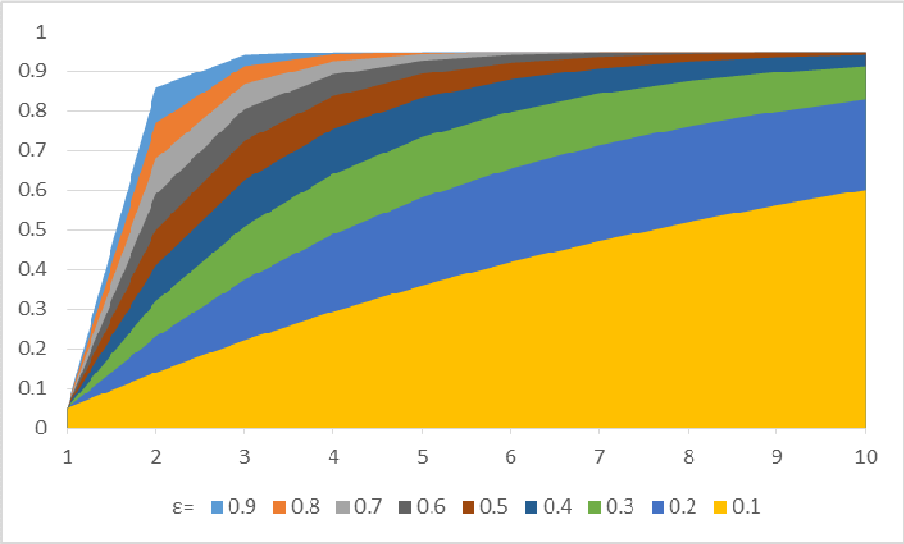

Figure1. Gain Probability Curves vs n for Different ε

On Figure 1, it is clearly seen that for $n=10$ the probability is around 0.6 for $\varepsilon=0.1$ and is already 0.9 for $\varepsilon=0.3$. This means it is a fast response for a radical change made on the environment (which happens when the exit and dead-end positions have been interchanged). Consequently this curve represents the gain curve only if the robot takes the path with the exit (previously a dead-end).

One should also provide a strong loss curve that may represent what is expected. When a loss occurs, the probability of uninteresting states is around 0.5, and one should make it change quickly and most smoothly around the extremities (0.95 and 0.05) as possible to have clear choices. Indeed, assuming that left path is a dead-end and straight path is an intersection, after n runs, the probabilities attained are respectively 0.05 and 0.95. If the guess of going straight is an exit and one gets an intersection, the reward is negative but actually it is not a so bad situation, therefore the probability must not change too much. Respectively the same problem occurs around 0.05 probability: if the loss was too high, the gain curve would not be strong enough to reverse the rate, even if this way was actually better.

The probability curve given in Figure 2 is split into two parts: the upper part (≥ 0.5) and the lower part (<0.5).

For $Pi \geq 0.5$:

$$P_i = P_i + (1-(m-1)*0.05-P_i-0.01)*G*\varepsilon \quad (4)$$

And for $Pi < 0.5$:

$$P_i = P_i + (P_i - 0.05) * G * \varepsilon \quad (5)$$

Where *Pi* represents the probability at previous intersection to take this path, *m* the number of different possibilities, *G* represents the reward (here only a negative reward can occur) and ε a learning coefficient (the smaller the coefficient is, the finer the learning curve becomes but the longer the training period is). Combining (4) and (5) in a digest algorithm with a programming test statement one gets:

$$P_i = P_i + ((P_i \geq 0.5)\,?(1 - 0.05*(m-1) - P_i - 0.01) : (P_i - 0.05)) * G * \varepsilon \quad (6)$$

Similar to gain probability curves, probability loss curves representation is shown on Figure 2 for various values of ε for $n=10$ runs with same gain $G=1$.

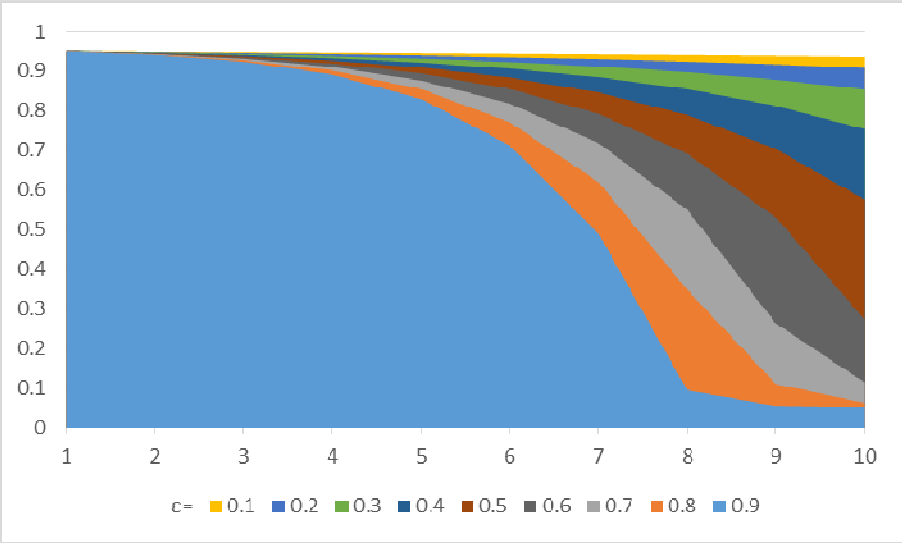

Figure 3. Loss Probability Curve vs *n* for Different ε

On Figure 2, it is clearly seen that for probability around *0.5* the changes occurs quickly while near *0.95* and *0.05* the curve decreases slowly. From both Figures 1 and 2, large (resp. small) probability values are reached faster with larger value of ε because slope is steeper due to higher convergence.

2. APPLICATION AND SIMULATION

Previous model will be applied to a labyrinth. Indeed this model is simple to understand, to study and to observe, but yet complex enough and on which one can easily make changes on the environment. All tests are performed on personalized software using algorithms explained in previous parts. The system first has to explore the labyrinth, the intersections when first met and get the random neutral probability for each possible way. It only knows the intersection at which it was located previously and its current position. The values of gain *G* used in simulation are:

- 0 for a dead-end;
- 1 for an intersection;
- 2 for an exit.

For the labyrinth the following pattern is used.

Table1. Symbols for the Different Labyrinth States

START EXIT WALL

And the system is represented by a small dark blue square. At each intersection the probability to take a specific way is recorded, and for all analysed examples the value is used to speed up the calculation.

As a first example, the basic two-way intersection is considered.

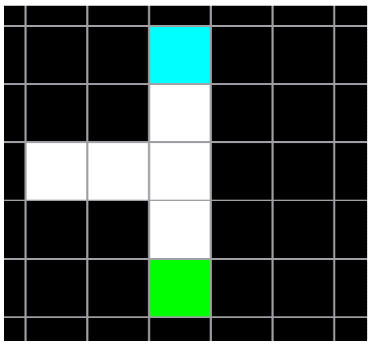

Figure3. Example 1 – Simple Intersection

As seen on Figure 2, there is a starting position, an intersection with two ways, on the left a dead-end and an exit standing forward. An exit is the highest gain while the dead-end is the lowest, so optimal observation is obtained with least runs.

Obtained probabilities at each sample are shown on Figure 4, on which it is observed that the system learns that it is clearly better to go forward at the intersection, therefore the probability to choose this way goes rapidly as high as *0.94*.

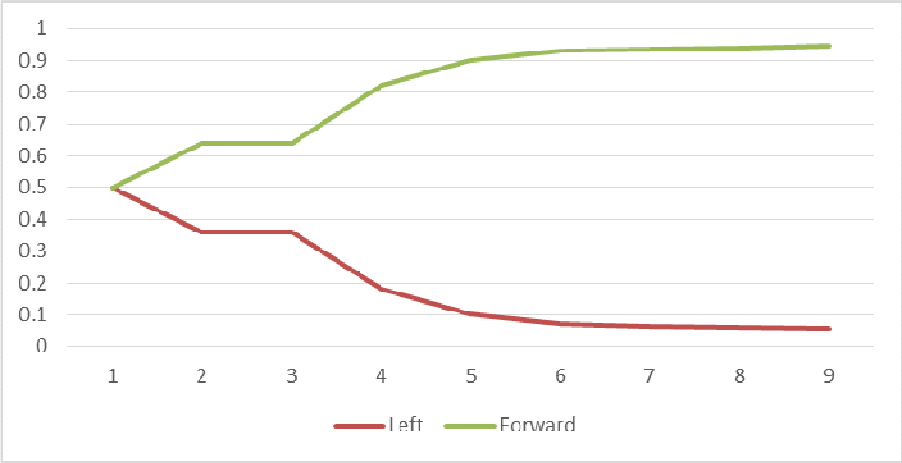

Figure4. Probability Curve until *0.95* Value for Taking the Exit (Forward Way) is Reached

Now the exit and the dead-end are exchanged while keeping the knowledge the system has already learned, see Figure 5.

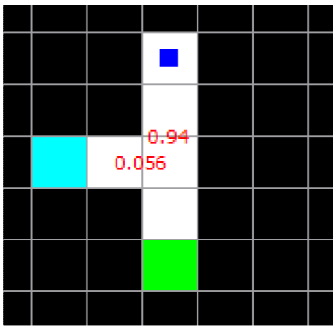

Figure5. Example 1 – Exit Switched with Dead-End

Evolution of probabilities in displayed on Figure 6.

Figure 6. Probability Curve vs Tries Number from Switched Initial Positions after Figure 5

It is seen that previously reached values are weakly modified up to the fourth try, where they drop down from *0.91* to *0.4* and raise steeply from *0.08* to *0.6*, and continue after a stop to increase further up to the same reversed values *0.94* and *0.06* they started from an initial state. Interestingly, the inversion is performed in the same number of tries (about 9) as the one the system had realized from its departure state. This indicates a strong robustness property of the proposed algorithm to the initial conditions, the convergence being here only dependent on parameter ε (at least for current value of ε).

For further testing analysis, the system maze is made more complex as shown on Figure 7. It still has a starting position and a two-way intersection (left-forward) but while the left leads to a dead-end again, the forward path leads to a two-way intersection (forward-right) with respectively an exit and a three-way intersection with, in successive order, an exit, a dead-end and a two-way intersection (right-forward) with an exit and a dead-end.

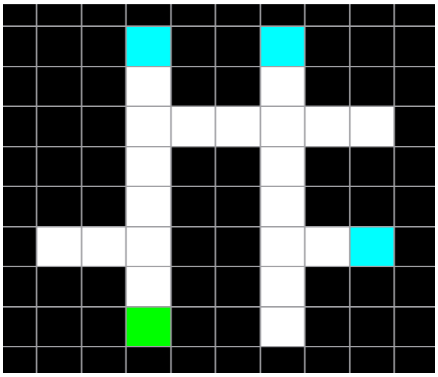

Figure7. New Testing Maze

System's probabilities when it browses the maze are displayed on Figure 8 vs run number.

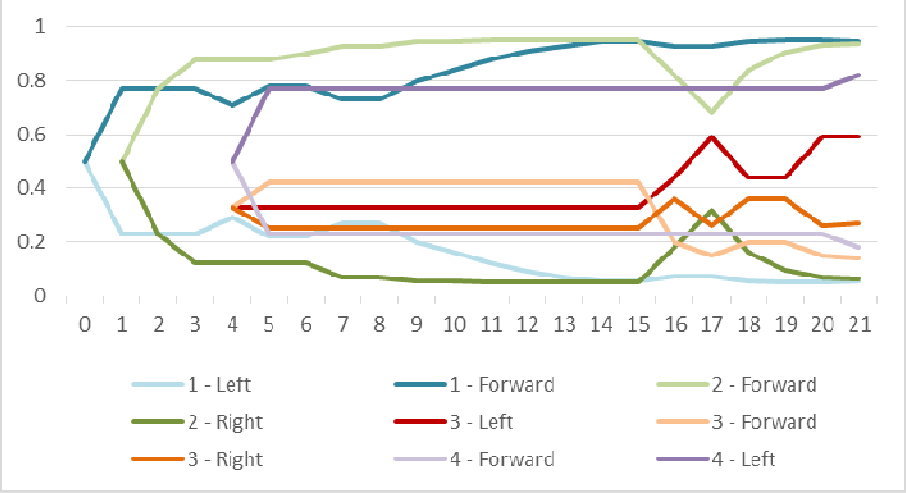

Figure 8. Probabilities Evolution vs try Number for the Different Paths

It is observed that all the different paths are not reached during the first run. This is logical since some paths are located after a possible dead-end or exit. Around the 15th run, abrupt changes are occurring on previously rather smooth curves. The probability limits are letting a chance to the system to look for other possible "good" paths. The curves are smoothing back to previous values as the results of exploration are impacting on system factor decision. On a general setting, many possible paths never reach their probability limits because the decision to take such path does not lead to the highest gain, and therefore it does not give a maximum chance for choosing it. The fact that some paths lead to other paths and so on in cascade, means that the probability tree will spread and give even less chance for each individual path. Final stabilized probabilities after the 21 runs are shown on Figure 9.

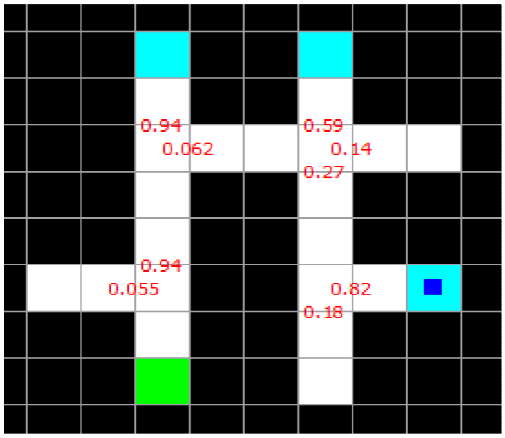

Figure 9. Stabilized Probability Distribution for the Various Paths after 21 Tries

From the probability values, clearly it is assumed that the system gets a better understanding of environment as it is nearby the start position, with a probability as high as *0.94* for the first gallery. This is to be compared with probability values of *0.59* for the first side exit and *0.82* for the second one once in the second gallery, which is explored with a probability of only *0.062*. So as it stands, the system over-weights neighbouring exits. Alternatively, observed dissymmetry is the mark that the system "remembers" its past in that it does not explore the same space to construct its trajectory from initial distribution. This implies the system has gained a further adaptation capability of the learning process for a better labyrinth structure understanding, and already stresses the efficiency of the simple proposed algorithm to rapidly orientate system exploration towards the most "efficient" exit.

4. CONCLUSION

In this paper, starting from reinforcement learning, we develop an adaptive learning method that provides to a system-robot the ability to evolve by its own and to take decisions depending both on its past experience and on changes occurring in its environment. The method is based on algorithms that use probabilities' analysis in order to incorporate gain and loss rewards.

According to this method, simulations are carried out to study the behaviour of the system-robot seeking the exit(s) of two types of maze: a simple one and a more complex one. The results of the simulations show that the system adapts smoothly and robustly when the positions of exit and dead-end are modified. It acquires a better understanding of the environment and its changes. Its possible behaviour when facing the different options, as interpreted by the probabilities, seems coherent with what could be expected from a human being.

Though simple, the tests here displayed of robot behaviour in a labyrinth are showing that reinforcement and adaptive learnings may have useful applications by giving to a system, with modest investment, reliable possibility of evolution within more complex environments in specific situations.

ACKNOWLEDGEMENTS

The authors express their gratitude to Dr F. Fauberteau for devoted coaching, to Pr. C. Duhart for guidance, to Pr. M. Cotsaftis for preparation of the manuscript, and finally to ECE Paris School of Engineering for having provided the setup in which the project has been developed.

REFERENCES

[1] L.P. Kaelbling, M.L. Littman, A.W. Moore, (1996) "Reinforcement Learning", a Survey, J. Artificial Intelligence Research, Vol.4, pp.237-285.

[2] S.F. Smith, C. Heng, H. Xi, (2003) "A Reinforcement Learning Approach to Production Planning in the Fabrication Fulfilment Manufacturing Process", Proc. Winter Simulation Conference, Vol.2, pp.1417-1423.

[3] T.C. Kietzmann, M. Riedmiller, (2009) "The Neuro Slot Car Racer: Reinforcement Learning in a Real World Setting", Proc. Intern. Conf. on Machine Learning and Applications (ICMLA '09), pp.311-316.

[4] Guang-Yi Cao, Bing-Qiang Huang, Min Guo, (2005) "Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance", Proc. Intern. Conf. on Machine Learning and Cybernetics, Vol.1, pp.85-89, Guangzhou, China.

[5] P. Gerard, M. Butz, O. Sigaud, (2003) "Forward and Bidirectional Planning on Reinforcement Learning and Neural Networks in a Simulated Robot", Springer-Verlag, Ch.11, pp.179-200.

[6] Bin Yao, (1996) "Adaptive Robust Control of Nonlinear Systems with Application to Control of Mechanical Systems", PhD Thesis, Mechanical Engineering, UC Berkeley.

[7] I. Kanellakopoulos, (1993) "Passive Adaptive Control of Nonlinear Systems", Intern. J. Adaptive Control and Signal Processing, Vol.7, pp.339-352.

[8] S. Sastry, (1989) "Adaptive Control: Stability, Convergence and Robustness", Prentice Hall, Englewood Cliffs, NJ.

[9] M.R. Endsley, K.A. Ericcson, N. Charness, P.J. Feltovich, R.R. Hoffman (Eds.), (2006) "Expertise and Situation Awareness", The Cambridge Handbook of Expertise and Expert Performance, pp. 633–651, Cambridge Univ. Press.

[10] Haibo He, (2011) "Self-Adaptive Systems for Machine Intelligence", Wiley and Sons, New York.

[11] K. Kavi, R. Akl, A. Hurson, (2009) "Real-Time Systems: an Introduction and the State of the Art", Wiley Encyclopedia of Computer Science.

[13] K.O. Stanley, R. Miikkulainen, (2002) "Efficient Reinforcement Learning through Evolving Neural Network Topologies", Proc. Genetic and Evolutionary Computation Conference (GECCO-2002), pp.569-577, San Francisco.

[12] C. Gonzalez, V. Dutt, (2011) "Instance-Based Learning: Integrating Sampling and Repeated Decisions from Experience", Psychological Review, Vol.118 (4), 523-551.

[14] B. Bonet, H. Geffner, (2006) "Learning Depth-first Search: A Unified Approach to Heuristic Search in Deterministic and non-Deterministic Settings, and its Application to MPDS", Proc. 16th Int. Conf. on Automated Planning and Scheduling (ICAPS-06), pp.3-23.

[15] E. Mizutani, S.E Dreyfus, (1998) "Totally Model-free Reinfor-cement Learning by Actor-critic Elman Networks in non-Markovian Domains", Proc. IEEE World Congress on Computational Intelligence WCCI'98, pp.2016–2021, Alaska, USA.

AUTHORS

Guillaume Valentis is an engineering student at ECE Paris on his last year. He follows courses in embedded systems and robotics. He has studied at Concordia University and Dublin Business School.

Quentin Berthelot is an engineering student in ECE Paris, and is to earn his master's degree next year. He is studying embedded systems and robotics. He has attended classes at Concordia University and at UCI.