10th International Conference on Higher Education Advances (HEAd'24)
Universitat Politècnica de València, València, 2024
DOI: https://doi.org/10.4995/HEAd24.2024.17362

Automatic code assessment in Robotics higher education courses

Lía García-Pérez[1], David Roldán[2], José M. Cañas[2]
[1]Universidad Complutense Madrid, Spain, [2]RoboticsLab Universidad Rey Juan Carlos, Spain.

How to cite: García-Pérez, L.; Roldán D.; Cañas J.M. 2024. Automatic code assessment in Robotics higher education courses. In: 10th International Conference on Higher Education Advances (HEAd'24). Valencia, 18-21 June 2024. https://doi.org/10.4995/HEAd24.2024.17362

Abstract

This paper describes two automatic assessment tools developed for Unibotics, a robot programming web platform for engineering education. The first one, for style assessment, measures some quality indicators in the source code itself such as complexity, number of loops, and PEP8 compliance. A second tool runs the robot application developed by each student and measures its performance when solving the task of each exercise. For students, both tools provide instant feedback which encourages them to improve their score and so their source code. For teachers, they are automatic assessment tools which provide additional information for student evaluation. Both tools aim at improving the technology enhanced learning when using that robotics education platform. They have been used in two courses at Universidad Rey Juan Carlos with 40 real students and some preliminary analytics have been collected.

Keywords: *Educational robotics, web-based learning, robotic simulators.*

1. Introduction

Code assessment has been an active topic in education for a long time. The assessment of programming assignments is not a trivial task. There are many characteristics to take into account, compiling and executing students' programs require a lot of time, which is worse if the number of students is high. For this reason, instructors cannot assess all characteristics of the assignments or have to perform a more superficial assessment. Without adequate feedback, students rarely have the opportunity to learn from their mistakes. When programming, students need instant feedback to improve, however giving immediate feedback is not a simple task. There are many errors that could be solved before submitting a piece of code with the appropriate feedback such as, syntax and semantic error detection, software metrics analysis, structural similarity analysis, keyword detector, plagiarism, diagram analysis, infinite loops logical errors or unused statements. These errors can be detected automatically but since many universities still check their student's code manually, some of them are overlooked (Striewe & Goedicke, 2014).

Rees (1982) suggested that by making programming issues visible and measurable for students they learn to pay attention to them. Based on his work, many authors developed code analyzers for programming languages such C, Ada, Fortran, etc and even relatively new programming languages such as Python have a module to evaluate the programming style of a given code. Important for any learning process, assessment can guide the students' learning providing feedback to both the student and the teacher. In this sense, automatic code assessment may help the teacher to evaluate their students' code in a fair and objective way. In addition, it will allow the students to do code checking more frequently (Yang, Liu, & Yu, 2018). There are many studies in the literature that provide assessment techniques and assessment tools to give student's appropriate and automated feedback for them to learn (Ihantola et al., 2010).

Regarding the teaching of robot programming, one of the main evaluation criteria is whether the robot succeeds in performing the task for which it was programmed and, if so, with what degree of efficiency. Robotic competitions typically use similar criteria to rank the participants in an objective and quantitative way, they have tools for evaluating the effectiveness of a robot's code and make fair comparisons between attendant's robot programs. Robotic competitions such as RoboCup, ARIAC, MBZIRC and Darpa Robotics Challenge have traditionally been a key element in research and in higher education. Also, VEX Robotics Competition, FIRST LEGO League, RoboCup Junior, World Robot Olympiad are examples of interesting competitions in secondary education where serve as a stimulus for students. Several illustrative examples may be found in Evripidou et al. (2020). Many of the competitions are with physical robots and recently some of them are also with simulated robots, which may simplify the automatic evaluation.

In this work we present the automatic code evaluation system implemented on the Unibotics platform. Unibotics is an open collection of exercises to learn robotics in a practical way, based on Gazebo simulator and ROS middleware. It includes an automatic code effectiveness evaluator and an automatic style evaluator. The system has been tested with 40 real engineering students and some preliminary results are also described.

2. Automatic code assessment of robot programming

The question of code quality assessment was born along with the programming. Code developed in a certain programming language to solve a specific problem can be assessed using three different questions: (1) Functionality: Does the developed code solve the problem? (2) Code Style: Is the developed code the most suitable for the proposed problem (speaking in terms of efficiency, re-usability, easy to read and understand for other programmers...)? (3) Algorithm: Taking into account a more abstract issue: is the algorithm implemented by the code adequate and effective to solve the problem?

Functionality is the easiest to assess. A key question here is a correct and precise definition of the problem proposed to students and the measurable requirements for the solution. Beyond robotics competitions, in some robotics areas some standardized quality metrics and benchmarks have already been proposed and are widely accepted. For robot navigation (Xia et al., 2020), for visual object detection in robot perception systems the accuracy and recall computed on certain datasets (Padilla et al., 2020), for self-localization algorithms the error on some SLAM datasets (Huletski, Kartashov, & Krinkin, 2015), for autonomous driving the combination of route completion and the infractions penalty (Yumaganov & Agafonov, 2021).

The programming style, or coding conventions, is an important aspect when writing code, since it can make the written program intelligible to other programmers. However, code style assessment could be less objective. Nevertheless several authors describe many style of coding criteria that any programmer agrees with. The style rules of programming are indeterminate, inconsistent and contradictory (Roque et al., 2019. Different ways of using factors such as indentation, alignment and comments can be found in several books that address the topic. Lack of consensus can be traced to the lack of definition of what exactly is programming style, and what factors contribute to its definition.

Third question intends to assess a more abstract and complex idea, if the algorithm behind the code is the most adequate to the proposed problem. Maybe the code solves the problem, the style is perfect but the algorithm that the code implements is inefficient or to complicate for a simple problem.

An interesting example of program assessment in a robotics context is described by Siegfried et al. (2017). The authors used simulated robots and provided real-time feedback and gamified hints to the students, who reduced the average time to write a correct program and the percentage of them successfully writing a correct program also increased.

3. Unibotics platform for educational robotics

Unibotics is a robot programming web platform with engineering higher education contents (Roldán-Álvarez et al., 2023). It allows the robot programming from the web browser and uses state-of-the-art robotics tools such as ROS middleware and Gazebo simulator. It provides more that 20 academic units on service robotics, autonomous driving, drones, computer vision and mobile robots.

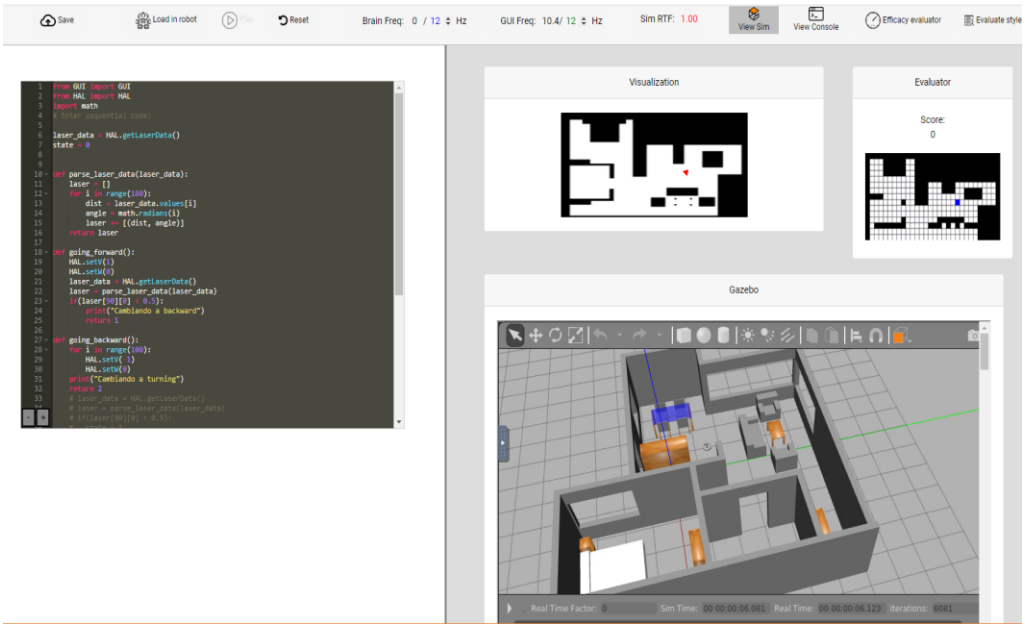

Figure 1. Example of the Vacuum Cleaner exercise web page

The platform provides several challenges where the student has to program the robot in Python to solve a specific challenge (such as cleaning a full room with a vacuum cleaner) in a simulated Scenario. For each exercise Unibotics already provides the Scenario, the robot sensors and actuators through ROS topics, and the task to be solved.

The web page of each exercise is divided in three main parts as it is shown in Figure 1. In the left side an inline text editor is be used to write the robot program. The right side includes a view of the simulated world to see the robot behavior, a debugging console for text messages and several exercise-specific widgets to show debugging information such as the processed images. Then in the top area of the web page there is a toolbar which is used to implement the basic operations of the platform: saving the current code, loading the code into the robot, running the simulation, resetting it, showing and hiding some GUI widgets and two buttons to evaluate the efficacy of the code and the programming style.

4. Automatic assessment tools in Unibotics

One of the main advantages of the platform is that the user is offered the possibility of evaluating the code automatically and receive feedback.

Regarding the *style of the code*, the platform uses the rules from the PEP 8 style guide. This evaluation offers information about how well-written the code is. This is an area that, in our

case, many students at the university often do not pay attention to when programming, focusing only in if the code works and solve the issue, leading them to write unintelligible code that can not be maintained nor updated in an easy way.

Regarding the *efficacy evaluation*, it heavily depends on the exercise itself. Two of them will be described here for illustrative purposes. First, in the FollowLine exercise the challenge is to program an autonomous F1 Car to complete a lap of the circuit in the shortest possible time by following a red colored line drawn throughout the circuit. The car is equipped with an onboard front camera and its motors accept forward speed V and angular speed W commands. Figure 2 shows the Gazebo scenario and the Formula1 car used. This exercise is designed to teach basic reactive control, including PID controllers as well as introducing students to basic image processing, for example color filters, morphological filters or segmentation of the red line from the track. The typical solution involves segmenting the red line and making the car follow it using a PID based control. The GUI of this exercise includes a bird's eye view widget to know the current position of the car inside the track.

The efficacy evaluator here gives a evaluation based on the time it takes for the car to go round the circuit. Then, the score is calculated according to this formula: $score = \min(10, 10 - (seconds - 60) * 1/60)$, where seconds represent the time it took the car to go round the circuit. Taking 60 seconds or less would give a score of 10. Then, the score is reduced one point for each extra minute. If a student did not manage to complete a lap, both the time and the final score will be a 0.

Second, in the BasicVacuumCleaner exercise the challenge is to program a robot, similar to iRobot Roomba, to clean an apartment in a given time. The more area covered in a stint of 5 minutes, the better its efficacy. It is intended to teach coverage algorithms. The covered percentage of cleanable surface is the score, which is shown to the student at the top right area of the exercise webpage, as can be seen in Figure 1.

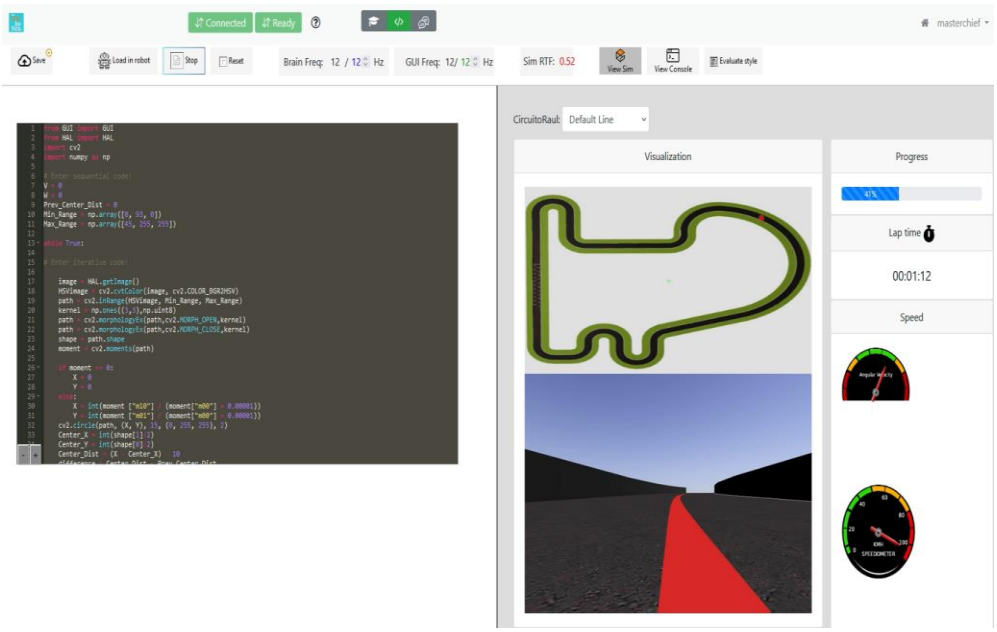

Figure 2. Example of the Follow Line exercise web page

5. Preliminary pilot study

In order to test the efficacy evaluator that Unibotics provides we carried out two pilot studies. 26 students from the Mobile Robotics course at the Robotics Software degree at URJC took part in the first one. 14 students enrolled in the Computer Vision Master at the URJC also participated. During their courses, they had to complete the Follow Line challenge. Both groups were able to use the efficacy evaluator while they were carrying out the task. The goal of this study was to check if the efficacy evaluator worked consistently and to analyse differences between both groups.

The final results of the efficacy evaluator for both pilot studies are shown in Table 1.We present both the time the student took to go round the circuit and the final efficacy score.

Thanks to the efficacy evaluator the students can have an idea about the performance of the code they have written. Moreover, they are able to improve their code to check if the score is higher after adding those improvements. The students know before starting to code, that the maximum score would be achieved if the car is able to do a lap in 60 seconds or less. Therefore, they had the basis to improve their codes before the submission deadline until their goal score was achieved. The final marks of the challenge were not calculated only with this score, but also other factors such as the style of the code.

Table 1. Students' efficacy score

	Degree students		Master students	
Id	**Time**	**Score**	**Time**	**Score**
1	240	7	0	0
2	154	8.43	223	7.28
3	60	10	215	7.42
4	120	9	79	9.68
5	0	0	90	9.5
6	50	10	173	8.12
7	45	10	105	9.25
8	90	9.5	162	8.3
9	166	8.23	110	9.17
10	126	8.9	157	8.38
11	120	9	156	8.4
12	122	8.97	142	8.63
13	54	10	138	8.7
14	86	9.57	133	8.78
15	0	0		
16	0	0		
17	110	9.17		
18	321	5.65		
19	360	5		
20	190	7.83		
21	71	9.82		
22	210	7.5		
23	92	9.47		
24	167	8.22		
25	0	0		
26	40	10		

6. Conclusions

This work presents the automatic code assessment system in the Unibotics robot programming platform. It includes a style evaluation based on PEP8 standard compliance and efficacy evaluation which heavily depends on the robot task. A preliminary study with 40 students has been performed URJC in which the students use an automatic efficacy evaluator to check the performance of their code when completing the FollowLine challenge. The students continuously received direct feedback about the performance of their code from the assessment system, and were able to further refine it before delivering the final version.

Regarding future lines, further work should study how they use the efficacy evaluator along the whole learning process, how many times students modify and evaluate their codes until they reach the final version. It would be interesting to study if the use of the efficacy evaluation tool affects the learning results of the students. Introducing spatial deviation from the red line as a new factor in the FollowLine score is under development.

References

Evripidou, S., Georgiou, K., Doitsidis, L., Amanatiadis, A.A., Zinonos, Z., Chatzichristofis, S.A. (2020). Educational robotics: Platforms, competitions and expected learning outcomes. IEEE access 8, 219534–219562

Huletski, A., Kartashov, D., Krinkin, K. (2015). Evaluation of the modern visual SLAM methods. In: Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT).pp.19–25. IEEE

Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In: Proceedings of the 10th Koli calling international conference on computing education research. pp. 86–93

Padilla, R., Netto, S.L., Da Silva, E.A. (2020). A survey on performance metrics for objectdetection algorithms. In: 2020 international conference on systems, signals and image processing (IWSSIP). pp. 237–242. IEEE

Roldán-Álvarez, D., Cañas, J. M., Valladares, D., Arias-Perez, P., & Mahna, S. (2023). Unibotics: open ROS-based online framework for practical learning of robotics in higher education. Multimedia Tools and Applications, 1-26.

Roque, L., Dantas, A., Camilo-Junior, C.G. (2019). Programming style analysis with recurrent neural network to automatic pull request approval. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE

Siegfried, R., Klinger, S., Gross, M., Sumner, R.W., Mondada, F., Magnenat, S. (2017). Improved mobile robot programming performance through real-time program assessment. In: Proceedings of the 2017 ACM conference on innovation and technology in computer science education. pp. 341–346

Striewe, M., Goedicke, M. (2014). A review of static analysis approaches for programming exercises. In: Computer Assisted Assessment. Research into E-Assessment: International Conference, Zeist, The Netherlands. Proceedings. pp.100–113. Springer

Xia, F., Shen, W.B., Li, C., Kasimbeg, P., Tchapmi, M.E., Toshev, A., Martín, R., Savarese, S. (2020). Interactive Gibson benchmark: A benchmark for interactive navigation in cluttered environments. IEEE Robotics and Automation Letters 5(2), 713–720.

Yang, C., Liu, Y., Yu, J. (2018). Exploring violations of programming styles: Insights from open source projects. In: Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence. pp. 185–189.

Yumaganov, A., Agafonov, A. (2021). Comparison of autonomous driving approaches. In: 2021 International Conference on Information Technology and Nanotechnology (ITNT). pp. 1–4. IEEE.