# Automatic Image Marking Process

Aeman I.G. Masbah
School of Computing and Engineering
University of Huddersfield
Huddersfield, UK
E-mail: Aeman.Masbah@hud.ac.uk

Joan Lu
School of Computing and Engineering
University of Huddersfield
Huddersfield, UK
E-mail: J.lu@hud.ac.uk

***Abstract*-Efficient evaluation of student programs and timely processing of feedback is a critical challenge for faculty. Despite persistent efforts and significant advances in this field, there is still room for improvement. Therefore, the present study aims to analyse the system of automatic assessment and marking of computer science programming students' assignments in order to save teachers or lecturers time and effort. This is because the answers are marked automatically and the results returned within a very short period of time. The study develops a statistical framework to relate image keywords to image characteristics based on optical character recognition (OCR) and then provides analysis by comparing the students' submitted answers with the optimal results. This method is based on Latent Semantic Analysis (LSA), and the experimental results achieve high efficiency and more accuracy by using such a simple yet effective technique in automatic marking.**

***Keywords-Automatic Image Marking Process; Optical Character Recognition; Test and Evaluation; Operational Test and Evaluation.***

## I. INTRODUCTION

Computer-based Assessment Systems (CAS) has grown exponentially in the last few years because of the growing number of university students and increasing contributions of e-learning approaches to asynchronous and ubiquitous education.

The marking of student assignments can be classified as manual and automatic. Unfortunately, instructors and teaching assistants are already overburdened with work teaching computer science courses; they have little time to devote to additional assessment activities. As a result, an automated tool for grading student assignments must be devised.

Many educators have used automated systems to assess and provide quick feedback on large volumes of student programming assignments [1][2]. Such systems typically focus on the compilation and execution of student programs against some form of instructor-provided test data. However, this approach ignores any testing that the student has undertaken, and fails to provide both the assessment and feedback necessary for facilitating Test-Driven Development (TDD) [3].

Marking the programming assignments of many students is not often an easy job for instructors. Thus, making the marking easier benefits the automated marking program. Such innovation in marking programming class assignments electronically is, arguably, as important as the learning curriculum for programming classes [4]. Automated marking applications are more accurate in detecting errors and providing feedback.

This paper presents a novel automatic image marking process technique. Section 2 discusses the related work, and section 3 describes the design of the proposed technique. Section 4 presents the experimental results, and section 5 concludes the study.

## II. RELATED WORK

The problem of marking automation has attracted much research attention. Early studies on computerisation considered the practicality of the general approach to different programming dialects by exploring diverse evaluation systems. The early Ceilidh framework checked understudy assignments in dialects that included Standard ML [5] in a similar way as that presented in this work, but without high-level, consistent joining using a cutting-edge Learning Management System (LMS) [6].

Recent studies have focused on the specifics of Java assessment and interactive learning. Truong et al. [7] attempted to assess semi-automatically Java programs via static analysis without compiling and executing programs. Tremblay et al. [8] assessed Java programs using a command-line tool available to students who use a Unix-based system and noted the possibility of a future Web-based application. Blumenstein et al. [9] developed a generic GAME system that can be used as a framework for the automated grading of assignments in programming languages, including Java.

Web-CAT is a web-based application that is implemented using the WebObjects framework of Apple [10]. This application is designed to be language independent, but focuses on grading object-oriented programs that are written in Java. For Java programs, students write JUnit-compatible test cases and submit them along with the assignments in their other classes. The reports produced by these tools are merged into one seamless source code mark-up, which can be viewed on the Web by students.

Redish et al. [11] developed a tool called AUTOMARK to evaluate student style based Pascal programs. Berry et al. [12] [13] developed another tool to assess student programs written in C language depending on

 ISBN: 978-1-61208-478-7 

style. Jones used the concept of testing to automate the evaluation of student programs [14].

Furthermore, Jackson and Usher developed a tool called ASSYST to automate student programs depending on their correctness, efficiency, complexity and style [1]. Jumaa developed a tool to evaluate structural languages such as Pascal, FORTRAN, C and Basic based on Halstead, McCabe, Style and Lipow and Thayler models [15]. The instructor evaluated student programs against it. The tool proved to be suitable for intermediate courses. As for advanced courses with big projects, it was impractical for the instructor to write a model program for each assignment. Also, in the industry it is difficult to write a model program in order to assess an industrial program.

One of the very early applications in the course of automated program marking, Hollingworth's grader, was specifically developed to test punched card programs [16]. Many other applications with similar functions have been introduced, for instance, the Online Judge [17], and, more recently, Sakai, which was developed with much more sophisticated ware introduced by Suleman [18]. Although the varieties of Automated Marker available can differ in name or other peripheries, their principal functionality is to evaluate programs written by students indirectly through the output. This process of indirect test is not without some deficiency. According to experts, the deficiency of the indirect approach of testing programs includes, but is not limited to, "limited quality of feedback", "heavy hindrances on evaluation" and "over sensitivity to minor errors". Another noticeable pitfall of the available series of automatic program markers is the inability to mark non-textual programs, interactive programs and tasks with specific algorithms; some examples are animation or drawing programs that students are sometimes required to write. Pragmatically, it is important to explore ways of upgrading the functionalities of the already existing automated markers and suggest solutions to the currently noticeable pitfalls. Other common approaches, amongst the available automated program makers in the literature, are those that apply the file-system-based organisational strategy. For instance, the Isong [19] automated program marker was developed to focus on compiling student programs automatically. This is done by comparing the instructor-provided data against the student program output. Isong's marker was written with the help of Unix-shell scripts. Reek developed a similar grader long before Isong's marker [2]. Like Isong's marker Reek's grader is also a Unix-based system that was developed for inductor programing courses. This grader also adopted the file-system organisational strategy for evaluating assignments and student submission. The submissions are graded against instructor-provided data. Hence, instructors control the grader's feedback and evaluation process.

BOSS is also an automated program marker developed with a battery of Unix-based programs, which adopts file-system-based organisational strategy for submission of test cases tested against instructor-provided test criteria [20]. Through these studies, the proposed system will differ from other systems by depending on accuracy and efficiency in the operations of automatic marking. It will use a new technology based on Images and OCR.

## III. RESEARCH APPROACH

A systematic investigative process was employed to increase or revise current knowledge of automatic marking. This section discusses the research approach, which consists of two sub-phases: (A) designing an improved automatic image marking process system and (B) testing and evaluating the developed system.

### *A. Automatic Image Marking Process System Design*

The entry point of the proposed system will be the submission of the programming assignments in image format. Optional Character Recognition (OCR) will then be used to extract the text from the submitted assignments and to save the text file. The proposed system is a combination of OCR, web technology and database. Web technology is used to develop a Web interface that enables students to submit their assignments, and teachers to mark the submitted answers and manage students' marks. Furthermore, this process will be explained with more detail in phases: (1) submission process and (2) marking process. The database is used to save the students' marks, and the saved data are used later by the teachers to generate their reports. This section consists of two sub-sections: (1) submission process and (2) marking process. Figure 1 shows the architecture of the proposed system.

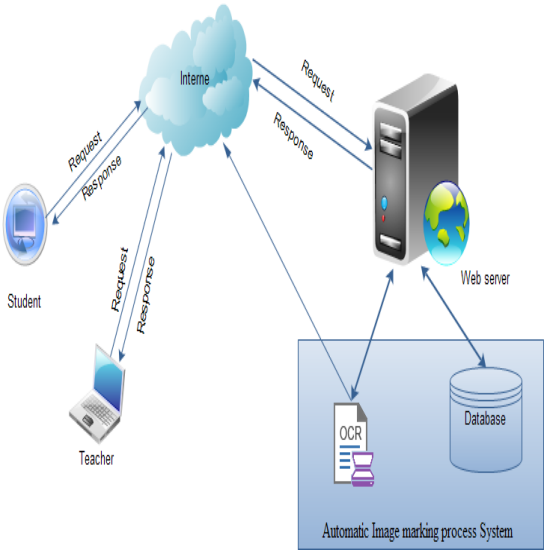


Figure 1. Architecture of the proposed system.

### 1. Submission Process

The computer science programming assignments undergo the stages of (1) compilation, (2) execution and (3) testing. The submission process, which begins after the execution stage, is described as follows:

- The student executes his/her programming assignment using programming IDE.
- The student converts the result of the execution into an image (e.g., a snapshot of the result).
- The student logs into the system using his/her metric number. In order to enforce proper security, each user must first register onto the system before he/she can use any of the other functionalities. Registration ensures that a proper ID and password are created for each new user.
- The student uploads the image that contains his/her answer.
- The system creates a folder named after the metric number of the student and then saves the uploaded image inside the folder.

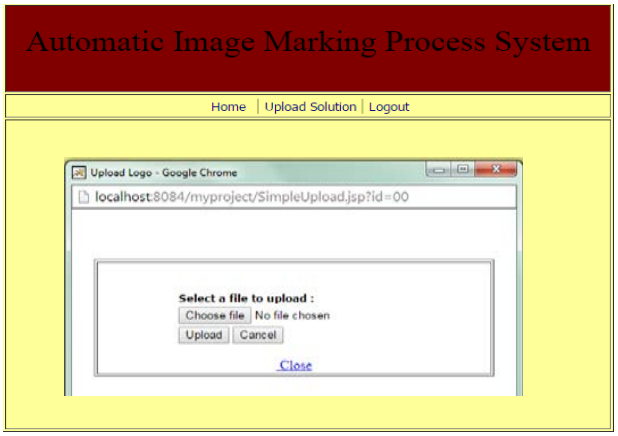


Figure 2. Web interface for uploading assignment answers.

The proposed system provides a web interface for students to upload their assignment answers. Figure 2 shows the web interface.

2. Marking Process

The proposed system provides an automatic marking process for the submitted answers. The marking process is described as follows:

- The teacher logs onto the system using his/her teacher ID.
- The teacher selects one of the submitted answers.

The system allows the teacher to upload the optimal Answer and to enter the assignment mark. Figure3 shows the upload of the optimal answer.

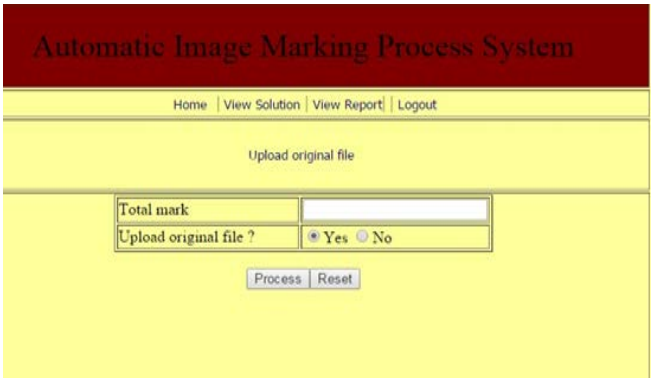


Figure 3. The upload of the optimal answer and the assignment mark box.

- The system uses the OCR web service to extract the text from the answer of the student and the optimal answer of the teacher.
- The system compares both texts (i.e., the submitted and optimal answer) and computes the similarity percentage.

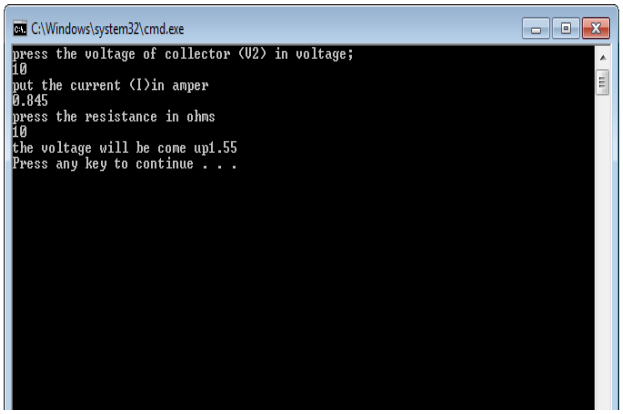


Figure 4. Submitted image that represents the student's answer.

The accuracy of the text extraction is positively affected by a high image quality. Figure 4 shows the submitted image for the student's answer.

 ISBN: 978-1-61208-478-7 

Figure 5 shows the text extracted from the image using the OCR web service.

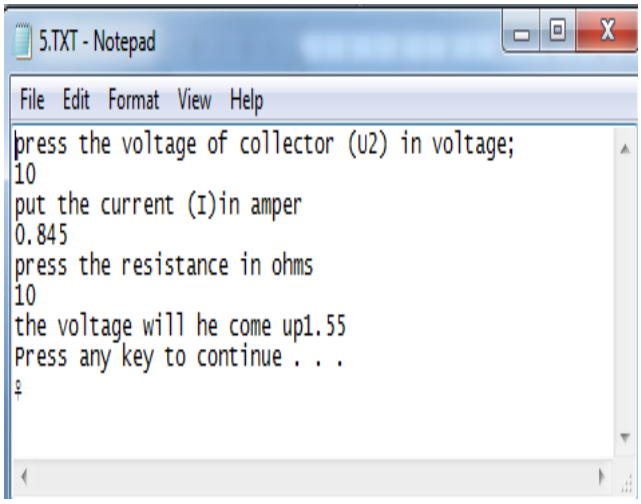


Figure 5. Text extracted from the image using the OCR web service.

### B. Testing and Evaluation

Test and Evaluation (T&E) is the process by which a system or its components are compared against the requirements and specifications through testing. The results are evaluated to assess the progress of design, performance, supportability and more. Developmental Testing and Evaluation (DT&E) is an engineering tool used to reduce risk throughout the acquisition cycle. Operational Test and Evaluation (OT&E) is the actual or simulated employment, by typical users, of a system under realistic operational conditions [21]. In this phase, the proposed system is tested against the student answer samples to check the matching percentage of the proposed system.

## IV. EXPERIMENTAL RESULT

The proposed system is tested on a sample of 65 student answers. The targeted samples are divided into five groups as follows: (1) Group A, (2) Group B, (3) Group C, (4) Group D and (5) Group E.

Group A consists of students' answers with zero matching optimal answers, while Groups B ,C and D consist of students' answers with partial matching of optimal answers. Group E consists of students' answers with identical matching of optimal answers.

The students upload different answers to computer science programming questions. The teacher selects a specific answer, uploads the optimal answer, and gives the total mark for a particular question. The system calculates the matching percentage using (1). Figure 6 shows the matching percentage for each group.

Percentage of matching =

$$\frac{Number\ of\ matched\ lines}{Total\ number\ of\ the\ optipal\ answer\ line} \ldots \text{“(1)”}$$

Mathematically, this can be explained as the simultaneous representation of all optimal answers uploaded in the assignments corpus as points in semantic space, with the initial dimensionality of the sequences of answers in the developing system. To classify the correct representation of optimal answers, we represent it as a vector, and determine which answer is nearest to the optimal answer, where the distance measure between two vectors $x_n$ and $y_n$ is defined as:

$$d\left((x_n - y_n)^n = log \sum_{k=0}^{n} (x_n - y_n) x^k\right)$$

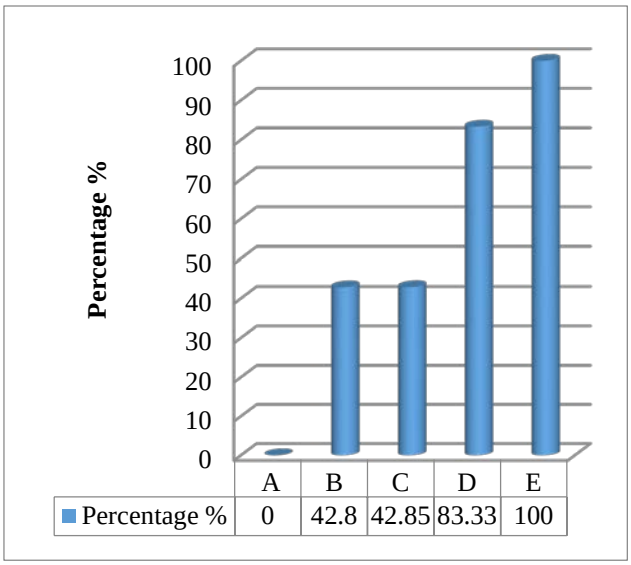


Figure 6. The matching percentage for each group.

The entire sample is submitted and evaluated. A zero matching percentage is obtained if these answers do not match each other as shown in Group A. B 42.8%, C 42.85% and D 83.33% decreased matching percentage is obtained if these answers partially match each other as shown in Group B, C and D. E 100% matching percentage is achieved in group E when the submitted and optimal answers exactly match each other.

The matching accuracy depends on the adoption of OCR and the advanced analysis that is applied to the submitted answers. However, such accuracy is negatively affected by the quality of the uploaded image that represents the student answer. Image quality is one of the most important factors for improving the quality of recognition. A resolution of 200 DPI to 400 DPI is recommended for a better recognition. An example of a system output is shown in Figure 7.

 ISBN: 978-1-61208-478-7 

The content of the submitted image is extracted, and each line of the extracted content compared with the corresponding line in the optimal answer to check whether they match each other.

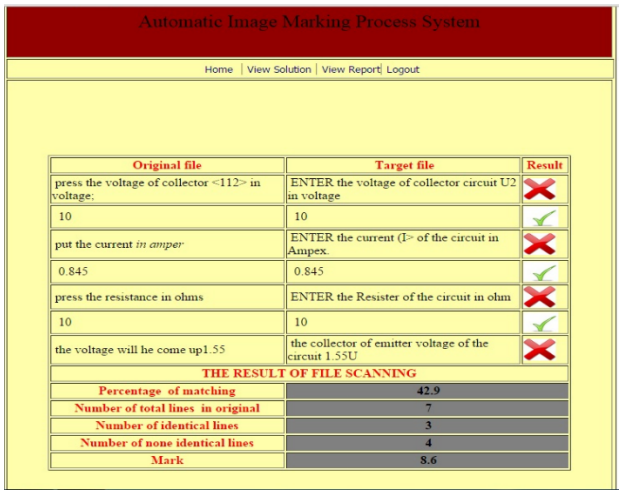


Figure 7. Example of output.

## V. Conclusion And Future Work

An approach to automatically evaluating computer students' assignments has been described. The framework is based on Optical Character Recognition (OCR). Automatic Assignment Scoring (AAS) aims to extend the system's capabilities to provide more efficient and accurate results, as well as to save teachers or lecturers time and effort. This paper has illustrated the students' group matching with optimal answers. The 100% matching percentage is achieved in group A. The experimental results validate the efficiency of the proposed system in the automatic marking process.

In future, the proposed system will be integrated with the interface of Huddersfield University website. Furthermore, the prototype will be evaluated through task-based trials and comparative qualitative evaluation and testing with other systems.

## References

[1] D. Jackson and M. Usher, "Grading student programs using ASSYST," in ACM SIGCSE Bulletin, 1997, pp. 335-339.

[2] K. A. Reek, "A software infrastructure to support introductory computer science courses," in ACM SIGCSE Bulletin, 1996, pp. 125-129.

[3] K. Beck, Test-driven development: by example: Addison-Wesley Professional, 2003.

[4] J. Al-Jáafer and K. E. Sabri, "Automark++: A case tool to automatically mark student Java programs," Int. Arab J. Inf. Technol., vol. 2, pp. 87-96, 2005.

[5] S. P. Foubister, G. Michaelson, and N. Tomes, "Automatic assessment of elementary Standard ML programs using Ceilidh," Journal of Computer Assisted Learning, vol. 13, pp. 99-108, 1997.

[6] Q. Wang, H. L. Woo, C. L. Quek, Y. Yang, and M. Liu, "Using the Facebook group as a learning management system: An exploratory study," British Journal of Educational Technology, vol. 43, pp. 428-438, 2012.

[7] R. Saikkonen, L. Malmi, and A. Korhonen, "Fully automatic assessment of programming exercises," in ACM Sigcse Bulletin, 2001, pp. 133-136.

[8] G. Tremblay, F. Guérin, A. Pons, and A. Salah, "Oto, a generic and extensible tool for marking programming assignments," Software: Practice and Experience, vol. 38, pp. 307-333, 2008.

[9] M. Blumenstein, S. Green, A. Nguyen, and V. Muthukkumarasamy, "Game: A generic automated marking environment for programming assessment," in Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference, 2004, pp. 212-216.

[10] S. H. Edwards and M. A. Perez-Quinones, "Web-CAT: automatically grading programming assignments," in ACM SIGCSE Bulletin, 2008, pp. 328-328.

[11] K. Redish and W. Smyth, "Program style analysis: A natural by-product of program compilation," Communications of the ACM, vol. 29, pp. 126-133, 1986.

[12] R. E. Berry and B. A. Meekings, "A style analysis of C programs," Communications of the ACM, vol. 28, pp. 80-88, 1985.

[13] W. Harrison and C. R. Cook, "A note on the Berry-Meekings style metric," Communications of the ACM, vol. 29, pp. 123-125, 1986.

[14] E. L. Jones, "Grading student programs-a software testing approach," Journal of Computing Sciences in Colleges, vol. 16, pp. 185-192, 2001.

[15] Jumaa, "A Computer Model for Evaluation of Programs," University of Engineering and Science, 1992.

[16] J. Hollingsworth, "Automatic graders for programming classes," Communications of the ACM, vol. 3, pp. 528-529, 1960.

[17] B. Cheang, A. Kurnia, A. Lim, and W.-C. Oon, "On automated grading of programming assignments in an academic institution," Computers & Education, vol. 41, pp. 121-131, 2003.

[18] H. Suleman, "Automatic marking with Sakai," in Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology, 2008, pp. 229-236.

[19] J. Isong, "Developing an automated program checker," Journal of Computing Sciences in Colleges, 2001, pp. 218-224.

[20] M. Luck and M. Joy, "A secure on-line submission system," Software-Practice and Experience, vol. 29, pp. 721-40, 1999.

[21] K. T. Weber and J. S. Janicki, "Cardiopulmonary exercise testing for evaluation of chronic cardiac failure," The American Journal of Cardiology, vol. 55, pp. A22-A31, 1985.

 ISBN: 978-1-61208-478-7 