# Method for Quick Identification of Computer Operations Performed by a Student

Kenta Morita
Graduate School of Information and Telecommunication Engineering
Tokai University
Tokyo, Japan
e-mail: morita.k@star.tokai-u.jp

Haruhiko Takase
Graduate School of Engineering
Mie University
Mie, Japan
e-mail: takase@elec.mie-u.ac.jp

Naoki Morita
Graduate School of Information and Telecommunication Engineering
Tokai University
Tokyo, Japan
e-mail: morita@tokai.ac.jp

***Abstract* – In classes where personal computers (PCs) are actively used as a part of lesson plans, it is important for teachers to be able to quickly locate the operations that students have performed on their individual machines during exercises. To identify such operations, a method using a global hook has been utilized previously. However, when this method is used, numerous successive screen-captured images must be examined and it is difficult to find the changed portions that allow the operations to be identified. Herein, we propose a method that presents changed portions in an easily recognizable manner and reduces the number of captured images to be examined.**

***Keywords- Compute-assisted instruction; Operation Process.***

## I. INTRODUCTION

Recently, classes in which students use personal computers (PCs) or tablet devices have become more commonplace [1]. In most such classes, students learn how to use applications, such as Microsoft Word or Excel, or practice programming techniques, by operating their PCs in accordance with instructions from a teacher.

When teachers present processes that students can manipulate on their PCs during class, lessons can proceed more smoothly and students can better understand the class contents. However, when a student is unable to operate his device in accordance with the teacher's instructions, the instructor must identify which parts of the student's operations are incorrect and correct the student's misunderstandings.

Unfortunately, when a student has made a mistake, the teacher may not be able to identify the errors in the student's process simply by viewing the student's screen. For example, as shown in Fig. 1, in which an exercise for calculating the sum of list of values using Excel is displayed, there is no visually identifiable difference between using the SUM function and directly inputting a value. More specifically, either 21 has been manually input into cell B6 or the "=SUM(B2:B5)" operation has been completed, but it is impossible to know which method has been used simply by looking at the screen.

Since it is difficult to identify where mistakes have been made in cases where screens look identical, determining the process the student used on his/her PC can help the teacher identify the incorrect operation. Additionally, in programming classes, it is important for teachers to know the processes that students use when editing source code because such processes contain information that the teacher needs to assess whether students followed instructions correctly. This allows the teacher to determine whether students have adequately grasped their lesson content.

Until now, while it has been possible for a teacher to observe all student operations if the student's PC screen is continuously captured in the form of video [2][3], this process includes all the screen time during which the student is not operating the PC, most of which is superfluous to the teacher's needs. Therefore, a method [4][5][6] that allows only (operational) changes made by the student to be collected is desirable.

Accordingly, in this paper, we propose a method by which all computer operations and the corresponding changes made by a student can be recorded and rapidly presented to the teacher, and by which unnecessary screen captures are eliminated. The use of this method can be

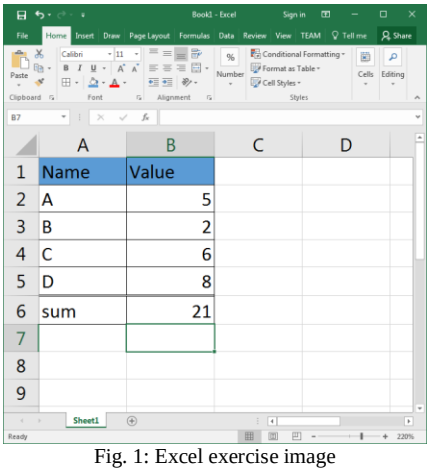

Fig. 1: Excel exercise image

 ISBN: 978-1-61208-541-8 

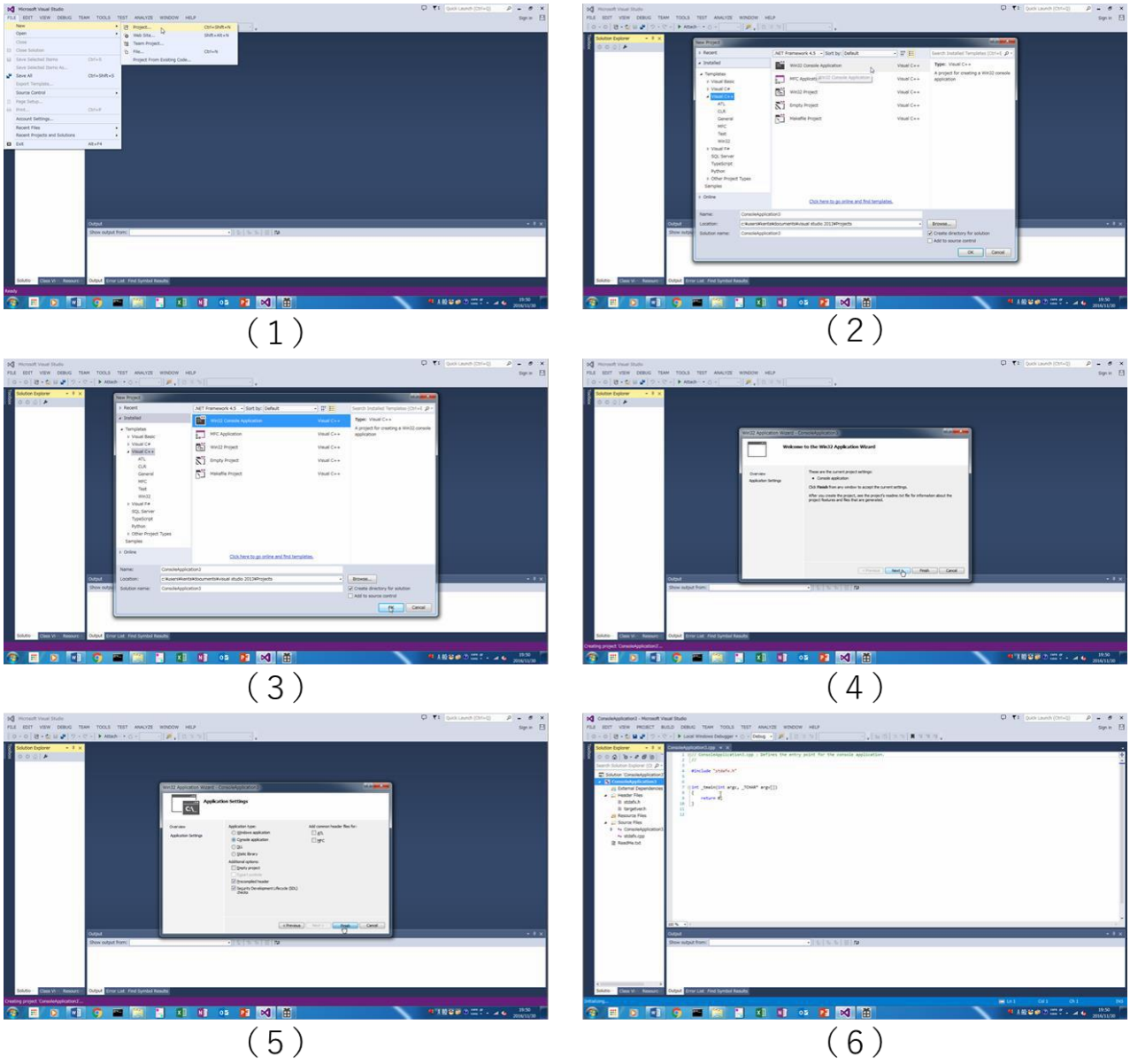


Fig. 2 Confirmation using global hook method.
(1)Select [New Project], (2) select [Win32 Console Application], (3) select [OK], (4) select [next], (5) select [Finish] (6) mouse click.

expected to help the teacher quickly ascertain the student's lesson comprehension.

This paper describes the previous methods and problems in Section II. In Section III, the method for quick identification is proposed. We discuss the effectiveness of our proposed method in Section IV. Section V presents the conclusion and future work.

## II. PREVIOUS METHODS AND PROBLEMS

Currently, there are two methods that use dedicated applications [4] [5], and one global hook method [6] that can be used to restrict the information provided to the teacher to just changes in the student's process.

However, the dedicated applications only target programming operations and acquire the process taken in creating source code by using a dedicated text editor. Thus, while this method is capable of presenting information regarding the process used, it can only acquire changes that were entered via the dedicated text editor. This means it will not acquire operations, such as file openings on the desktop, nor can it acquire and recode the computer operations performed by students. As a result, this method cannot present all operations performed by the student.

There is also a global hook method that can be used to collect all of the operations carried out on a PC. This method monitors messages the operating system (OS) sends to an application, takes screen captures when it detects keyboard

 ISBN: 978-1-61208-541-8 

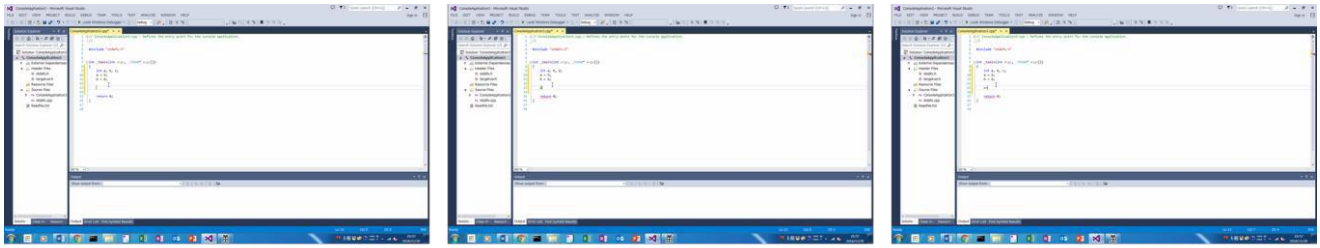


Fig. 3 Editing programming source code.
(1) Insert [Enter], (2) insert [c], (3) insert [=].

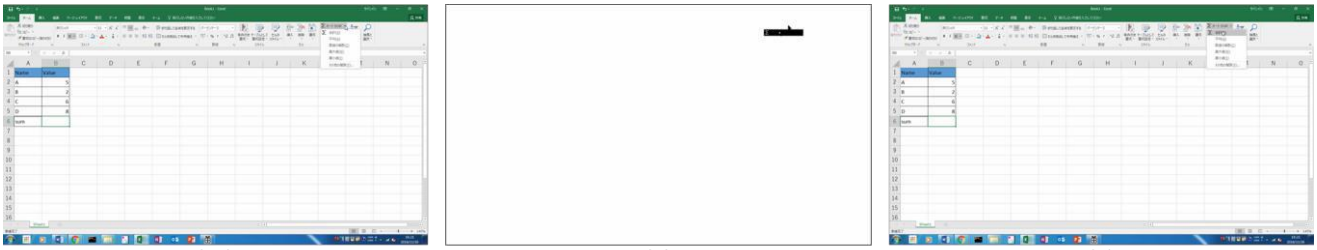


Fig. 4 Confirmation using proposed method.
(1) Previous image, (2) differential image, (3) next image.

input or mouse clicks, and saves the screen captures as image data. The process can then be determined by reviewing the captured images one by one, in order, as shown in Fig. 2, and noting the changes. By looking at the order in which they were captured, faulty operations can be determined.

However, while the global hook method acquires all of the operations that the student has performed and can present the information to the teacher, the problem with this method is that it is difficult to locate the path taken because a screen capture is made whenever student operates the keyboard or mouse, and each captured image must be examined one by one. Moreover, editing programming source code results in less visible changed portions of the screen because the input characters appear very small, as shown in Fig. 3, thereby making it necessary to scrutinize each image carefully. This makes reducing the number of screen capture images desirable.

Our goal was therefore to develop a technique that teachers can use to quickly identify operations performed by a student, while also reducing the number of images captured via the global hook method.

## III. PROPOSED METHOD

In this section, we propose techniques that will permit teachers to identify computer operations performed by a student by identifying changes in successive screen captures while reducing the number of captured images required.

### *A. Finding changed portions*

The basic idea behind our method of finding changed portions is to present only the portions that have been changed. For example, if there is an operation B that takes place after operation A, existing methods presents an image of B after the image of A is presented. In contrast, our proposed method presents the image of A next to an image that shows only the changed portions, which is the difference between images A and B. By examining the differences between previous and successive images, a teacher examining a student's process can predict where changes will occur in the next image, and more easily understand the operations performed, which is where his/her attention should be focused.

The image (difference image) used to display the changed regions is made by comparing two successive screen captures and consists of black and white pixels. Black pixels correspond to pixels that have changed between the two images, and white pixels correspond to pixels that have not changed.

To clarify the changed portion, the system shows the difference image between two successive images. For example, Figure 4 shows transition images of confirmation using the proposed method. Here, it can be seen that even if a teacher examines captured images one by one, he/she can easily understand the student's process simply by noting the black pixels in each successive image.

### *B. Reducing captured images*

The basic idea for reducing the number of images to be captured is to refrain from creating screen captures if the changes in the screen are insignificant.

Existing methods generate a screen capture whenever the student operates the mouse or keyboard, which means that numerous captured images have very small changed portions, or no changes at all. However, it is still necessary for the teacher to examine each image to be sure no important changes are present. In contrast, our proposed method

 ISBN: 978-1-61208-541-8 

compares each new captured image with the one before it, calculates the number of changed pixels, and then creates a computer screen capture when the number of changed pixels exceeds a certain preset threshold value.

Using our proposed method, the number of screen captures is significantly reduced compared to existing methods. This method captures the screen if the number of black pixels exceeds the threshold, thus reducing the number of captures. For typing programs, the system captures occasionally, since black pixels appear only for characters that are being typed. For application menus or windows, the system captures frequently, since black pixels will appear over a wider area. Therefore, since only major operations are captured, it is possible to reduce the number of screen captures.

## IV. EXPERIMENTS

In this section, we discuss the effectiveness of our proposed method based on the results of a simple experiment.

### A. Experimental setup

In this experiment, in which the proposed method is compared to an existing method, we found it is easy to identify the changed portions, the number of captured images is low, and it is easy to determine the operations carried out by the student.

We selected a class of undergraduate students studying Java programming to test the method. This class involved a three-hour long practice session that mixed small operations with operations producing large changes to the screen. Students were required to manipulate text-editor-typed source code as well as run an application that converts the source code into an executable file. Three test subjects confirmed the operations.

Regarding the total number of changed pixels used as the threshold to reduce the number of captured images, 1/20 of the screen size of the PC seemed to be appropriate.

### B. Results

We will begin by discussing whether it was easy to determine the portions of the screen changed by student operations using our method. One of the authors asked all test subjects whether it was possible to identify operations performed by the students when looking at their screen capture images one by one. The subjects stated that it was easy to find the changed portions because all they had to do was find the parts displayed as black pixels.

Next, we asked whether the number of captured images was low compared to existing methods. An existing method captured 1860 images, while the proposed method captured 584 images. Since the programming of the proposed method limited captures to large screen changes rather than capturing all changes, including relatively small screen changes caused by programming, there was a significant reduction in the number of images captured. Since the teacher wants to know all the operations of the student, we think that all these images are useful.

Finally, regarding whether it was easy to locate changes, we asked all test subjects to interact with the tool and to identify any operations performed by the students. Three test subjects responded with the name of the operation the student had actually performed, such as “press this button”, or “type A”. Teachers can find mistakes such as, student is not operating PC as instructed.

## V. CONCLUSIONS

We discussed methods that teachers can use to determine the computer operations being performed by his/her students. We began by discussing an existing method that is difficult for teachers to use because it requires them to examine numerous sequential screen capture images in order to identify changed portions in the images.

We then introduced our proposed method in which new images are compared to the image captured immediately before it, and the portions that exhibit significant differences are output using different pixel colors. In our method, no screen capture is performed if the number of changed pixels is less than a preset threshold.

After examining the results of our experiment, we confirmed that our proposed method makes it easier for teachers to identify the changed portions and is capable of reducing the number of required captured images.

The proposed approach always captures the screen when the user switches from applications repeatedly. Such an image may not be useful for the teacher, so it is necessary to reduce it.

In the future, we want to confirm the effectiveness in more classes Since the number of subjects and students in the experiment of this paper is small.

## ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 16K16324.

## REFERENCES

[1] G. Thavamalar, "Successful implementation of e-learning: Pedagogical considerations" The internet and higher education, Vol.4 No.3, pp.287-299.

[2] R. Stannard, “Screen capture software for feedback in language education” Second International Wireless Ready Symposium, MAR. 2008, pp. 16-20, ISSN 1995-4557.

[3] J. Nigel, P. Georghiades, and J. Gunson, "Student feedback via screen capture digital video: Stimulating student’s modified action." Higher Education Vol.64 No.5, pp.593-607.

[4] Y. Katagiri, Y. Tateiwa, D. Yamamoto and N. Takahashi, “Implementation of a Context-Aware Programming Instructor Assistant Using an Analysis Function of Programming Activity” IEICE Technical ReportET2009-83, pp.181-186.

[5] H. Igaki, S. Saito, A. Inoue, R. Nakamura and S. Kusumoto, “Programming Process Visualization for Supporting Students in Programming Exercise” IPSJ Jornal, Vol.54 No.1, pp. 330-339.

[6] N. Morita, “A Programming Process Visualization System With Global Hooking” Society for Information Technology and Teacher Education(SITE2014), MAR. 2014, pp. 1945-1953, ISBN 978-1-939797-07-0.

 ISBN: 978-1-61208-541-8 