Indonesian Journal of Electrical Engineering and Computer Science
Vol. 23, No. 3, September 2021, pp. 1350~1356
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v23.i3.pp1350-1356

Advanced UI test automation (AUTA) for BIOS validation using OpenCV and OCR

Eissa Abdullah Awadh Mohammed[1], Muslim Mustapa[2], Hasliza Rahim[3], Mohd Natashah Norizan[4]
[1,2,3,4]Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis (UniMAP), 02600 Arau, Perlis, Malaysia
[2]Advanced Computing, Centre of Excellence (AdvComp), Universiti Malaysia Perlis (UniMAP)
[3]Advanced Communication Engineering, Centre of Excellence (ACE), Universiti Malaysia Perlis (UniMAP)
[4]Centre of Excellence Geopolymer and Green Technology (CeGeoGTech), Universiti Malaysia Perlis (UniMAP)

Article Info

Article history:

Received Feb 18, 2021
Revised Apr 14, 2021
Accepted Apr 20, 2021

Keywords:

BIOS validation
Computer vision
OCR
Test automation
UI automation

ABSTRACT

Basic input output system (BIOS) validation is performed on both graphical user interface (GUI) and command-line interface (CLI) by a test engineer. Keyboard and mouse are used to insert test cases commands into system under test (SUT). Test engineer monitors test cases progress on a monitor for validation. This method is time-consuming and relatively more expensive than automation. In this project we designed an independent automation system that able to mimic human interaction in BIOS validation. The approach can be divided into two main parts. The first part is the input device to enter commands into SUT and the second part is the advanced image recognizer. The keyboard and mouse emulator is used as an input device to reproduce test commands and send them to an SUT. The image analyzer algorithm is developed using OpenCV and optical character recognizer (OCR) tools to help automate some test challenges. Our result shows that advanced user interface (UI) test automation (AUTA) can perform a 125 test cases within 5 hours compared to 48 hours for a human to complete the job.

Corresponding Author:

Muslim Mustapa
Faculty of Electronic Engineering Technology
Universiti Malaysia Perlis, Kampus Alam Pauh Putra, 02600 Arau, Malaysia
Email: muslim@unimap.edu.my

1. INTRODUCTION

The continuous growth in electronic products has demanded advanced applications to support the rigorous engineering design and development phase. An engineering product then need to be validated or tested before it been released into the market. Test engineer [1] needs to enhance current validation system to catch up with the rapid changes in technology [2]. Tedious and labor-intensive manual tests in engineering validation team are gradually missing the deadline as the electronic products are becoming more complicated and involving an agile development process. There were efforts done by some technology company to develop a test automation that focuses on increasing test efficiency, coverage, and effectiveness [3]. An automated test can be run repeatedly with comparably lower costs with faster speed [3], [4] and fit well with the requirement of an agile software and hardware development process. The recent technological advances in system on chip (SoC) [5], internet of things (IoT) [6], [7], computer vision (CV) [8], cloud computing [9], real-time technology [10], [11] and networked control systems (NCSs) [12] have made the full automation in validation and test industry become possible.

Most of the test conducted in semiconductor industry involve hardware and software integration and is often performed by a human tester using a keyboard and mouse. Any test involving user interface (UI) need a keyboard and mouse to drive a test run [13]. Test engineer need to control the keyboard and mouse along with monitoring the system behaviors to validate the test run [9]. Since the steps in the test run are

often repetitive and monotonous, testers often try to find a suitable testing tool to automate the task. The basic requirement of a software testing automation is the ability to reproduce a keyboard and mouse events to drive a test on an system under test (SUT) with accuracy at higher speed.

There are two types of user interface (UI) that are mainly use in industrial applications which are graphical user interface (GUI) and command-line interface (CLI). Examples of GUI are desktop platforms [14], [15], website [16], [17], and smartphone applications [18], [19]. The examples for CLI are Basic input output system (BIOS) interface, Linux terminal [20], and extensible firmware interface (EFI) shell. BIOS validation engineer usually interacts manually with UI to ensure that the BIOS firmware completed its job without any error (e.g. check for successful booting to several operating systems or check how it handles the power management). This project focused more on the UI involving BIOS validation, such as Windows, Linux, EFI-shell, and BIOS interface.

2. THE PROPOSED METHOD

There are not many works done on UI based BIOS validation or BIOS testing procedures automation can be found in any scholarly article [13], [21]. On the other hand there are many works done on UI automation in general where some of them are applicable in BIOS validation [22]. UI automation approach can be divided into two aspects that are SUT-dependent and Host-dependent.

SUT-dependent is a tool or method that requires SUT to be able to execute the automated scripts. Almost all automated UI testing tools are designed based on SUT-depend, such as:

- Sikuli [15] is a famous GUI automation tool that uses image recognition to search for GUI components on the display and react to it. Sikuli lets a tester select a region of interest on the screen, submit the image in the region as a query to the search engine, and browse the search results. Sikuli supports different operating systems such as Windows, Linux, and Macintosh.
- The proposed system in [23] addresses the desktop version of programming by demonstration problem. It stands at the crossways of intention-inference, software usability, and action identifying and predicting. It allows a tester to teach a bot to do a repetitive task. The process is similar to teaching a human for the first time. The tool predicts and recovers a user's intended loops by analyzing the data from their initial demonstration.
- The work in [24] proposed a method using a machine learning method that automatically identifies GUI widgets in screenshots to improve GUI testing. The study found that recognizing GUI widgets in screenshots and using this information to guide random testing achieved significantly higher branch coverage when compared to traditional random testing.

Challenge: There are a few challenges in applying the SUT-dependent method for BIOS validation automation. This method cannot run automated scripts without OS, which means SUT cannot be controlled during the booting process or in BIOS interfaces. Another challenge is SUT-dependent method does not support a CLI such as EFI-shell and Terminals, which are essential in BIOS validation.

Host-dependent is a method that runs automated scripts on an independent platform called host machine and usually has a hardware connection between the host and SUT. This method is having full control of an SUT and can run an automated scripts. It does not rely on SUT OS. Few past studies have been done on the host-dependent method such as:

- The work in [25] proposed an automated GUI testing method based on image detection which uses the image detection technology to recognize components in the GUI image. The input device simulates input signals into SUT. Using this method, they developed a GUI testing platform called AGTP, a universal and non-intrusive GUI testing tool for SUT. When the test script runs, it will call the image recognition and input device simulation to complete the corresponding operations according to the instructions. After executing the test script, test results information will be recorded and stored in the database for subsequent playback and regression testing.
- A capture/replay testing tool called KORAT [21]. KORAT adopts a hardware component to intercept and emulate keyboard/mouse signals to drive a SUT as if the SUT is interacting with a human. A tester can design and operate a test case to record the intended behaviors into KORAT test case script on a correct SUT without programming skills. In a regression run, the test case is replayed, and the correctness is asserted automatically by analyzing SUT's video output (images) and sending keyboard and mouse signals smartly. Since KORAT only interfaces the video output of a SUT, this platform is considered independent and non-intrusive, which means there is no performance interference caused by KORAT's capture and replay.
- KORAT is introduced again in [13] with major improvements. ARM-cortex M4 development board is used as a USB emulator and at the same time the work combined both SUT-dependent and Host-dependent approaches for mouse automation. The improved KORAT has improved the detection rate of optical character recognizer (OCR) from nearly 25% to nearly 85%.

Challenge: The host-dependent method is suitable to automate BIOS validation because it can control an SUT at every stage, starting from the BIOS interface, boot up, and OS level. However, there are few challenges faced in past works such as the capture/replay method in GUI automation [26]. This method has a high risk of failure because a system or software with various loading times that causes automated scripts can be easily mistimed. The other challenge faced is the image recognition tools based on the host-dependent method can only automate the mouse position and does not support any UI system method that only accepts keyboard as an input device such as BIOS. Our proposed method offers a solution to all limitations faced by previous proposed methods.

3. RESEARCH METHOD

This section explains our proposed method for UI automation process on BIOS validation. Figure 1 shows the overview of the proposed method. The explanation about the AUTA project are divided into three primary phases: Phase 1: KM emulator, Phase 2: OCR improvement in BIOS interface, and Phase 3: AUTA library.

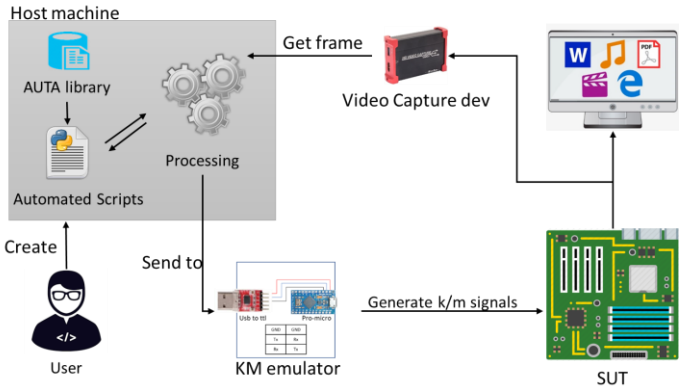

Figure 1. Overview of the AUTA project

3.1. Phase 1: KM emulator

Keyboard and mouse emulator is developed in phase 1 to send keyboard keystrokes and mouse motions from a host machine to an SUT. A tiny microcontroller (pro micro) with the ATmega32U4 chip is used to emulate keyboard keystrokes and mouse motions. USB to TTL converter module is required to connect the microcontroller with a host machine. CP2102 module with 300 bps to 1.5 Mbps Baud rates is used to transmit data from a host machine to the pro micro. The microcontroller is controlled from the host machine through serial communication by using a set of predefined instructions. Incoming instructions from the serial port are split into two parts as head and body. The first three characters are considered as head and the remaining characters are considered as body. Head-data is used to describe the required function from the microcontroller. Table 1 shows the command line instruction examples.

Table 1. How the host machine uses serial communication

Functions	Predefined Head-data	Command-line	Example
Keyboard printing	Kp:	Kp:text	Kp:hello word
Keyboard buttons	Kb:	Kb:button1+button2+….	Kb:ctrl+alt+delete
Keyboard holding	Kh:	Kh:button,state	Kb:ctrl,1
Mouse moving	mm:	Kp:direction,pixcel	mm:up,25
Mouse clicking	mc:	Kb:button	mc:r
Mouse position	mp:	mp:X , Y	mp:265,801

3.2. Phase 2: OCR improvement in BIOS interface

An OCR is used to recognize text in BIOS menu as shown in Figure 2. It requires an improved image to function accurately. In such a case, image analyzing and improving are two main important things in this project. OpenCV is used to capture the video frame from the target device and execute the frame's image processing. Tesseract-OCR is used to recognize the character in the image. Image improvement in the BIOS interface can be divided into three main parts which are:1) image analyzing on selected text (IAST), 2) image analyzing on text menu (IATM), and 3) image analyzing on pop-up menu (IAPM).

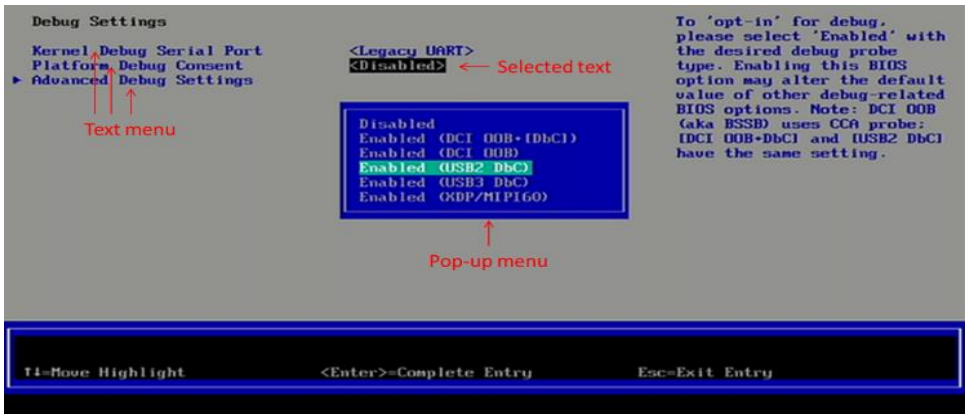

Figure 2. The target text on the BIOS graphic user interface

3.2.1. IAST

Image analyzing on selected text (IAST) function is developed to recognize the selected text in the BIOS GUI, usually highlighted with a black box. The function has seven stages before going to OCR, as shown in Figure 3. Each frame has to go through all the stages. Convert to grayscale stage is converting each frame to grayscale image. Then it has to undergo noise removal to increase the accuracy in image thresholding and find contours stages. The approxPolyDP function is used to detect polygon with four-sided shape and height that is not exceeding 25 pixels. The detected polygon will be cropped and reverse colur to increase the detection accuracy in OCR. The outputs of this function are the location of the highlighted text and its extraction.

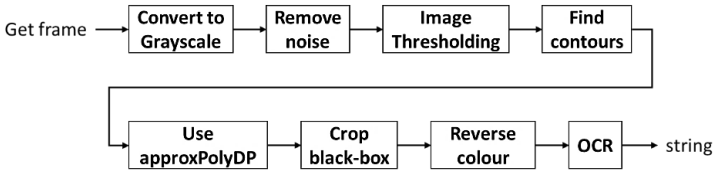

Figure 3. IAST stages

3.2.2. IATM

Image analyzing on text menu (IATM) function recognizes all the text menu in the BIOS GUI except the selected text. This function contains four stages, as shown in Figure 4, which is less than IAST because it does not need much pre-processing. After the frame has been coverted to grayscale the data from character from the images are extracted. The text position will be located and arranged in order to assist the selection process later.

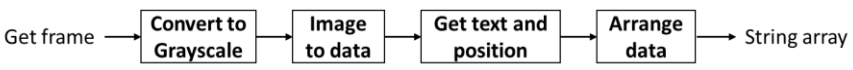

Figure 4. IATM stages

3.2.3. IAPM

Image analyzing on pop-up menu (IAPM) function recognizes the text on the pop-up menu from the BIOS GUI. This function contains ten stages to improve OCR accuracy before going to OCR, as shown in Figure 5. Most of the stages used are the same except in IAPM there is an erosion stage used to removes pixels on selected object boundary.

3.3. Phase 3: AUTA library

This section discusses functions in the AUTA library developed using python language, which will help a tester to create an automated script by calling available functions. Python is one of the most popular and powerful programming languages with the capabilities on cross-platform interpreter.

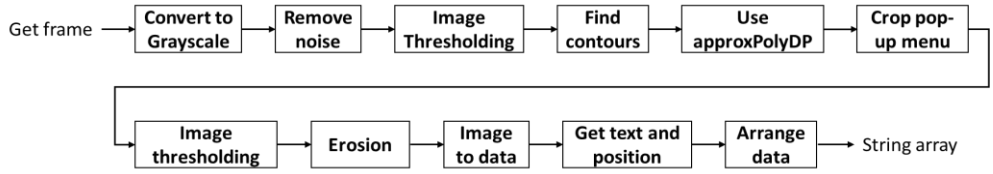

Figure 5. IAPM stages

3.3.1. Bios_goto function

One of the firmware test engineer routine is to go through BIOS menus to verify BIOS settings. Different test cases in BIOS validation requires different settings. BIOS menus and settings do not have a fixed position, and with every new firmware release, it may change their position or order. Human eyes can easily recognize a change in a position, but a machine does not have this recognition ability. The image analyzer and keyboard emulator are combined to solve the recognition problem and allows the host machine to detect and select the correct menu. Two functions have been created using Python code that are selecting the required BIOS menu and selecting the pop-up menu's required setting.

There are few steps involved in selecting the BIOS menu. The first step is to scan the currently selected menu using the IAST function. If it is matched with the required menu, then the function will return True Value. If it does not match, the black box location will be saved inside the variable and goes to the next step, scanning the whole current page using the IATM function. If there is a match, the IATM function will return the position of the matched menu. The keyboard emulator will send either arrow up or arrow down keystroke, depending on the required menu's location. IAST will be used every time the keyboard emulator sends a keystroke to ensure that the required menu has been selected. If there is no match, the keyboard emulator will send a page down button and repeat all the steps until the last page. If all pages have been checked and no matched menu, then the test will stop, and the test result is considered as failing.

To select the BIOS setting in the pop-up menu, the first step is to use the IAPM function to detect and crop pop-up menu and do some image enhancement on the text to make it clearer. The second step uses the IAST function to scan the selected text and compare it with the required setting. If it is matched, then the function will return true value. If it does not match, the black box or selected text location will be saved inside the variable and goes to the next step, scanning the pop-up menu using IAPM and IATM functions. If there is a match the IATM function returns the position of the required setting. By comparing those positions (actual and target), the keyboard emulator will send the keystroke to the required setting. IAST function is used upon every keystroke to ensure the required setting is selected. If there is no matched, the test is considered as failure and the function is terminated.

3.3.2. Mouse_movetotemplate function

The task to include a mouse position function within an automated script is not an easy task for a tester due to the difficulty in obtaining the targeted area exact coordinates. To solve this issue, the template matching method is used. The thresholding value of 0.8 is used to recognize the region of interest on the screen covering text, symbol, and button. Then the mouse position function will point towards the location.

3.3.3. Waitfor_template function

Instead of using estimated time or delay time for the next step, this function is used to keep the automated script in a hold state and consistently check in SUT's screen until matching the template.

4. RESULTS AND ANALYSIS

This section discusses the AUTA's result and its comparison with the manual tester implementation in terms of the completion time. The results are obtained through several specific test cases implemented on the real test environment. An application with a GUI interface is developed based on the AUTA library to test KM emulator functions on different platforms. The test results showed that KM emulator functions worked perfectly on different platforms such as BIOS interface, EFI-shell, Windows, and Linux.

For the OCR improvements, all three functions are tested on BIOS menus and consistent results are obtained with high accuracy detection in alphabet characters reaching 90.32%, which covers capital letters A-Z, small letters a-z and numbers 0-9. In (1) is used to measure OCR accuracy. Although the frame has been enhanced with multiple methods, some of the alphabet and number characters have similar shape which confuses the OCR to distinguish between those characters. The confusing character list is:

– I⇔1 ---> I(alphabet) ⇔ 1(number).
– O⇔0 ---> O(alphabet) ⇔ 0(number).
– U⇔V ---> U(alphabet) ⇔ V(alphabet).

$$OCR\ accuracy\ \% = \left(1 - \frac{incorrect\ characters\ detection}{total\ characters}\right) \times 100 \quad (1)$$

BIOS menus containing the confusing characters will reduce the OCR accuracy. To improve the OCR accuracy, we investigate OCR output that involves confusing characters to find any relationship between OCR input and output. We tested OCR on different BIOS menus and found out that each incorrect character has only two output possibilities: actual character and similar shape character. To solve this issue, we created a function that can generate all possible outcomes, as shown in Figure 6. The number of possibilities will increase exponentially based on the number of incorrect characters 2^n where n is equal to the number of incorrect characters.

To test the AUTA library, one automated script have been created using our library for 15 random test cases in BIOS validation. It took around 42 minutes to complete all 15 test cases. Each test case has different completion time. The estimated time to complete 125 cases using AUTA is around 4-6 hours compared to 48 hours completion time by a tester, as shown in Table 2. Test automation has many advantages compared to the test performed manually. Automated tests can be run repetitively with comparably lower costs, less time, and low error.

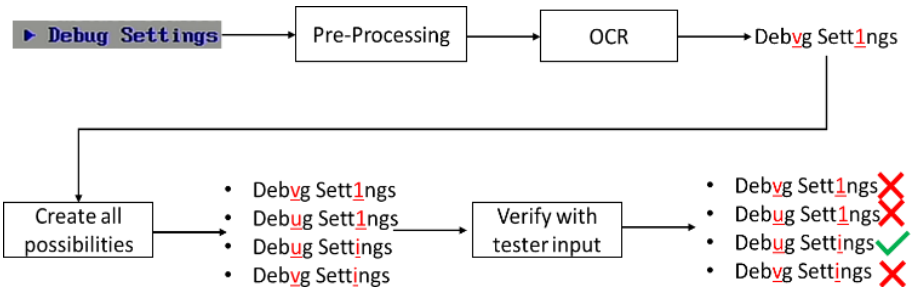

Figure 6. Example on how to create all possible outcomes from the text with suspected characters

Table 2. Comparison between manual and automated tests

	Manual test	Automated test
Time-consuming (average)	48 hours	4-6 hours
Efficiency	Low	High
Testing time flexibility	Working hours only	Anytime
Cost	High	Low

5. CONCLUSION

In this work, we have proposed a method to automate BIOS validation using OpenCV and OCR. This work has shown that by having a fully automation BIOS validation it could reduce the test time by a factor of eight. The image enhancing and processing technique proposed in our method has increase the accuracy up to 90%.

REFERENCES

[1] S. Braun, F. Elberzhager, and K. Holl, "Automation Support for Mobile App Quality Assurance - A Tool Landscape," in *Procedia Computer Science*, vol. 110, pp. 117–124, 2017, doi: 10.1016/j.procs.2017.06.129.
[2] M. Portolan, "Automated Testing Flow: the Present and the Future," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, 2019, doi: 10.1109/tcad.2019.2961328.
[3] A. Ismail, S. Intu, S. Zambri, "A GUI-driven prototype for synthesizing self-adaptation decision," in *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, pp. 792-800, 2020, doi: 10.11591/eei.v9i2.1716
[4] S. Jamaludin, N. Zainal, and W. M. D. W. Zaki, "Deblurring of Noisy Iris Recognition," in *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 1, pp. 156-159, 2021, doi: 10.11591/eei.v10i1.2467
[5] S. Suhartono, F. Nugroho, M. Faisal, M.A. Yaqin, and S. Suvanta, "Speaker Recognition in Content-based Image Retrieval for a High Degree of Accuracy," in *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 3, pp. 350-358, September 2018, doi: 10.11591/eei.v7i3.957

[6] N. Moreira, J. Lázaro, U. Bidarte, J. Jimenez, and A. Astarloa, "On the Utilization of System-on-Chip Platforms to Achieve Nanosecond Synchronization Accuracies in Substation Automation Systems," *IEEE Trans. Smart Grid*, 2017, doi: 10.1109/TSG.2015.2512440.
[7] N. H. M. I. Zaihani, R. Roslan, Z. Ibrahim, K. A. F. A. Samah, "Automated segmentation and detection of T1-weighted magnetic resonance imaging brain images of glioma brain tumor," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 3, pp. 1032-1037, June 2020, doi: 10.11591/eei.v9i3.2079.
[8] H. Dibowski, J. Ploennigs, and M. Wollschlaeger, "Semantic Device and System Modeling for Automation Systems and Sensor Networks," *IEEE Trans. Ind. Informatics*, 2018, doi: 10.1109/TII.2018.2796861.
[9] M. Narayana, R. R. Reddy, Hyndavi R. N., "High speed script execution for GUI automation using computer vision," in *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 231-236, February 2019, doi: 10.11591/ijece.v9i1.pp231-236.
[10] H. Hamledari, B. McCabe, and S. Davari, "Automated computer vision-based detection of components of under-construction indoor partitions," *Autom. Constr.*, 2017, doi: 10.1016/j.autcon.2016.11.009.
[11] C. Jittawiriyanukoon, "Proposed algorithm for image classification using regression-based pre-processing and recognition models," in *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 2, pp. 1021-1027, April 2019, doi: 10.11591/ijece.v9i2.pp1021-1027.
[12] H. J. Park, K. B. Kim, "Depth image correction for intel realsense depth camera," in *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 2, pp. 1021-1027, August 2020, doi: 10.11591/ijeecs.v19.i2.pp1021-1027.
[13] W. Dai, L. Riliskis, P. Wang, V. Vyatkin, and X. Guan, "A Cloud-Based Decision Support System for Self-Healing in Distributed Automation Systems Using Fault Tree Analysis," *IEEE Trans. Ind. Informatics*, vol. 14, no. 3, pp. 989–1000, Mar. 2018, doi: 10.1109/TII.2018.2791503.
[14] S. Huang, C. Zhou, N. Xiong, S. H. Yang, Y. Qin, and Q. Zhang, "A general real-time control approach of intrusion response for industrial automation systems," *IEEE Trans. Syst. Man, Cybern. Syst.*, 2016, doi: 10.1109/TSMC.2015.2469688.
[15] A. F. H. Alharan, H. K. Fatlawi, and N. S. Ali, " A cluster-based feature selection method for image texture classification," in *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 3, pp. 1433-1442, June 2019, DOI: 10.11591/ijeecs.v14.i3.
[16] L. Qiu, F. Yao, G. Xu, S. Li, and B. Xu, "Output feedback guaranteed cost control for networked control systems with random packet dropouts and time delays in forward and feedback communication links," *IEEE Trans. Autom. Sci. Eng.*, 2016, doi: 10.1109/TASE.2014.2353657.
[17] A. AlQaisi, M. AlTarawneh, Z. A. Alqadi, A. A. Sharadqah, "Analysis of color image features extraction using texture methods," in *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 17, no. 3, pp. 1220-1225, June 2019, doi: 10.12928/telkomnika.v17i3.9922.
[18] Y. P. Cheng, C. W. Li, and Y. C. Chen, "Apply computer vision in GUI automation for industrial applications," *Math. Biosci. Eng.*, vol. 16, no. 6, 2019, doi: 10.3934/mbe.2019378.
[19] T. Yeh, T. H. Chang, and R. C. Miller, "Sikuli: Using GUI screenshots for search and automation," in *UIST 2009 - Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, 2009, doi: 10.1145/1622176.1622211.
[20] P. Ramya, V. Sindhura, and P. V. Sagar, "Testing using selenium web driver," in *Proceedings of the 2017 2nd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2017*, 2017, doi: 10.1109/ICECCT.2017.8117878.
[21] Y. P. Cheng, D. Liang, and W. J. Wang, "KORAT - A platform independent test automation tool by emulating keyboard/mouse hardware signals," in *AUTOTESTCON (Proceedings)*, Oct. 2016, vol. 2016-October, doi: 10.1109/AUTEST.2016.7589572.
[22] F. Natalia, H. Meidia, N. Afriliana, J.C. Young, S. Sudirman, "Contour evolution method for precise boundary delineation of medical images," in *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no. 3, pp. 1621-1632, June 2020, doi: 10.12928/TELKOMNIKA.v18i3.14746.
[23] L. Belhadef, Z.M. Maaza, "Lossless 4D Medical Images Compression Using Adaptive Inter Slices Filtering," in *International Journal of Advances in Applied Sciences (IJAAS)*, vol. 7, no. 4, pp. 361-368, December 2018, doi: 10.11591/ijaas.v7.i4.pp361-368.
[24] T. Intharah, M. Firman, and G. J. Brostow, "RecurBot: Learn to auto-complete GUI tasks from human demonstrations," in *Conference on Human Factors in Computing Systems - Proceedings*, 2018, doi: 10.1145/3170427.3188532.
[25] T. D. White, G. Fraser, and G. J. Brown, "Improving random GUI testing with image-based widget detection," in *ISSTA 2019 - Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, doi: 10.1145/3293882.3330551.
[26] Z. Yu, P. Xiao, Y. Wu, B. Liu, and L. Wu, "A Novel Automated GUI Testing Echnology Based on Image Recognition," in *Proceedings-18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016*, 2017, doi: 10.1109/HPCC-SmartCity-DSS.2016.0031.