GUI-Based Automated Testing Tool with Exploratory Behavioral Analysis for Enhanced Software Quality Assurance

Sasikala Devireddy$^1$, S M K Sukumar Reddy$^2$, K.G.S.Venkatesan$^3$

$^1$Department of Computer Science and Engineering, Vaagdevi Institute of Technology and Science, Proddatur, Kadapa District, Andhra Pradesh, India.

$^2$Department of Electronics and Communication Engineering, Vaagdevi Institute of Technology and Science, Proddatur, Kadapa District, Andhra Pradesh, India.

$^3$Department of Computer Science and Engineering, MEGHA Institute of Engineering and Technology for Women, Edulabad, Hyderabad, Telangana, India.

Received: 05.09.2023  •  Accepted: 19.09.2023  •  Published Online: 30.09.2023

Abstract: Testing web-based apps is difficult because of the intricate nature of these programmes. Automated methods used in computerised testing make ensuring that tasks are distributed fairly and reduce the influence of human mistake. This article’s goal is to provide our system for automating the testing of web applications. It was possible to effectively construct this state-of-the-art automated testing framework thanks to the Selenium WebDriver tool. As a result of this setup, PyAutoGUI may be used to automatically traverse through websites.

*Correspondence: Assistant Professor, Department of Computer Science and Engineering, Vaagdevi Institute of Technology and Science, Proddatur, Kadapa District, Andhra Pradesh, India. Email: sksasireddy@gmail.com

https://doi.org/10.58599/IJSMIEN.2023.1902

Volume-1, Issue-9, PP:11-19 (2023)

This work is licensed under a Creative Commons Attribution 4.0 International License CC BY-NC-ND 4.0.
and match widgets to a pre-trained dataset. With the framework’s screenshot feature, designers may examine their previous work with more ease. Our team has come up with a novel approach to finding, categorising, and rating widgets by using machine learning and image processing techniques. Using this method, you may quickly get the images needed for the GUI widgets from the server by inputting the URL for the training data. Datasets are checked for the presence of GUI widgets using BLOB text detection. After locating all widgets, they are classified according to their various uses. Widgets may take several forms, including labels, buttons, input fields, checkboxes, and links. Finally, we make sure the overall GUI component categorization is as precise as possible by performing a series of checks. In order to identify the different kinds of widgets, our project used image processing rather than graphical user interfaces coded in languages like Java, Python, or Visual Basic. Test coverage was increased to 98.4 percent after we successfully implemented automated testing for online apps.

**Key words:** Automated Testing, Graphical User Interface, Quality Assurance Tools, Test Framework.

1. **INTRODUCTION**

The bulk of modern software relies on a GUI, or graphical user interface. Both the user interface (UI) and the programming that supports it must undergo rigorous testing to ensure proper functioning [1]. Several image processing (IP) processes are called back from the GUI mapping to perform tasks that are typically associated with IP capabilities [2]. Picture recognition and object categorization are two examples of these image processing operations. Once a picture has been uploaded in one of these formats, it may be altered in many ways with the help of the provided editing tools. The software infrastructures necessary for web-based applications are essential to their efficient operation. Therefore, it is of utmost importance that web applications excel in implementation, or the act of putting certain procedures into action. It is crucial to run the program’s tests rapidly and frequently [3]. Automated testing offers the opportunity of enhancing software quality while decreasing the need for human intervention. Watir, JMeter, Selenium, and QTP are just some of the open-source and commercial technologies that may be used to do the testing procedures [4]. When it comes to testing websites, many developers turn to Selenium, an open-source automation tool. The major goal of this project was to develop an automated testing tool using several computer vision methods. The many methods of image processing, such as feature extraction, widget identification, widget categorization, comprehensive testing, and comparison, are all explored in this study.

2. **RELATED WORK**

The findings in [1] suggest that employing machine learning techniques to identify and categorise widgets might be a huge help when evaluating user interfaces. It was also shown that a training sample, URL linkages, and screen connections could be used to recognise, classify, and report on GUI components in screenshots based on their positions (x, y coordinates) and types. If machine learning algorithms could automatically distinguish GUI widgets in screenshots, it might greatly cut down on testing time [2]. As training data for automating the extraction of information from widgets, we
produce randomly generated user interfaces. To translate UI design pictures into GUI blueprints, we built a neural device decipherer [3] that makes use of state-of-the-art methods in computer picture and instrument interpretation. We use a hybrid approach to GUI element detection that draws from both deep learning approaches [4] and more conventional methods grounded on more familiar image processing aspects. Unfortunately, the need for more exact localization in the name of GUI components was overlooked throughout the development of these CV approaches [5]. Image characteristics are extracted using Convolutional Neural Networks (DNNs), and significant tags are represented using word embed vectors. We begin by giving you a set of models inside the Canonical Correlation Analysis (CCA) framework to utilise for analysing both the textual and visual components of your data. The major goal of the User Interface Element Detection (UIED) toolkit is to provide individuals with a reliable technique of identifying what various GUI components are. In its pursuit to understand the breadth and depth of GUI graphical user interfaces, UIED makes use of a wide variety of detection methods, including traditional computer vision (CV) approaches and deep learning models [6]. Later applications have found UIED to be helpful in aiding in-depth identification. The greater the number of variables examined, the broader the study’s scope [7]. In this method, the code is executed in a "sandbox," or a secure, isolated space. When auditors are tasked with manually running a large number of programmes with limited resources, productivity suffers. In mobile applications, UI actions often stand in for inputs. Modern software development [8] includes system testing to make sure all requirements are met consistently. The interface is an abstraction layer that hides the underlying system from the user. Automated tests for graphical user interfaces (GUIs) share the goals of other types of automation in this regard. Several kinds of tests (such as unit and integrated testing) have been proved to save money thanks to automated testing in actual industrial practises. This is due to the fact that manually testing every potential GUI situation is both time-consuming and error-prone. We recommend using the open-source and free software TESTAR to supplement scripted testing with automated testing. Jonathan A. Saddler is responsible for developing Event Flow Slicer. This application may be used by a GUI tester to build and organise all the tests required to reach an objective. The user must first record an action in the UI in order to limit its scope. You may choose and choose the resources you need to complete a task with the help of a timeline of events. The author has trained a network to predict where and how large widgets should be put in a display [9]. This has ramifications for the availability and methodology of GUI testing. Our SM may be useful in determining future areas of study in the field of GUI testing by providing a picture of existing methods. For example, bridging the gap between theoretical model-based methodologies and commercially available technologies involves significant effort. Current tools will need to be compared to the most advanced graphical user interfaces (GUIs) that have been the focus of academic research. Deep learning is a framework that uses state-of-the-art deep icon-behavior learning to train and uncover intention-behavior mismatches in a variety of popular applications. In particular, deep learning uses programme analysis methods to infer labels for UI widgets from developer behaviour. This means that
a large, high-quality training sample may be produced. The requirements of the output writers do not align with the aims of the developers or the designs of the user interfaces they create. Second, the programme will lose its unique selling point if other developers are also using similar user interface (UI) ideas. Finally, it may be challenging for programmers to keep up with modern GUI design trends if a significant portion of the retrieved GUI is based on an outdated design style.

3. PROPOSED APPROACH

The primary goal of this study is to provide a methodology for developing automated prototype testing software for web applications and to offer examples of its use. Furthermore, to explore and assess the use of computer vision techniques to software testing. This design use the Python OpenCV library’s testing concept assessment archives to identify the individual GUI widgets that are on display. Our new approach also eliminates the need for massive amounts of data showcasing GUI widgets, because the URL is provided as an input. This is so because we are the ones supplying it. The only way to tell a GUI apart from another is by its proportions and any immediately noticeable colour differences. The ideal dataset, prepared by the experts, includes both the input data and the evaluations of the replies. This development inevitably leads to more precise efforts like distinguishing, categorising, evaluating, and analysing. We may infer the relationship between unproven data sets and GUI graphic widgets using computer visualisation technologies. Labels, buttons, input fields, checkboxes, and links are all examples of widgets that need human classification before they may be sorted into their respective domain categories. Finally, it is calculated to enable the determination of an average GUI component categorization. Our studies have focused on image processing methods rather than graphical user interfaces like Java GUI, Python GUI, or Visual Basic GUI to determine the types of widgets. Machine learning-based image processing that can mimic human vision and analyse photos automatically.

According to the documentation of Selenium web-driver, the utilization of screenshots is discouraged for both successful and failing test cases. New methods for capturing screenshots in both normal and error states have been implemented. By employing this approach, a function tester can readily detect flaws within a web application. This method enables the designer to conduct a more thorough examination of their testing. Once a series of tests has been conducted and either deemed successful or unsuccessful, the corresponding screenshots of the widget links are stored in the catalog of a database that is capable of network communication. The methodology for capturing a screenshot:

Initially, there exists a build catalog that facilitates the storage of screenshots depicting both successful and failing widgets and links. Identify the results of conducting selenium web driver testing. Determine whether the result may be classified as a success or a failure. In addition, a capture of the website’s interface is obtained irrespective of the result. In order to incorporate the name of the network, it is necessary to rename the snapshot file. It is recommended to save the photo files in distinct directories. The Selenium web-driver is capable of supporting a diverse range of sensors in order to locate page components. The execution of automated screenplay languages is a prerequisite for the
The operation of the selenium web driver. The compatibility of the selenium client library with the chosen major programming language is necessary. Python users now have access to the whole capabilities of the Selenium web driver client library.

The selenium neutral server facilitates decentralized and autonomous testing of browser-based functions. One can navigate across a website by utilizing various navigational aids like labels, text boxes, combo boxes, buttons, and other similar elements. The various widgets present on a website are contained within the entity source. Consequently, maintaining the training environment in a current and efficient state will be facilitated. The 'Login' button, for example, has been a fundamental component of web-based applications since its creation. In subsequent iterations of the web application, the 'Login' button had a modification and was subsequently relabeled as the 'Login Now' button. Consequently, it became necessary to provide additional training for the widgets. In order to proactively address potential challenges, a repository of objects has been established and populated with the examples that have undergone training. There will be a reduced requirement for storage space, as well as decreased expenses associated with upkeep. If there are any changes in the structure of the web application's components, it is unnecessary for the testing team to update the object repository. In contrast to the current architectural framework, the training examples of this approach do not depend on extensive datasets. The present study utilized the Frozen East Text Detector Dataset. The dataset utilized by the OpenCV EAST text detector is a prototype of a Deep Neural Network (DNN) that incorporates a novel structural architecture and guiding layout. The utilization of advanced technology enables the system to efficiently process high-definition images with a resolution of 720p at a rate of 13 frames.

Figure 1. Block diagram of the proposed system
per second (FPS). Additionally, the system exhibits commendable accuracy in recognizing and interpreting textual content. Researchers have utilized OpenCV’s EAST sensor to facilitate the automatic detection of text in both still pictures and videos.

To navigate the multiple links contained in an n-url, we have been utilizing the PyAutoGUI automation python module. Instead of employing a DNN classifier, you can categorize the GUI with OpenCV’s object recognition method and check the following snapshot for additional discrepancies. If a GUI element looks different from the original GUI image, it may not have been designed to act as a hyperlink or a button. We’ve developed a brand new tool, the “image decision function,” to handle these sorts of tasks. The algorithm for the image decision function is given below:

![Algorithm for Image Decision Function](image.png)

OpenCV, a freely available open-source computer vision library, was used in this project. A free and accessible program that use computer vision and object recognition to manage icon libraries. Surprisingly, this assignment is the shortest when compared to others that involve the prototyping of images on computers. Whether you’re working with a regular webcam or a high-precision depth sensor, the OpenCV library in Python can swiftly analyze video, set up still images, and locate objects. The ability to take images, edit them, and share them is severely limited. Because the OpenCV Python extensions use a standard data format that is compatible with scientific libraries like NumPy, beautifulsoup, and imutils, they can be used to aid in the discovery of proofs for these requirements. Information gathered from copies of old jumping containers for GUI components is not permanent due
to restrictions in photo-editing tools or shifts in design processes such as drawing mock-ups digitally or physically with pen demos, tablets, or paper. Because a mockup artifact may only exist as an image in certain scenarios, CV techniques must be employed to extract crucial GUI-component data. To address this problem, our approach employs CV algorithms to ascertain the extents of GUI components. Like the GUI processes in an image, this method maintains a number of CV approaches that can infer joining frames between pieces. To begin, we utilize the BLOB text detection technique to find the boundaries of items in the image. The merging of adjacent edges is facilitated by their enlargement. Finally, the edges are utilized to map out where the frames will leap over the various GUI-components.

It suggests a novel approach to measuring academic development. We took advantage of the Python automation module PyAutoGUI to programmatically activate each of the URL's many links. After that, a DNN classifier may be hired and fed a screenshot of the final result to make the necessary adjustments and classify the user interface. A screenshot is considered to be of a Hyperlink/Button UI if it significantly deviates from the prototype UI that was first envisioned. The Input boxes can be identified by their corresponding cursor pictures, whereas the Combo Boxes can be identified by the list of alternatives that is provided on the same page. This evaluation method has to be tested on a variety of active locations to ensure its accuracy and validity. As a result, the proposed approach can still yield accurate results in detecting and classifying GUI components even in the absence of controlled training instances. In order to find areas in a digital image that are different from their surroundings in terms of properties like brightness or color, computer vision is used in blob text detection approaches. This technique is necessary for separating the recovered text blobs into horizontal text segments. In citation 21, this procedure is called "text segmentation." This categorization takes into account the relative sizes and placements of the blobs. A combination of an image classification technique and an exploratory behavioral analysis algorithm is used to categorize the widgets. A computer software is used for this purpose.

4. RESULT AND DISCUSSION

Existing studies use a deep learning approach for GUI creation, with supervised training. The graphical user interface (GUI) components are classified with the help of deep neural network (DNN) architecture. The DNN model is trained with a dataset consisting of labeled images from a central repository. After all is said and done, it is used to mimic an input image and harvest GUI code. This concept opens up a wealth of interesting possibilities for the assessment of software GUIs. Several methods have emerged to evaluate the accuracy of automated testing as a direct result of this worry. The quality of the test coverage is compared and contrasted between the proposed method and the current setup. After amassing links using the proposed method, they are compared and sorted using the same criteria as the preexisting system’s connections. Root Mean Square Error (RMSE) is used to measure the accuracy of the new system, whereas Mean Squared Error (MSE) is used to measure the accuracy of the current system. Root-mean-squared error (RMSE) provides a graphical representation of the testing model’s performance by measuring the degree to which expected and observed values of a given
parameter differ. Therefore, better performance is represented by a smaller root mean square error (RMSE) value. Additionally, it has been established that the suggested testing model has a smaller root mean square error (RMSE) than the current testing model. The Root Mean Square Error is illustrated in Figure 2.

![Figure 2. Illustrating the Root Mean Square Error](image)

Finally testing coverage and categorization accuracy of existing and proposed system is tabulated and presented in Table-1.

<table>
<thead>
<tr>
<th>Existing system categorization Accuracy in %</th>
<th>Proposed system categorization Accuracy in %</th>
</tr>
</thead>
<tbody>
<tr>
<td>97.3</td>
<td>99.4</td>
</tr>
</tbody>
</table>

Accuracy with the actual numerical values obtained from your testing and analysis. This table format provides a clear and concise way to compare the testing coverage and categorization accuracy between the existing and proposed system is 97.3%.

5. CONCLUSION

In this research, we present a fresh automated examination framework for evaluating selenium web-driver’s web-based testing abilities. The time required to create test cases is cut in half, and the test coverage pass rate is increased, when an effective automation framework is used to evaluate a web application. In addition, it reduces the stress and anxiety experienced by the tester. Using this method, one can examine screenshots of failed test runs in order to develop exact test assertions and investigate software flaws. The test manager has full access to all information in one convenient location. This approach proves to be quite efficient when making significant changes to existing web-based applications. In this study, we explore how an automated framework for testing web applications
in enterprise settings might help save money without sacrificing quality. This research also looks at the automation testing of real-time web applications’ user interfaces and user experiences. Advantages of the aforementioned device include low ongoing costs and a short time required to recoup initial investment.

References


