Testing mobile games with OCR

Yuan Tu s1252005 Supervised by Prof. Maxim Mozgovoy

Abstract

This research is dedicated to the possibility of integrating Optical Character Recognition into game automated GUI testing. Besides manually functional testing by QA(Quality assurance) stuff, there are many different ways to test a mobile application like automated smoke testing and automated GUI testing. OCR (Optical Character Recognition) is a great method to recognize the text from pictures, which is now widely used in scanning documents, translating applications, and auto drive. etc. However, there is not much research using OCR for mobile application automated GUI testing, which arouses the interest of integrating OCR into automated GUI testing with well-trained Tesseract Open Source OCR Engine on a tennis game application.

1 Introduction

Game testing plays an important role in game development as game genres getting more complicated and various. Automation testing (Test Automation) is a commonly used software testing technique that performs using special automated testing software tools to execute a test case suite. Smoke testing is one of the automation testing processes to check whether deployed software or game is stable or not. However, there is a problem still yet completely solved that Automated smoke testing is especially challenging for the applications with nonstandard GUI, such as games made with Unity. Previous research [1] studied by M.Mozgovoy and E.Pyshkin utilized Image recognition to solve this issue. Although Image recognition for mobile testing requires saving all the images of User Interface elements that are needed to be tested. This study is based on the research above [1] and to see the possibility of applying Optical Character Recognition with Tesseract OCR into automated GUI testing.

2 Method

For integrating OCR testing into GUI Automation testing, a mobile device with the tested game, a software testing tool and testing scrips are required.

2.1 TestBed

The testbed of this study is a 3D Tennis game developed with Unity which is also able to adapt to Android mobile devices. The game is called Tennis Arena can be seen in Figure 1. This game has both image and text-based buttons, which provides a good platform to test the efficiency of OCR GUI testing and the possibility of merging OCR with Image recognition GUI testing.

Figure 1: Testbed Game: Tennis Arena

2.2 Tools

As mentioned before, this study is based on Unity Application Testing Automation with Appium and Image Recognition [1]. Researchers of the previous study used Appium [2] which is a test automation framework designed to assist functional testing of compiled native (iOS, Android or Windows), and hybrid applications. [3]. In our study, we also used the same tool for keeping testing work. Besides, in order to realize Optical Character Recognition testing, this study utilized Tesseract OCR [4] which is an optical character recognition engine for various operating systems to realize the goal. Tesseract OCR is also used in google translate and was considered one of the most accurate open-source OCR engines then available in 2006. [5]

2.3 Testing Steps

2.3.1 Set Up Testing cases

In order to discover the possibility of using OCR testing, there are two sets of testing are set up by writing specific testing instruction scrips. Appium allows the scrips to access applications in a similar way as to end players. Therefore, the testing script can be used to click the button, wait for specific times, select checkboxes, etc. The first set of testing is set up with 12 basic buttons functionality tests that test if the text-based button can be recognized correctly and function as it's expected. Most of these 12 buttons are high frequently used in most of the games shown as Table 1.

Table 1: 12 Basic button testing set

"LOGOUT"	"RESET"	"OK"
"CANCEL"	"SKIP"	"X" or "×"
"SEASON"	"RATING"	"LVL"
"BACK"	"START"	"PLAY MATCH"

Another set of testing is series of game steps testing, which is combination of several waiting for responding of game and button functionality testing. The purpose of this testing is to verify whether GUI OCR testing can be integrated into regular automation testing routine. This part of testing is set up as follow:

1. Start Match – Quit – Continue – Quit – Confirm
2. Generate New Character – Skip Tutorial
3. Open Season Window – Go back – Open Rating Window – Go Back – Open Level Window – Go back
4. Reset – Confirm: Cancel – Reset – Confirm: OK

2.3.2 Test

Before starting the test, the connection build between the testing device and server is built up stably so the scrips can run smoothly on the device like how the player plays the game. The setup procedure is similar to the procedure in the previous work [1] except in this study there is only one testing device and the server is a local server set up on the computer.
For every singular button functionality test, there are five basic steps.

1. Firstly, get a screenshot from a mobile device when appium received the response of the game is waiting for instruction. Shown as Figure 2
2. Next, we put this screenshot into image processing to make the screenshot easier to be recognized by OCR Tesseract. Shown as Figure 3
3. OCR successfully finds the target text, then return the coordinate of the target button on the screenshot.Shown as Figure 4
4. Then calculates the corresponding coordinate of the button on the mobile screen, return the new coordinate for the mobile device.
5. Click the button coordinate on the mobile device, go to the next step.Shown as Figure 5

Figure 2: Step1:Take a sceen shot

Figure 3: Step2:Image Processing

The series of game steps testing, as mentioned above, is a combination of several waiting for responses and button functionality tests. Therefore, to test series of game steps need to repeat the above 5 steps till the test is over or the test is not able to continue. Also, during this test there are also image-based buttons, in this OCR testing is replaced by image

```
SKIP left top width height
372
333
47
16
```

Figure 4: Step3:Get Coordinate

Figure 5: Step5: Click target button

recognition testing to ensure the testing going continuously and smoothly.

2.4 Image Processing

Image pre-processing is commonly used for OCR. The feature of the image such as resolution, messiness, and low contrast can be the factors that influence the accuracy of the OCR result. In our case, as Figure 2 shows that the texts on the original screenshot can not reach a fair result, so some common process methods such as convert image to grayscaled as Figure 7, binarization, Noise Removal Figure 7 are tried.

After these attempts, a feature of this game has been noticed. More than 95% of texts in this game are in white, aiming this point, the most direct image processing approach is to make pixels not white enough black. Thereby, the messiness of the background will be greatly increased. The processed image is shown as Figure 8.

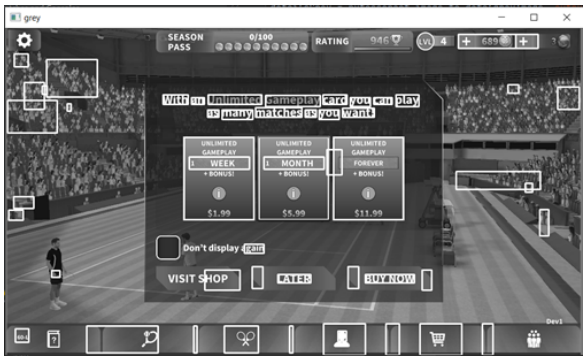

Figure 6: Gary Scaled Screenshot

Figure 7: Denoised and smoothed Screenshot

Figure 8: Aiming to white pixel Processed Screenshot

3 Evaluation

After testing as the method illustrated above, the result of 12 button functionality tests shows as Table 2. In this table, the first volume shows the button's text, the second volume shows the fraction of successful cases over tested cases and the percentage of test pass rate. The total pass rate of this set of tests is 84% which shows that OCR testing is able to test simple text-based button functionality tests. However, there

are also cases that didn't work fairly. For example, "LVL" has a 0% passing rate, the reason for this result is because the shape of this button is a circle with a white outline, which makes tesseract OCR very difficult to ignore the white circle to recognize the inner words. Also "RATING" facing the similar issue: the background is still comparably messy for OCR testing. Besides this factor, too few letters in the word also can create unexpected trouble for OCR. The cases "X" or "×" and "OK" are the example for such situation.

Table 2: 12 Basic button testing set result

"LOGOUT"	5/5 100%
"RESET"	5/5 100%
"CANCEL"	3/3 100%
"SKIP"	9/9 100%
"SEASON"	7/7 100%
"RATING"	6/7 85.7%
"BACK"	5/5 100%
"START"	6/6 100%
"OK"	6/8 75%
"X" or "×"	2/3 66.6%
"LVL"	0/7 0%
"PLAY MATCH"	5/5 100%

And the result of other 4 series of game steps testing shows in Table 3. This table shows that 3 out of 4 tests passed, which also proves that OCR is able to test a series of game steps to continue the automated testing. In conclusion, OCR is able to be integrated into GUI automated testing. Among these testing cases, Generating a new character requires more interaction with the game like selecting a message box and typing a name into it, or shooting the ball during the tutorial section. Those additional steps make this test difficult to finish so that we couldn't say OCR can work well under such circumstances.

Table 3: 4 series of game steps tests result

Quit match series	Finished
Generating Character and skip tutorial	Not Finished
Open windows and go back	Finished
Reset confirm	Finished

4 Conclusions and future work

This study successfully integrated OCR testing into Automated GUI testing by testing both singular text-based button functionality and series of game steps. However, there are many parts of this study that have a large space to improve.

First of all, the image processing algorithm is only aiming for testbed Tennis Arena only, Other mobile games have a very high possibility to use other colors text. Therefore the image preprocessing for mobile game screenshot need to be improved for increasing the accuracy of OCR result.

Secondly, in our case, the target string of OCR testing is too strict. For "LVL", "OK" and long string, if OCR can not recognize them all then the test is failed, however, in real cases, the target string might be partially recognized or be recognized in similar words. In this aspect, OCR testing should be improved to tolerant fragile string or similar string to keep the testing running.

Moreover, J.Tuovenen and M.Oussalah mention that OCR recognition tools tend to be slower than other types of tools because they need to scan the whole screen for the text. [6] During this study testing, the same issue has been realized. OCR testing is indeed slower compare to image recognition testing. For further work, we can try to solve this issue by reducing screen area or improving the image processing algorithm to make OCR recognition tools scan less or scan faster.

Last but not least, nowadays most games are still image-based, although there are amount of text buttons, also image buttons are not avoidable. So one of the next steps of this work could be to find the most efficient way to combine OCR testing and image recognition testing.

References

[1] M. Mozgovoy and E. Pyshkin, "Unity Application Testing Automation with Appium and Image Recognition," *Communications in Computer and Information Science*, vol. 779, pp. 139–150, 2018.

[2] "Appium project homepage," http://appium.io, [Online; accessed 1-Nov-2016].

[3] M. Hans, "Appium essentials," https://www.packtpub.com/application-development/appium-essentials/.

[4] "Tesseract ocr project homepage," https://github.com/tesseract-ocr/tesseract, [Online; accessed 2005].

[5] L. Vincent, "Announcing Tesseract Ocr," http://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html, 2006.

[6] J. Tuovenen, M. Oussalah, and P. Kostakos, "Mauto: Automatic Mobile Game Testing Tool Using Image-Matching Based Approach," *The Computer Game Journal*, vol. 8, pp. 215–239, 2019.