Experiment study of Android Software Test using Moment Invariants Algorithm

Won Shin[1], Tae-Wan Kim[2], Chun-Hyon Chang[1, *]

[1] Dept. of Computer Engineering, University of Konkuk, Seoul, Korea
{wonjjang, chchang}@konkuk.ac.kr[1]
[2]Dept. of Electrical Engineering, University of MyongJi, Seoul, Korea
twkim@mju.ac.kr

Abstract. There are many tools for testing android software. However, previous tools are only focused on a functional test. The conventional tools thus, cannot analyze or detect distorted or broken image on the screen due to characteristic of device and android platform. To resolve this problem, in this paper, we propose an experiment model using a moment invariants algorithm, which is one of image comparison algorithms. In the experiments, we compared a normal screenshot with a test screenshot firstly. Secondly, we analyzed the experiment how much variants are changed. As a result, the resolution is a most important factor to affect an image. And a platform version is not related to the result. Developers can simply use the proposed moment invariants algorithm to find error candidates in a general case.

Keywords: Moment Invariants, Android Software, interoperability, Software Test

1 Introduction

Developers must consider various android platforms and devices during development phase because android does not support interoperability. On the other hands, it is not sure that software has no problem when environments of software are changed. Thus developers have to verify whether their software operates differently, or whether screens become corrupted according to type of devices. To overcome their difficulties, many kinds of supporting tool have been releasing in recent such as JUnit [7], Robotium [12], Android Testing Framework [2]. Those tools provide functionalities such as automatic test-case generation, automatic running test-case and analyzing software logs. Those functionalities are very helpful for developers, however, it is not enough because those tools only focus on functionality of software. For instance, if distorted images or broken image are existed on the screen of devices, the tools do not report it, but users may be uncomfortable during use the software. Therefore, a methodology, which finds strange things in the screen, is needed to developers. To resolve this problem, we survey image comparison algorithm to find differences

* Corresponding author.

among screenshots and then analyze which screenshot is most abnormal image from normal image. In this paper, we explain our several experiments and its results. In the experiments, we use moment invariants algorithm which is shape-based technique and it is effective features for recognition under changing viewpoint and illumination [5].

The rest of the paper is organized as follows. Section 2, presents the experiment environment. Section 3 explains the experiment results. Finally, Section 4 concludes and explains future works.

2 Experiment Environment

A goal of this experiment is to check whether image comparison technique can generate meaningful value for testing non-functionalities, especially usability. To archive the goal, we analyze relationship between android software and effect elements like a platform version, device and resolution.

For experiment, we select several devices and software. There are five kinds of devices in Table 1. Those have different resolution and are made by different manufacture. Also, there are four kinds of software in Table 2. Two of them are game and those have their own images. The others are just made by using default android UI components such as a label, a button and so on.

Table 1. Devices Description

Product Name	Resolution	Manufacture	Platform Version
Galaxy S	WVGA(800*480)	Samsung	2.3
Vega	WVGA(800*480)	Pantech	2.3
Atrix	qHD(960*540)	Motorola	2.3
Sensation	qHD(960*540)	HTC	2.3
Galaxy Tab	WSVGA(1024*600)	Samsung	2.2

Table 2. App Description for Testing

App Name	Description
Android's Fortune	The well-known Fortune for Android: print a random, hopefully interesting, adage [1].
AnkiDroid	A flashcards application for Android [3].
Crazy Penguin	Game. Fend off invading Polar Bears by hurling courageous penguins at them with your trusty catapult [4].
Open Sudoku	A simple open source sudoku game [11].

We use the OpenCV to generate meaningful value which represents difference between two screenshots. OpenCV is a library of programming functions for real time computer vision [10].

To compare screenshots require a basis of comparison, so we choose that resolution is WVGA(800*640) and platform version is 2.2 because it comprise a large proportion of whole devices. API named matchShapes is used to compute the meaningful value and it offers three methods for comparing such as cv_controus_match_i1, cv_controus_match_i2, cv_controus_match_i3 [8]. In this paper, we use m1, m2, m3 as a name of method instead of cv_controus_match_i1, cv_controus_match_i2, cv_controus_match_i3.

3 Experiment Result and Evaluation 3.1

Analyze Experiment Result

There are three experiments for discovering whether developers need to consider about resolution, device and platform version during testing software.

(1) **Experiment 1** : possibility of using emulator screenshot for testing

The purpose of this experiment is to discern difference between screenshot of emulator and screenshot of device. If the difference is existed, it means that developers need to test their software on both of emulator and device, otherwise they just use either one.

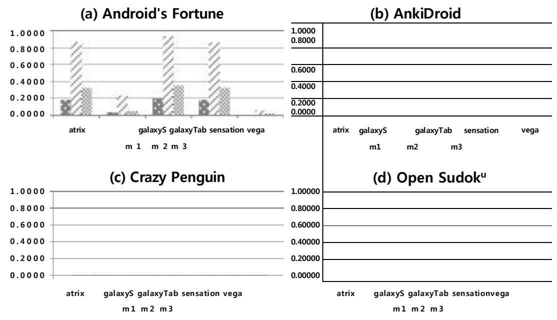


Fig 1. Result of difference between device and emulator

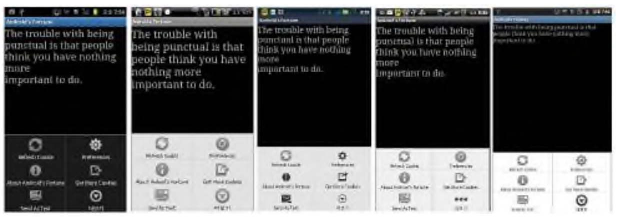

(a) galaxyS (b) vega (c) atrix (d) sensation (e) galaxyTab

Fig 2. Android’s Fortune screenshot on diverse devices

In Fig 1, (a) Android’s Fortune software just has high value but the others have not. A characteristic of (a) is that its components are almost label components and contains lots of letters like Fig 2. Devices have a special their own font type each other, therefore its value must be higher than the others. The result of this means that consideration to use emulator or device is not essential except text based software.

(2) **Experiment 2** : to verify whether platform version influence result

The purpose of this experiment is to verify whether platform version affects screenshot or not. On the other hands, same pictures are generated if capturing different platform versions.

In Fig 3, it is difficult to discern difference among (a), (b), (c) and (d). Only the shape of top is slightly changed from (b) to (c) and (d). A key pad, in the bottom of (d), is optional so it can be hided from the screen.

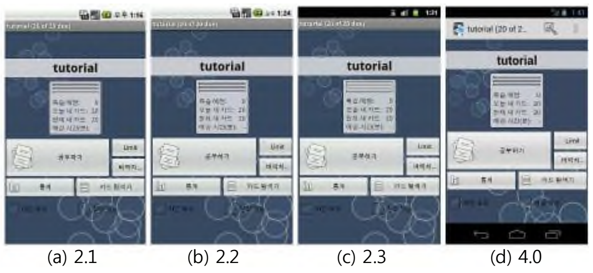


Fig 3. AnkiDroid screenshot on various platform versions

Fig 4 describes version two point three and four point higher than two point one and two point two but it is a shade difference. Version two point one does not even have a value. As mentioned above, android's fortune software only has huge value but, it is not related to platform version.

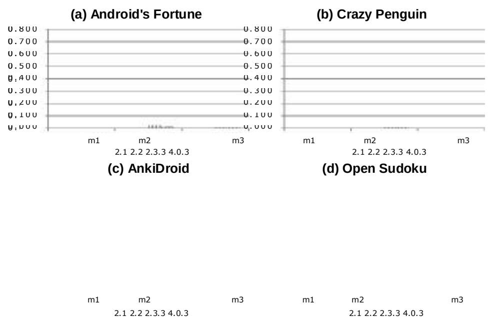


Fig 4. Result of various platform versions

(3) **Experiment 3** : to verify whether resolution affects result

In Fig 5, right side contains distorted image in 854*480. Right sides of 960*540 and 1024*600 are removed, and they are smaller than 800*480. We can expect that 960*540 and 1024*600 would have bigger value than 854*480.

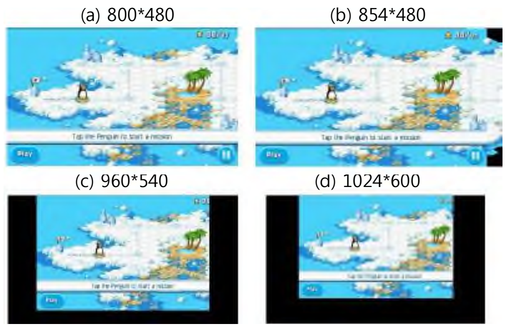
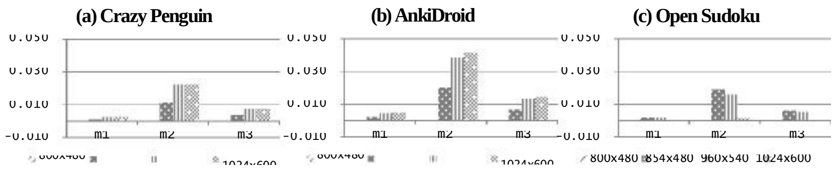


Fig 5. Crazy Penguin screenshot on various resolutions

Fig 6 illustrates 960*540 and 1024*600 have bigger value than 854*480 in (a) Crazy Penguin. Also, m2 have bigger value than m1 and m3 so that it requires consideration when interpreting the result of m2. Moreover, values of (a) and (b) are bigger than (c) respectively. Software (c) consists of android UI components but (a) and (b) have their own image or make a graphic image. It means that android UI components like a button, label and list-box does not



affect a result.

Fig 6. Result of various resolutions

3.2 Evaluation Result

Several factors are discovered by experiments. First of all, platform version does not change a result except special case which software contains many letters because those produce much change. Secondly, it does not matter screenshot of emulator instead of device for testing. Thirdly, if software is made by own image or graphic method, it is quite possibly that there is a lot of distance on the result. Finally, a value from comparing between images is valuable because it describes how many differences are existed. It is clear that each of software has different range of values in the experiments, thus comparing each software values is not meaningful. However, a result from computing same software is valuable because the software may have problem on the screen if value is existed. In conclude, developers can use a value from comparing images to decide whether check their software or not.

4 Conclusion

Android software can make distorted or broken image in a screen of device based on android platform because devices have different resolution, platform respectively. Due to the problem, developers have to check whether android software makes distorted or broken image in screen or not. To resolve this problem, we try to verify that it is possible to use moment invariants algorithm, which is image comparison algorithm. We do several experiments and then prove that platform versions are rarely influential to the result, though, resolution most affect to it. Also, it is possible to test using emulator instead of device.

Developers find problem simply using moment invariants algorithm and reduce the time for testing their software. However, it may take huge time to make images of devices and platforms for testing.

In the future, we develop tool for supporting to compare and analyze images automatically.

Acknowledgments. This research was supported by R&DB Support Center of Seoul Development Institute, Korea, under Seoul R&BD Program (ST100107)

References

1. Android's Fortune, "https://launchpad.net/androidsfortune/"
2. Ankidroid, "http://code.google.com/p/ankidroid/"
3. Android Testing Framework, "http://developer.android.com/guide/topics/testing/index.html"
4. Crazy Penguin, "http://www.digitalchocolate.com/mobile/crazy-penguin-catapult/"
5. Datta, R., Joshi, D., Li, J., and Wang, J. Z., "Image retrieval : Ideas, influences, and trends of the new age", Journal of ACM Computing Surveys, Vol 40, Issue 2, Article 5, 2008
6. Gary Bradski, Adrian Kaehler, "Learning OpenCV : Computer Vision with the OpenCV Library", O'Reilly, 2008
7. JUnit, "http://www.junit.org"
8. Moment Invarients, "http://opencv.willowgarage.com/documentation/c/structural_analysis_and_shape_descriptors.html"
9. Monkey, "http://developer.android.com/guide/developing/tools/monkey.html"
10. OpenCV, "http://opencv.willowgarage.com/"
11. Open Sudoku, "http://code.google.com/p/opensudoku-android/"
12. Robotium, "http://code.google.com/p/robotium/"