# Automatic Mobile App Speed Measurement with Robot

Soohyun Kim, Hyunsu Mun
*Dept. of Computer Science and Engineering*
*Chungnam National University*
Daejeon, Korea
shkim95, munhyunsu@cnu.ac.kr

Youngseok Lee
*Dept. of Computer Science and Engineering*
*Chungnam National University*
Daejeon, Korea
lee@cnu.ac.kr

***Abstract*—Mobile app performance measurement is important to improve user experience by optimizing mobile apps. For this purpose, many mobile app testing methodologies. However, most of mobile app testing software depend on operating systems and binary execution files. In this work, we present a robot-based automatic mobile app speed measurement method. Our tool automates speed testing of mobile apps on either Android or Apple iOS. In particular, we employ a robot arm to simulate human behavior to generate user input events such as touch and swipe. For UI automation with a robot arm, we have devised UI automation by random or clickable component detection methods. We have adopted a web speed index to the mobile app to reflect user experience of performance. From experiments with our robot arm and automation software of generating user events and computing speed index, we have classified fast or slow mobile apps under the speed index metric.**

***Index Terms*—mobile app, speed index, robot, image recognition, machine learning, cnn**

## I. Introduction

Mobile app speed measurement is important to understand and improve user experience. Developers as well as users need to compute the speed of a mobile app when it runs on the device. Many mobile app testing software is publicly available to test performance as well as functions. For instance, Appium[1] provides GUI unit tests for iOS, Android, and Web to perform tests for each component. AppTest.ai[2] is a web-based mobile app test software. If developers provide apk or ipk files of mobile applications to test, Apptest.ai automatically runs test scenarios and finds possible bugs by drawing a dynamic activity graph. Firebase Test Lab[3] is a cloud-based app testing infrastructure that checks the speed and performance of the application. Speed Index is a performance metric used by WebPageTest[4] to quantify how quickly a web page content is rendered and displayed. Speed index is calculated as the accumulated remaining area that has not been rendered or visually complete based on javascript web navigation timing event[5] and snapshots. Since Speed Index indicates the response time or throughput of user screen content, it is directly related to the user experience, which is how fast the user feels. For the comprehensive testing of user experience of the same mobile app, we present a robot-based mobile app speed measurement system to test mobile apps independent of the operating system. Our system consists of a robot arm, a UI analyzer, a speed index analyzer, and a test manager. The robot arm generates a hardware-based touch event. Unlike the existing method for HTML web page performance measurement, which requires a navigation timing event, our method calculates the speed index, which is the visual completion performance of the mobile application, with only the robot and MP4 video.

We carried out robot-based mobile app speed measurement experiments to analyze the performance of 23 Android and 53 iOS mobile applications. Speed index is suitable as a performance index for mobile applications, but we had to implement a speed index calculation module tailored of the mobile environment. Then, we calucated the speed index using MP4 and robot click time file. In addition, we classify clickable elements using CNN image analysis for automated experiments without test scripts.

[1]http://appium.io/
[2]https://apptest.ai/
[3]https://firebase.google.com/docs/test-lab/
[4]https://www.webpagetest.org/
[5]https://www.w3.org/TR/navigation-timing-2/

## II. Related Work

### A. Robot Testing

Although image analysis and machine learning technologies are often used for app testing, they are still in the infant stage that relies on partial image recognition of specific images in an application rather than finding click elements for automatic testing.

Robots have been used by many studies for testing mobile apps [1], [2]. To test the application using a robot, it is necessary to analyze elements in the application through video and image analysis. Craciunescu proposed a screen analysis method to test a mobile application using a robot [2]. The screen analysis method finds a corresponding icon on the mobile screen when a specific application icon or a specific icon in the application. SURF and SIFT, which are image analysis algorithms, were used for mobile phone screen analysis. In their experiments, the maximum of 77.59 % and 24.31 % can be used to find icons on the mobile screen for SURF and SIFT, respectively.

In Jih-Gau Juang's study [3], three cameras were used for a robot test to distinguish letters. On one camera, hue, saturation,

and brightness were transformed to match the pattern of letters, and two cameras were configured to differentiate between left and right. In the experiment to distinguish letters from 1 to 9, they showed a success rate of 100 % except for classifying three letters.

### B. Image Analysis

Image analysis has been widely used to classify mobile application activities. Ariel Rosenfeld [4] proposed an activity classification method using machine learning to automate the functional testing of Android applications . In addition to application performance and functional testing, image analysis was also used in research to find sensitive icons that could expose personal information. Xusheng Xiao [5] conducted a study to distinguish all widgets that could leak sensitive personal information data that requires user permission from the UI . Sensitive personal information was divided into 8 categories. Image buttons corresponding to each category were classified using the image and text analysis, and similar images were classified using an image analysis algorithm. Similar images and texts were found using UI layout file analysis.

### C. Speed Index

The speed index is a performance metric on how quickly a web content is rendered to display the content to the user. From the user's perspective, it is not possible to check all contents until loading is complete. Therefore, the speed index is useful to compute the web page loading time to the user. In WebPageTest, two methods are used to compute the speed index as follows.

1) Calculate the rendered area based on pixel variation
2) Calculate the rendered area according to a triggered event

The method for calculating the rendered area computes the amount of pixel change until the webpage loading is finished. When the video frame changes, we can know the event by finding the pixel difference value. The event-based area calculation method collects visual changes for each frame. In mobile apps, we have to compute the implicit loading complete-time event.

Mun [6] has conducted a study on mobile application performance testing using speed index . In addition to the speed index, the authors include XML log, packet trace, and CPU usage for the detailed performance metrics.

## III. A Robot-based Mobile App Speed Measurement System

We propose a method to measure app speed using a robot arm that tests automatically in Fig. 1. A robot-based mobile app speed measurement system consists of four modules: test manager with a web server, UI analyzer, robot controller, and speed index analyzer. Once we start the test by test manager, UI analyzer records application screens. The coordinate-send-program sends click-coordinates to test manager, then robot arm clicks mobile screenshot. The test manager will collect test output data in mp4 files which records mobile screens and the point time in CSV files. From mp4 files, we analyze speed index with a speed index analyzer.

### A. *Testing UI events with a robot*

**Test manager** controls modules and set up the experiment configuration. It initializes the number of click events and test application names with the test manager. After setting up the experiment environment, the test manager sends commands. UI analyzer recording the application screens and sending coordinate data to the test manager. The test manager controls the robot arm through a function to the robot controller.

**Robot Controller** drives a robot arm to touch a mobile application screen. after receiving the test and send application coordinates, robot controller moves the robot arm by converting it into coordinates that robot arm can move. We implement a robot arm with an open-source project, called Tapster[6], to control a robot arm for touch events. Tapster controls the robot arm with servo motors connected to Arduino.

To control the robot arm through the test manager, we transmit an HTTP message including the coordinates that the robot should click. In Fig. 2, the robot controller receives coordinates that match the resolution of the mobile phone through coordinates-send-program and converts them into coordinates that the robot arm can click. We puts the converted coordinates and the name of the application, which we want to experiment, in the HTTP message, and send it to the test manager. In the HTTP message, the key is the coordinate that is the data information to be transmitted, and the value is the app name and coordinate values were converted to JSON format to create an HTTP POST message. The test manager calls a function to move the robot arm. At this time, the robot controller moves the robot arm with the received coordinates.

### B. *UI Analyzer*

While we are testing mobile applications, we need to calculate where to generate user behavior through image analysis. For this purposes, the UI analyzer is in charge of application experiment screen transmission, recording, and video analysis. The screen of the mobile device is input to the UI analyzer through Miracast and capture board, and the UI analyzer finds clickable elements through image processing and deep learning.

Next, the web server records the test video file. A robot arm moves its touching device to the point on the smartphone display. For this purpose, the UI Analyzer send the coordinate information of the clickable component of a mobile app display to the test manager. Given a mobile app display, we can perform the touch operation on the mobile app display randomly or intelligently. Without any knowledge of the mobile app, we can generate random user input events until we meet the hit event. However, under the random test, it will take a long time to complete a test and it does not reflect the average user behavior. Therefore, we devise an intelligent way of determining clickable elements on the mobile app display.

[6]https://github.com/tapsterbot/tapsterbot

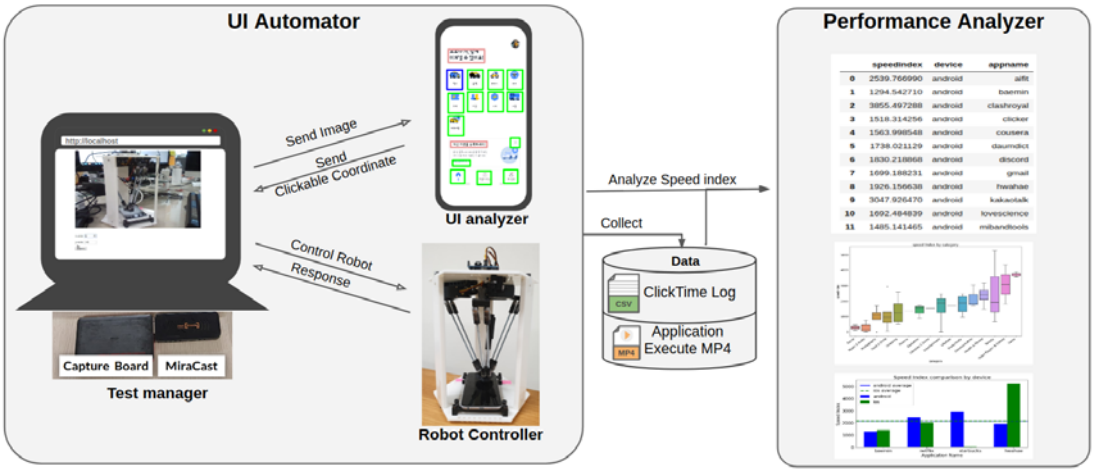


Fig. 1. A robot-based mobile app speed measurement system

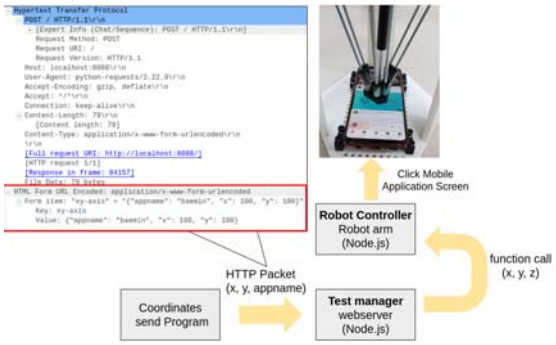


Fig. 2. The process of robot operation

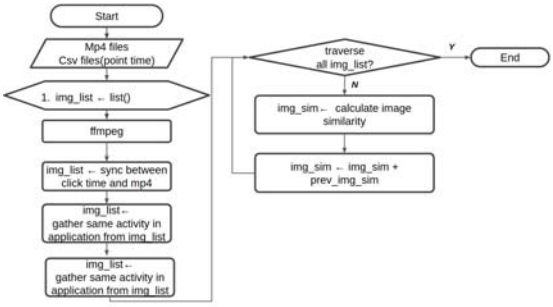


Fig. 3. The flow chart of speed index calculate

For this purpose, we use a deep learning-based CNN image recognition method. We have collected the training data from Android's ADB tool that extracts the layout of a mobile app display from XML dump files corresponding to the application activity.

We store the coordinates for click elements in CSV files. For training data, we extracted clickable elements in the format of the image. It is observed that icons in the mobile app display become clickable components. We have extracted 9,832 touchable images and 35,202 untouchable images. Then, we adjusted the image size to 39 x 39 resolution for the fast training.

### C. *Mobile app speed index analyzer*

Mobile app speed index analyzer calculates the speed index of the application to measure mobile app performance. We used a method in Fig. 3 to measure the speed index of a mobile application using mp4. We used only the captured video to calculate the speed index in Fig. 1. From the video, we detect the scene change by comparing each frame according to the similarity metric.

We record the application execution video with ffmpeg and convert it to a jpg file. Besides, we save the time when a robot arm hits the target in the CSV file. As there is no explicit loading event notification in the mobile app, we find a load complete event by comparing images based on the similarity computation function. We estimate the split point in the forward direction of image progress. We synchronize clocks between the robot arm and the analyzer with Sync function.

## IV. Performance Evaluation

### A. *Experiment environment*

TABLE I
THE NUMBER OF MOBILE APPS BY CATEGORY

| Category | Count | Category | Count |
|---|---|---|---|
| Beauty | 3 | Lifestyle | 1 |
| Communication | 3 | Music & Audio | 3 |
| Education | 3 | Photography | 4 |
| Entertainment | 4 | Productivity | 6 |
| Food & Drink | 5 | Shopping | 6 |
| Game | 2 | Social | 2 |
| Health & Fitness | 4 | Video Players & Editors | 2 |
| Libraries & Demo | 1 | Finance | 1 |
| | | Total | 50 |

Figure 4 shows the experiment environment. We install a video capture board and a Miracast device that streams the screen of a mobile phone. We capture the video from the

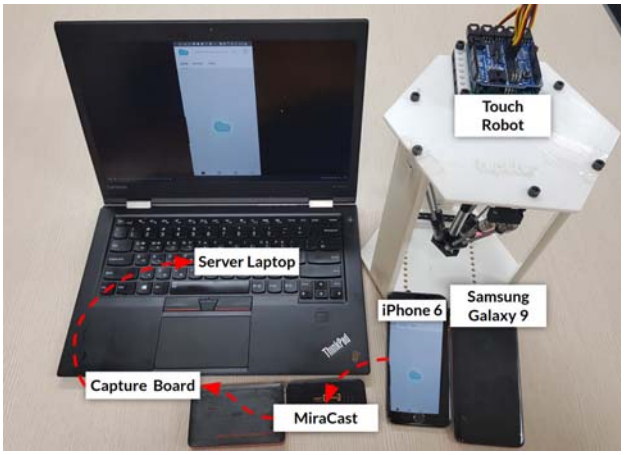


Fig. 4. Experiment Environment

Miracast device at the server with Python. We use an iPhone in iOS 6 and a Samsung S9 in Android 10. We have tested 18 Android applications and 32 iOS applications of 22 categories in the table I.

### B. Results

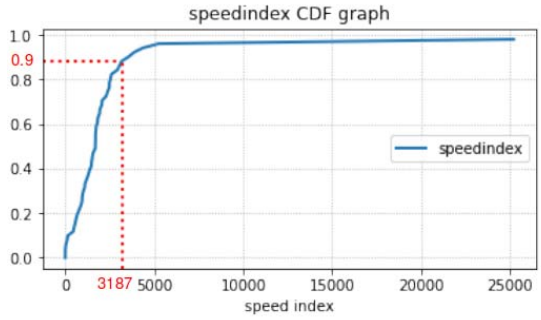


Fig. 5. The speed index by application.

**Accuracy of Image Similarity:** We have verified the accuracy of the image similarity of 100 image samples. It was observed that the accuracy of SSIM is 0.78 and correlate2d is 0.80. From the experiment of finding the load complete events, we compute the speed index. For all the clickable events, we can test 40% of clickable events.

**Speed index by category:** We performed speed index analysis by category in Fig. 6. The category of the slow speed index is game, and the fast speed index is social network services including Instagram and Naver Band. Game apps usually load a large volume of update data at the beginning, which causes slow performance. However, when moving the screen with a click of a button, the speed index of game mobile apps is faster than other applications. In the category of beauty, it is observed that the slowest app was 5000 and the fastest app fell below 1000.

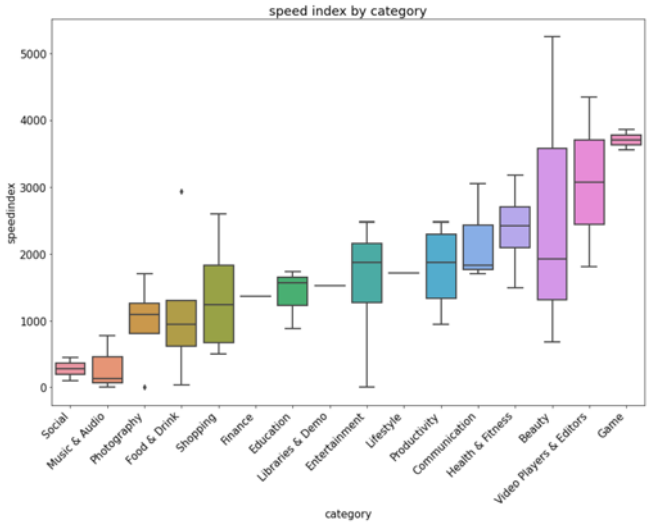


Fig. 6. Speed index by category

## V. Conclusion

In this study, we have presented a robot-based mobile app speed measurement system. After implementing a robot arm that can hit the mobile app display, we have performed iOS and Android mobile app speed measurement experiments. For the metric, we have adopted the speed index used in web page testing. From experiments, we can find that the same mobile apps show different speed index according to operating systems. In the future, we plan to perform comprehensive mobile app speed measurement experiments.

## Acknowledgment

This work was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT)(No.2019-0-01343, Training Key Talents in Industrial Convergence Security). Corresponding Author is Youngseok Lee.

## References

[1] Ke Mao, Mark Harman, and Yue Jia. Robotic testing of mobile apps for truly black-box automation. *IEEE Software*, 34(2):11–16, 2017.

[2] Mihai Craciunescu, Stefan Mocanu, Cristian Dobre, and Radu Dobrescu. Robot based automated testing procedure dedicated to mobile devices. In *2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 1–4. IEEE, 2018.

[3] Jih-Gau Juang and I-Hua Cheng. Application of character recognition to robot control on smartphone test system. *Advances in Mechanical Engineering*, 9(3):1687814017693181, 2017.

[4] Ariel Rosenfeld, Odaya Kardashov, and Orel Zang. Automation of android applications functional testing using machine learning activities classification. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*, pages 122–132, 2018.

[5] Xusheng Xiao, Xiaoyin Wang, Zhihao Cao, Hanlin Wang, and Peng Gao. Iconintent: automatic identification of sensitive ui widgets based on icon classification for android apps. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 257–268. IEEE, 2019.

[6] Hyunsu Mun and Youngseok Lee. Appspeedxray: A mobile application performance measurement tool. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 1010–1012, New York, NY, USA, 2020. Association for Computing Machinery.