Adding Multi-Touch Gesture Interaction in Mobile Web Applications

Shah Rukh Humayoun[1], Franca-Alexandra Rupprecht[1], Steffen Hess[2], and Achim Ebert[1]

[1] Computer Graphics and HCI Group
University of Kaiserslautern
Gottlieb-Daimler-Str., 67663, Kaiserslautern, Germany
{humayoun,ebert}@cs.uni-kl.de, frupprec@rhrk.uni-kl.de
[2] Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
steffen.hess@iese.fraunhofer.de

Abstract. This paper describes the MTGest framework, an open library for adding multi-touch gesture interaction to HTML-based mobile web applications. MTGest was used in a comparative study to evaluate the multi-touch gesture interaction in a mobile web application in comparison to a native iOS mobile application. The results indicates that in most cases the web based gestures efficiency is either approximately the same or higher than the iOS-based app. The study was carried out as an initial experiment using isolated gestures, targeting the iOS platform only. For generalizing the results there is a need to perform detailed user evaluation studies with different platforms and for more complex interaction scenarios.

Keywords: Smart Devices, Smartphones, Tablets, Mobile Apps, Web Apps, Multi-Touch Gesture, Interaction Design, Mobile Environments.

1 Introduction and Related Work

Due to the popularity of smart devices and mobile applications (also called mobile apps), companies are offering more and more their product support in them. In order to reach to a broad pool of potential users, companies need to develop their applications for many of the existing mobile platforms (e.g., Google Android [2], Apple iOS [1], Microsoft Windows Phone [5]). Developing mobile apps separately for each platform is costly and time consuming while keeping focus on just one platform reduces the number of accessible users. We found in our previous study [3] that developing mobile apps through cross-platform development frameworks (where the application is developed once and deployed on the possible target platforms) still lacks in many aspects such as: few platforms support, only partially support of the targeted platform's interaction schema, far behind the native development environments.

Mobile web applications (also called *mobile web apps*) are based on web technologies like HTML, CSS, and JavaScript. HTML5[1] as an upcoming standard by W3C[2] provides the possibility of offline browsing as well as accessing many device resources (such as localization services or sensors); hence, HTML5-based web apps can be used as an alternative to the native mobile apps in many cases. There are many benefits of this approach such as: requires less efforts and resources for developing, supports all platforms, provides a consistent user experience and interaction concept across all platforms. However, HTML5 lacks built-in tags for the functionality of current multi-touch gesture interaction paradigm, which reduces the web apps' applicability compared to the native mobile apps. HTML5 provides a set of interfaces for the basic touch events but does not have built-in functions to support directly most of the current multi-touch gestures. In this work, we focus on adding the current multi-touch gesture interaction paradigm support in HTML-based documents in order to provide the current multi-touch gestures (e.g., *double tap, swipe, flick, zooming, rotation*) in mobile web apps.

We provide this support through our developed library, called **MTGest** (**M**ulti-**T**ouch **Gest**ures) library. This library enables mobile web apps the provision of multi-touch gesture interaction inside them in order to give the expected user experience and interaction concept of the current mobile paradigm across all platforms. For checking the efficiency and user satisfaction with the provided multi-touch interaction support in mobile web apps by our MTGest library, we conducted a user evaluation study. In this conducted study, users from different backgrounds and expertise tried two simple apps, i.e., one mobile web app based on MTGest library and the other one a iOS-based mobile app based on iOS native gesture support. Users tried each gesture on both apps one-by-one and gave their feedback using a questionnaire form. Results show the same level of efficiency and user satisfaction in many cases, as well as better in few cases and lower in some other cases. Overall, results indicate that mobile web apps through MTGest kinds of libraries can be an alternative solution in the future.

Some other frameworks for the support of multi-touch gestures are: jQMultiTouch [7] web tool-kit, inspired by JQuery, for creating multi-touch interfaces; Gesture Coder [6] for generating code to recognize multi-touch gestures. However, we chose JQuery[3] and hammer.js[4] as foundations due to their powerful abstraction from low-level implementation details and their cross-browser compatibility. One way of working with the MTGest library has already been described in detail in [4], whereas this paper focuses on the study comparing MTGest with native gestures.

The remainder of the paper is structured as follows: In Section 2, we introduce our MTGest library. In Section 3, we provide details of the conducted user evaluation study. In Section 4, we present and discuss the results. Finally, we conclude in Section 5.

1 W3C - HTML5. `http://www.w3.org/TR/html5/`

2 World Wide Web Consortium. `http://www.w3.org/`

3 The jQuery Foundation - `http://jquery.com/`

4 Eight Media - `http://eightmedia.github.com/hammer.js/`

2 The MTGest Library

HTML5, the new specification of HTML by W3C, is still in working-in-progress process. HTML5 provides a set of interfaces[5] for touch events. However, it lacks built-in tags for the functionality of multi-touch gestures.

Our **MTGest** (**M**ulti-**T**ouch **Gest**ures Library) library, based on JavaScript and JQuery, enables the support of multi-touch gesture interaction in HTML5-based documents. It is built on top of the hammer.js library, which is also based on JavaScript, for controlling gestures on touch devices. It supports most of the single and multi-touch gestures in the current mobile domain. Moreover, it is possible to define own gestures in which the developer specifies the criteria for such a gesture, e.g., tapping three fingers together.

The standard gestures supported by our library are: *tap*, *double tap*, *hold*, *drag*, *swipe*, *transform* (*pinch*), *rotation*, *flick*, *zoom and rotation* together, and *shake*. Additional customized gestures (e.g., *three-fingers tapping* or *multi-fingers swiping*) are also provided for using in some specific interaction context.

The MTGest library works as follows. The provided functions, corresponding to each gesture, by the library are attached to a container representing a specific area in the HTML document. The *hammer.js* is also attached to the same container to get the touch events happen to this container. It is possible to attach more than one gesture to the same container. Then the specific area in the HTML document, representing the container, gives the interaction according to the attached gestures. Figure 1 provides the overall architecture.

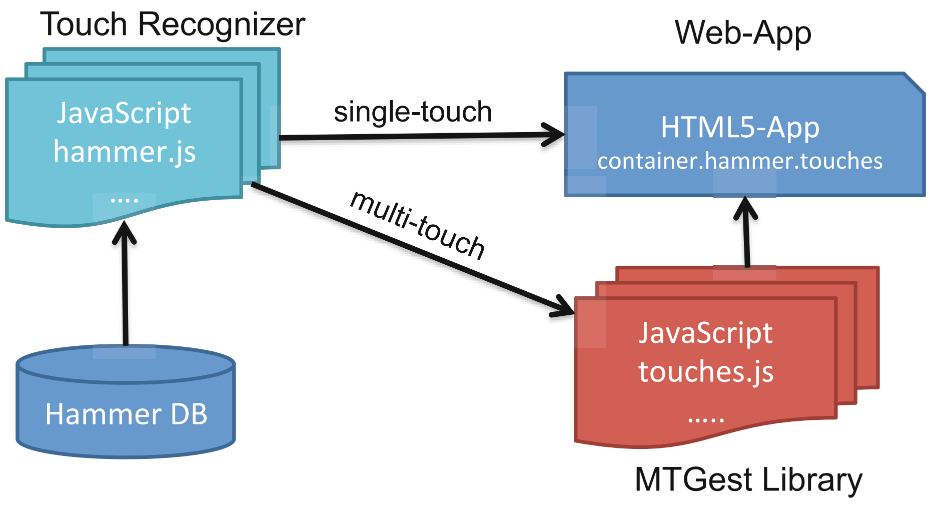


Fig. 1. The overall architecture of the working of MTGest library

[5] https://dvcs.w3.org/hg/webevents/raw-file/tip/touchevents.html

3 The User Evaluation Study

We performed a user evaluation study in a controlled environment, where the focus was on comparing the multi-touch gesture interaction support provided through our MTGest library and a native mobile platform. This was done through developing two simple mobile apps, one was a mobile web app that provides the desired multi-touch gestures through our MTGest library while the other one was a native iOS-based mobile app that provided these gestures' support through the native iOS library.

In the following, we provide details of the both developed apps (i.e., the mobile web app and the iOS-based mobile app), the study goal and hypothesis, and the experiment settings.

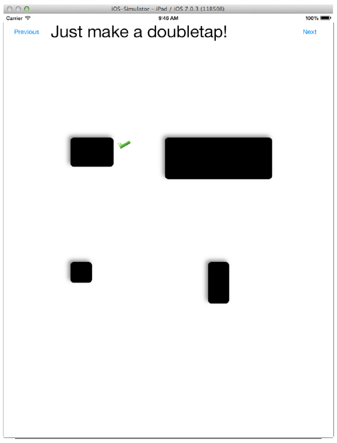
3.1 The Testing Apps

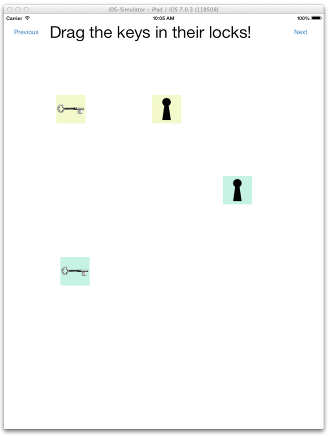
For testing the multi-touch gesture interaction support through our MTGest library and a native mobile platform library, we developed two simple apps. The first one was a mobile web app that used our MTGest library for providing the multi-touch gesture interaction support, while the second one was an Apple iOS-based native mobile app that provided the multi-touch gesture interaction support using the iOS native gestures support. Both apps provided the same level of functionality and there was no difference in the interface style or layout. This was done in order to avoid any biasedness in the user evaluation study.

Eight touch- and multi-touch gestures, mostly the standard ones provided by most of the current platforms, were implemented in both apps. These gestures include: *tap, double tap, hold, drag, swipe, flick, zoom* (both *zoom-in* and *zoom-out*), and *rotation*. In both apps, each gesture was covered up on one page where each page contained several (up to four) containers having the implementation of the underlying gesture. These containers were different in size and orientation with the same gesture support in order to provide a variety of user interaction with the underlying gesture. When a user interacts correctly with the container through the specified gesture, a feedback is shown to the user for this correct interaction; otherwise nothing is shown. The user can go to the next page for interacting with the next gesture any time or after finishing the interaction with all containers on the current page.

In the cases of *tap, double tap,* and *hold* gestures, we implemented four containers on each page having the underlying gesture support. Figure 2 (a) shows the screenshot of the native iOS-based app where the four containers have the *double tap* gesture interaction. The green correct mark indicates that the user has successfully interacted with this container with the *double tap* gesture. In the case of *drag* gesture, there were two sets of containers. One set was showing a key while the other one was showing the lock, as shown in Figure 2 (b). The container set showing the key shape were linked with the *drag* gesture. When a use drags this key container to a lock container, the app indicates a successful execution of the gesture.

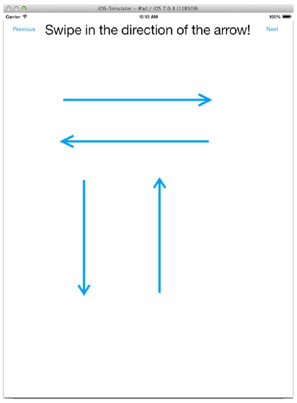
In the case of *swipe* gesture, both apps provided four containers to provide the interaction support in four directions (i.e., left-to-right, right-to-left, up-to-down, and down-to-up), as shown in Figure 3 (a). The user needs to swipe the finger from the tail to the head of the arrow in order to execute the *swipe* gesture interaction correctly. In the case of *flick* gesture (here the *flick* gesture represents same as its representation in iOS), both apps provided one container that had the interaction of *flick* gesture in four directions same as with *swipe* gesture case, as shown in Figure 3 (b). Finally, the *zoom* and *rotate* gestures were implemented through two containers in both apps. In the case of *zoom* gesture, both containers were different in size and it was up to users to play with them for checking zoom-in and zoom-out interaction. While in the case of *rotate* gesture, two figures were given in the containers up-side-down orientation so that users can rotate them in normal orientation.

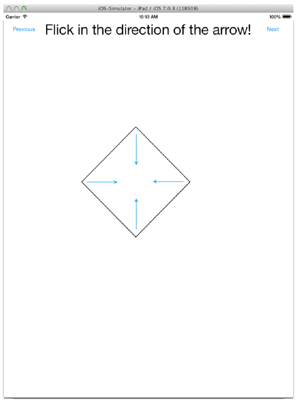




(a) (b)

Fig. 2. (a) A screen-shot of the page with the *double tap* gesture support, *(b)* A screen-shot of the page with the *drag* gesture support





(a) (b)

Fig. 3. (a) Four directed arrows show the *swipe* gesture interaction in the same direction, *(b)* the arrows inside the container represent the *flick* gesture interaction in the same direction

3.2 Study Goal and Hypothesis

The goal of this user study was to analyze whether our developed MTGest library can provide the touch- and multi-touch gestures interaction support in mobile web apps compared to native mobile apps (here we target only the Apple iOS platform) from the perspective of efficiency of such interaction support and user satisfaction level with the underlying interaction. We compare the results from the following criteria:

- *Efficiency:* We check whether the underlying gesture worked accurately and the interaction-response time was appropriate. In this regard, we collect subjects' feedback for both apps and compare them.
- *User Satisfaction:* We collect subjects' feedback for both apps and compare them.

Our hypothesis is that in term of efficiency and user satisfaction, our proposed MTGest library provides approximately the same interaction support for the underlying gestures compared to the native platform (i.e., the iOS platform) support.

3.3 The Experiment Settings

We performed the evaluation study with 12 subjects (3 females and 9 males). We categorized them according to their experience with smart-devices and mobile platforms. Four subjects were experienced users of Apple iOS platform, three subjects were experienced of Android platform, while the remaining 6 subjects were without much expertise in any specific mobile platform. The age of subjects were between 20 and 36 years old with a mean of 27.5.

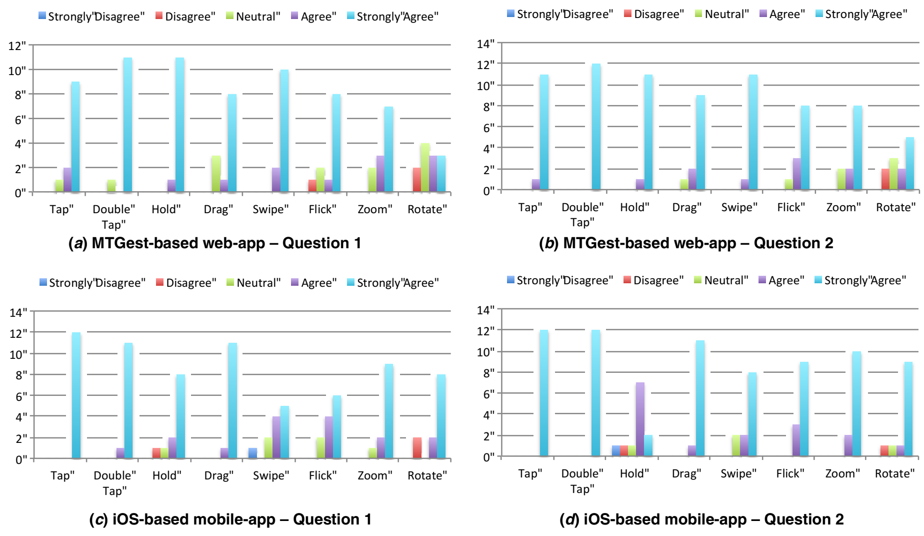
The test devices for both developed apps (i.e., the mobile web app based on our MTGest library and the iOS-based native mobile app) were Apple iPad 2 with the same specifications. We installed the web app on one device while the native app on the other device. Before start of the experiment, a brief tutorial was given to each subject about the goal of the experiment. For each tested gesture, subjects were asked first to try all the containers having the underlying gesture support on both devices. Then they were asked to fill a closed-ended questionnaire form with a likert scale from 1 to 5, where 1 meant *strongly disagree* and 5 meant *strongly agree*. There were total four questions in this mode, same for both apps separately. The aim of first two questions was to get the subjects' feedback for checking the efficiency of the underlying library (i.e., the MTGest library or the native iOS library) in providing the support of the tested gesture interaction. The aim of the later two questions was to get the subjects' feedback for checking their satisfaction level with the tested gesture for both apps. These four questions were:

1. *The gesture works accurately.*
2. *The interaction-response time of the gesture was appropriate.*
3. *Overall, I am satisfied with this gesture facility through the underlying app (i.e., the web app or the iOS-based app).*
4. *In future, I would like to use this gesture through the underlying app (i.e., the web app or the iOS-based app.*

At the end of closed-end questions for each gesture, subjects were also asked that which mode (i.e., the web app or the native app) for this gesture is preferable by them for the future usage, with the option of selecting one or both apps. In order to avoid any biasedness towards the second testing app due to the learning effects, half of the subjects were asked to test the web app first and then the iOS-based native app, while the other half were asked to test the iOS-based app first and then the web app.

4 Results and Discussions

In this section, we provide the results of our conducted user evaluation study and discuss them to check whether they reflect our initial hypothesis. After testing each gesture on both apps, subjects were asked to answer the set of questions regarding the tested gesture.



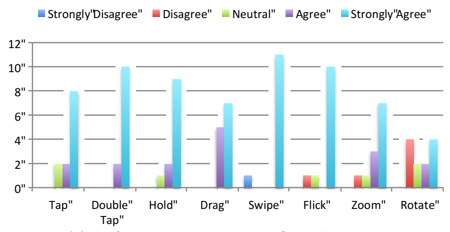
(*a*) MTGest-based web-app – Question 1
(*b*) MTGest-based web-app – Question 2
(*c*) iOS-based mobile-app – Question 1
(*d*) iOS-based mobile-app – Question 2

Fig. 4. The subjects' feedback for questions 1 and 2, collected through the likert-scale

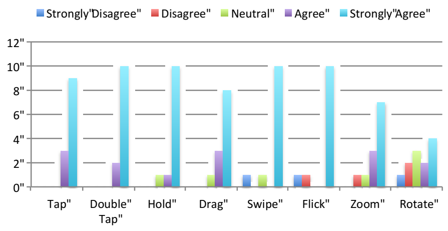
Figure 4 provides the subjects' feedback with regard to the first two questions for each of the tested gesture on both apps. For the *tap* gesture, all subjects strongly agreed for the accurately work of iOS-based app, while 9 subjects strongly agreed and 2 subjects just agreed for the web app. Regarding the second question, all subjects strongly agreed for both apps except one that rated agreed for the web app. Results for the *double* tap gestures are also nearly the same in both cases for both apps. This indicates that our MTGest library provides the same level of efficiency for these two gestures. The case of *hold* gesture is interested, as the subjects' feedback for the web app is far better than the iOS-based app. We observed that iOS gives a too quick interaction response, which the subjects might not expected from the *hold* gesture as they were expecting a little wait for keeping hold the touch. That might be the reason for this lower ranking by subjects. We also observed that subjects from the Android platform or non-experienced background were more reluctant in liking the iOS-based app response, while they felt happy with the web app response. In the case of *drag* gesture, subjects' feedback was a bit better for the iOS-based app compared to the web app. However, the difference was not very noticeable.

In the case of *swipe* gesture, subjects' feedback about the web app was much higher than the iOS-based app. We observed that again this is because the too quick response in iOS, as even if the subject swiped just little more than half of the swipe area length it started working. In the case of web app, it worked only when subjects swiped the whole length of the swipe area. Due to this, subjects felt more confident in web app compared to the iOS-based app. This is also indicated in the case of *flick* gesture, where subjects' feedback about the web app was slightly better than the iOS-based app. In the case of *zoom* gesture, subjects rated iOS-based app better than the web app. However, again the difference is not much significant. Finally, in the case of *rotate* gesture, subjects rated quite higher the iOS-based app compared to the web app. We observed that this is because of the image drawing performance issue in the web app, as the image is drawn on the page each time the user moves the fingers for the rotation.

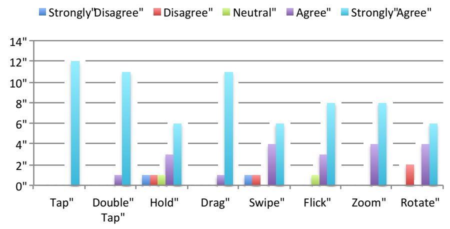
Overall, the subjects' feedback of the first two questions indicates that in most cases the web app efficiency is either approximately same or higher than the iOS-based app. While in some complex gestures such as zooming and rotation, it is behind the iOS-based app. However, this can be improved in future. In summary, we can say that the overall feedback of these two questions confirm our hypothesis regarding the efficiency of our developed MTGest library.



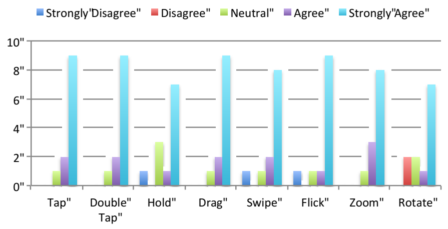
(*a*) MTGest-based web-app – Question 3



(*b*) MTGest-based web-app – Question 4



(*c*) iOS-based mobile-app – Question 3



(*d*) iOS-based mobile-app – Question 4

Fig. 5. The subjects' feedback for questions 3 and 4, collected through the likert-scale

Figure 5 provides the subjects' feedback with regard to the later two questions for each of the tested gesture on both apps. The aim of these two questions was to check the user satisfaction level with the tested gesture interaction. For the *tap* gesture, the subjects' feedback with the iOS-based app was a bit higher than the web app. Moreover, 11 subjects preferred for using this gesture through iOS-based app and 9 subjects also went for web app too. It is noted that in the case of future usage of the tested gesture, subjects were free to choose one or both. In the case of *double tap* gestures, subjects' feedback was approximately the same. Also, 11 subjects choose the iOS-based app while 10 mentioned the web app for the future usage of this gesture. We observed that subjects' feedback improved towards positive with the web app after getting experience. In the case of *hold* gesture, the subjects' feedback reflects the feedback of question 1 and 2, as their satisfaction trend for the web app was quite higher than the iOS-based app. They significantly also preferred the web app for the future usage of this gesture (12 compared to 4). In the case of *drag* gesture, the subjects' feedback was a bit higher for the iOS-based app compared to the web app. However, 10 subjects choose web app while 9 subjects choose iOS-based app for the future usage of this gesture.

In the case of *swipe* gesture, subjects' feedback about the web app was much higher than the iOS-based app. Moreover, 10 subjects preferred web app and 5 preferred iOS for the future usage of this gesture. We observed that subjects' feedback for the web app was increased because the web app provided better the expected interaction (i.e., working when user swipes through the whole area rather than just a part of it) in this gesture implementation. The same also went for the *flick* gesture, where subjects' feedback again was more towards the web app. However in this case, the subjects' preference for the future usage of this gesture was nearly the same for both the web app and the iOS-based app, i.e., 8 and 7. In the case of *zoom* gesture, subjects were a bit higher satisfied with the iOS-based app than the web app. However, the difference is unnoticeable. Finally, in the case of *rotate* gesture, subjects significantly rated higher the iOS-based app compared to the web app. Also, only 2 subjects preferred the usage of this gesture in web app compared to 9 for the iOS-based app.

Overall, the subjects' feedback of the later two questions reflects their experience with the tested gesture on both apps and approximately the same as of the previous two questions. Except in the cases of *drag* or *rotate* gestures, the subjects' feedback about the later two closed-ended questions for the web app was either approximately the same as for iOS-based app or higher than it. However, the subjects' feedback regarding their satisfaction level has many limitations. There are many factors (e.g., users' expectations, curiosity, their interests in new experiences, their expertise with gestures, their positive attitude towards Apple, their low expertise with MTGest library, etc.) that can affect users' satisfaction level. In spite of this, results of the study provide an indication that the web apps have the potential of providing an alternative to the native mobile apps if they get support by multi-touch gestures libraries. MTGest library is one of the candidate libraries; however, it needs to be improved for providing better performance in the cases of some complex gestures (e.g., *zoom* and *rotation*). Moreover, as the study targeted only the iOS platform and the tested scenarios were quite simple; hence, for generalizing the results there is a need to perform detailed user evaluation studies with different platforms and with more complex interaction scenarios.

5 Conclusion

In this paper, we showed that the MTGest library is feasible to be used for multi-touch gestures with regard to efficiency and user satisfaction. We performed an initial evaluation study with 12 subjects comparing the MTGest library with the native iOS gestures. A main issue of the study was to compare single multi-touch interaction in an isolated scenario.

The results showed, that MTGest works accurately for *tap*, *double tap*, *hold*, *drag*, *swipe*, *flick* without showing a significant difference to the native iOS gestures. Indeed, *swipe* and *flick* gestures were rated significantly better. The rotation gesture worked significantly more accurate on the native iOS implementation. In general, the study indicates that in most cases the web app efficiency is either approximately same or higher than the iOS-based app.

With regard to user satisfaction, the results of the study indicated not a clear result. The subjects' satisfaction level reflects their experience with the tested gesture on both apps. In general, there is no significant difference between web based gestures and native gestures. This leads us to the conclusion, that using MTGest is reasonable although it needs to be improved for providing better performance in the cases of some of the more complex gestures.

Future work will deal with the performance of a detailed user evaluation considering the usage of gestures within a concrete scenario and also do a comparison with other mobile operating systems, such as Google Android or Microsoft Windows Phone.

References

1. Apple iOS, `http://www.apple.com/uk/ios/`
2. Google Android, `http://www.android.com/`
3. Humayoun, S.R., Ehrhart, S., Ebert, A.: Developing Mobile Apps Using Cross-Platform Frameworks: A Case Study. In: Kurosu, M. (ed.) HCII/HCI 2013, Part I. LNCS, vol. 8004, pp. 371–380. Springer, Heidelberg (2013)
4. Humayoun, S.R., Hess, S., Kiefer, F., Ebert, A.: i2ME: A framework for building interactive mockups. In: MobileHCI 2013, pp. 606–611. ACM, New York (2013)
5. Microsoft Windows Phone, `http://www.windowsphone.com/`
6. Lü, H., Li, Y.: Gesture coder: A tool for programming multi-touch gestures by demonstration. In: CHI 2012, pp. 2875–2884. ACM, New York (2012)
7. Nebeling, M., Norrie, M.: 2012: jQMultiTouch: Lightweight toolkit and development framework for multi-touch/multi-device web interfaces. In: EICS 2012, pp. 61–70. ACM, New York (2012)