Touchless Interaction Using Magnetometer for Mobile Games

Jason Wihardja
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia

Raymond Kosala
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia

***Abstract* — This paper proposes an alternative method for conducting touchless interaction. Using a magnetometer sensor, we attempt to map some common gestures into something that can be understood by a magnetometer sensor. To demonstrate the concept, an application and a punching game that utilize this principle were developed. Using the application, users are able to perform a common swiping gesture using a magnetometer sensor. When the game detects a punch gesture, it will respond accordingly. The application was tested and compared against an existing touchless interaction technology. The test results showed that there are some aspects where the magnetometer solution is better compared to the existing solution. This suggests that touchless interaction using magnetometers might have a promising future.**

Keywords—Touchless Interaction; Magnetometer; Mobile Games; Human-Computer Interaction

I. INTRODUCTION

Mobile phones today are equipped with many types of sensors [1]. Starting from the touch screen technology that enables users to interact and control their smartphone by touching the screen, orientation sensors are usually used in mobile games for the users to interact with the application by tilting the device. Magnet sensors are usually used within a compass application, along with temperature sensors and increasingly, some mobile phones are being equipped with fingerprint sensors.

The sensors that have been commonly used to facilitate human-computer interaction (HCI) are orientation sensors. However, despite the fact that mobile devices are equipped with many different types of sensors, people still mainly interact with their mobile phone by touching the screen. Many games use orientation sensor to control objects in the game. However, it is only used by a small number of mobile games.

In regards to touchless interaction, not many technologies have been implemented. One of the techniques that is currently used to achieve this is through optical tracking. Some smartphone manufacturers have tried to implement a touchless way of interacting with smartphones. For example, they have included a feature called air gesture [2] or combination of air and touch [3]. By performing some hand motion in front of the phone, users can perform some basic tasks without actually touching the phone, such as answering a phone call, flipping across images in the gallery, scrolling a web page in the internet browser, and some other tasks. However, looking at the whole smartphone market for comparison, this type of interaction is still not widely used. The reason is because it uses the phone's front camera to detect motion. This technique is not very efficient in terms of CPU usage and power consumption, which drains the phone's battery much faster [4].

With that in mind, this paper aims to explore an opportunity to provide an alternative way to accomplish touchless interaction. The use of a magnetometer sensor for touchless interaction seems to be overlooked, even until today. Knowing the fact that most modern smartphones are equipped with a magnet sensor, a user can then expose a magnetic object to the smartphone and change the magnetic field around the device. By monitoring the changes in the magnetic field, a specially built application can track the location of the magnetic object and interpret the changes to provide a new way to achieve touchless interaction.

Although its main usage is to provide metrics for compass and navigation applications, it can also be used as a way to interact with mobile devices. Since the magnetometer is able to detect changes in magnetic field, then any magnets that interfere with the magnetic field around the device can be detected and translated into actions inside an application.

In this paper, an Android phone that is equipped with a magnetometer was used as a testbed. An application was developed to demonstrate and utilize the magnetometer as an interaction media between the user and the smartphone. The application that was implemented only works if and only if the smartphone is equipped with a magnetometer. In addition, the user will also need to wear a magnet on his/her hand to interact with the application.

This paper proposes an application that serves as a proof of concept for a new way of how people interact with their smartphones. By utilizing the magnetometer inside a smartphone, we can show that it is actually possible to have interactions without actually touching the phone. As of the moment, the interactions that are accepted are left swipe, right swipe, and selection or punching gestures. The experiments

suggest that this approach bring several benefits over the currently existing solution in touchless interaction. Firstly, it consumes far less energy compared to using the smartphone's front camera to detect motion. Secondly, the algorithm used to perform such sensor reading is also lighter in terms of CPU usage.

This study is expected to encourage other people or researchers who are interested in the same area to implement or even extend the scope of the research of this kind of interaction method and create better and more interesting applications that utilize the same magnet principle.

There is a limitation for this concept. The magnetometer can only detect the magnetic field around the device. Therefore, it is impossible to use multiple magnets. In the case where multiple magnets are used, then the changes that the magnetometer will detect is the result of changes to the magnetic field caused by the multiple magnets.

The rest of this paper is organized as follows. In the next section, the relevant background and technologies related to magnetometers and magnet sensors will be explained. Next, the design of the game and the application are presented. Then the experiment's design, results and evaluation are presented. Finally, the findings and future research directions are discussed.

II. Background

A. Magnetic Field and Magnetometers

A magnet is an object that has two special properties. First, it must be made of some specific materials, such as iron. Next, it has two poles, namely 'north pole' and 'south pole'. Interestingly, when other magnetic objects are placed within the object's magnetic field, they will attract each other or repel each other, depending on the polarity. North poles attract south poles and repel north poles, and vice versa.

Since the early days of electronic devices, magnets have been noted for negative effects on those devices, including mobile telephones. However, that is no longer the case for today's smartphone. Several components that were said to be experiencing negative effects have been replaced by new technologies. According to data provided by Supermagnete [5], they found no danger detected from placing magnets near smartphones.

A magnetic field is the space around a magnetic object that is affected by the magnetic force coming out from the magnetic object. It has two important components, namely flux density and polarity. By convention, it is said that this magnetic force emanates from the north pole of a magnet and is absorbed into the south pole of the magnet.

Magnetometer is a device that can measure the strength of magnetic fields. Typically, a magnetometer is used in a device that can assist people when searching for objects underground or underwater and are commonly used in the airport to check for the presence of metallic weapons. Recently, it can also be found inside many smartphones to provide navigation data [6].

B. Magnet Sensors in Smartphones

Two of the many types of magnetometer sensor that are typically used in today's smartphone, are a 'Hall effect sensor' and 'anisotropic magneto resistive' (AMR) sensor [7].

The most common magnetometer method is the Hall effect method. The Hall effect sensor primarily relies on a semiconductor material. When this material is placed within a magnetic field, the flux deflects the electron from its original straight movement. As these electrons move sideward, it will create a potential difference between two sides of the semiconductor material. This difference is then detected and returned as an output voltage called Hall voltage [7].

An AMR sensor works based on the magnetoresistant effect on a specific material which causes the material to change the value of its electrical resistance when placed within a magnetic field [7].

A case study on the comparison of these two types of magnetic sensor was published in [8]. In this case study, it was discovered that the measurement result differs significantly. It was also found that the Hall effect sensor is less sensitive compared to the AMR sensor. In addition, the Hall effect sensor can only operate in the near field of the magnet while AMR sensor can operate over a larger distance.

C. Related Work

At the time of writing, there was not much research in this area. As has been stated earlier, magnets have not become standard tools used in human-computer interaction. A related work, however, has been done in this area. Bianchi and Oakley

presented a similar work that utilizes the same magnetism principle. By creating some specially built magnetic objects, they were able to demonstrate their ability to detect some forms of interactions, such as token identification, orientation interaction, linear movement of tokens, etc.

III. Application And Game Design

A. Application Overview

To demonstrate the concept of touchless interaction using a magnetometer sensor, an application was implemented in the form of a game which works primarily based on the principle of the ability of magnet sensors in sensing the presence and position of a nearby magnet.

The application consists of three main parts, namely magnetometer debugging information, human-computer interaction demonstration, and the game itself. We believe that by implementing a game, the concept can be properly demonstrated in terms of how well the concept works and how fun and intuitive it is from the perspective of the user.

In magnetometer debugging information, !, " , and # values that are obtained from the magnetometer sensor are displayed along the length of the resultant vector that is calculated from those three values. The results are calculated using the Euclidean norm equation as shown in (1).

$$\|!\| := \sqrt{!_1{}^2 + \cdots + !_n{}^2} \quad (1)$$

In the human- computer interaction demonstration, the user will be presented with a gallery-like user interface. The user can swipe through four images either using their finger or magnetometer sensor. There is a switch at the top of the screen to change between input modes.

In the game part of the application, the user can select an image from their gallery or Facebook. If the user has a Facebook

application installed in their phone, there is no need to re-enter their credentials. Log in credentials will be fetched automatically by the application. The user will only have to grant this application access to the user's information and his friends list. After a photo has been selected, the application will attempt to do face detection of the image. When no face is detected or multiple faces are detected, the program will display an error message and the game will not start.

If there is exactly one face detected, the user can then proceed to play the game. The user, who is already wearing a magnet in his hand, can then perform a punching gesture towards the phone. When a punch gesture is detected, the user's score will increase by 10. In addition, some bruises will also be displayed on top of the face of the person in the image. After the user decides to finish punching, he or she can then save the image to their local storage or post it to his or her Facebook wall.

B. System Architecture

Fig. 1 is a high level diagram that shows the components of the application. As in Android applications in general, there is one part of the application that serves as an entry point when the user clicks the application's icon on the application launcher. In this application, this component is called StartScreenActivity. From this activity, the user can then navigate to the three different parts of the application as described in the previous section.

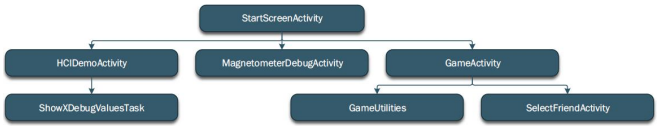

Fig. 1 A High Level Diagram of the Components of the Application

The magnetometer debugging information is named MagnetometerDebugActivity in this application. The implementation of this activity is rather straightforward. It only contains several TextView to display values and a sensor event listener to monitor the changes in magnetic field around the device.

The human -computer interaction demonstration part of the application is named HCIDemoActivity. It contains four ImageView which serve as the container for the image to be displayed. There is also a ToggleButton for the user to switch between finger input and magnetometer input. When an input is detected, the activity will dispatch an AsyncTask which displays X position of where the input is first detected and where the last X position is detected. The AsyncTask is called ShowXDebugValuesTask.

The game part of the application is called GameActivity. This activity is organized in such way that it satisfies the Model-View-Controller (MVC) paradigm. The GameActivity itself will serve as the view in MVC model. The GameActivity will maintain exactly one instance of GameUtilities. This is the class that serves as the controller in MVC model. It contains all the functions and logic needed to run the game. In addition, GameActivity can also dispatch a SelectFriendActivity when a user chooses to select a friend from the user's Facebook.

In the punching game layout as shown in Fig. 2, there will be an image containing the picture of the person being punched. The image displayed here will always have a rounded corner. At the bottom of the layout, there will be buttons to do some actions for loading and saving image. The buttons will not be displayed all at once. Depending on the state of the game, only some buttons will be visible to the user at a time.

Fig. 3 describes the scenario that the users can go through when playing the punching game. The user will first need to choose an image from either the phone's gallery or from the list of the user's Facebook friends. When there is exactly one face detected, the game will start. The user can then punch the person as much as they can. After the user decides to finish punching, they can choose to save the image to the phone's storage or post it to the user's Facebook timeline.

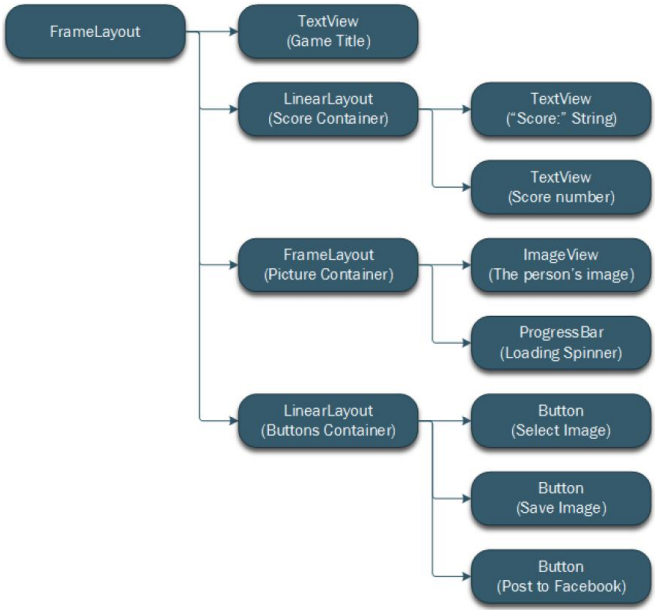

Fig. 2 The Game Layout Structure

IV. Experiments And Results

For the experiments, the following is the hardware and software specification. The application would run on any Android device with minimum API version 11 (Honeycomb). While testing the application, we use a Samsung Galaxy Note 3 device with the following specifications:

- OS: Android 4.4.2 Kitkat
- Chipset: Exynos 5 Octa 5420
- CPU: Quad-core 1.9 GHz Cortex-A15 & Quad-Core 1.3 GHz Cortex-A7
- GPU: Mali-T628 MP6
- RAM: 3GB
- Storage: 32 GB internal memory

A. Implementation and User Interface

In the development period, we were faced with a problem of memory limit. In Android, each application has its own memory budget allocated by the Dalvik VM. Since the game requires a lot of bitmap processing, it exceeds this memory limit immediately. This issue was overcome by enhancing the memory management.

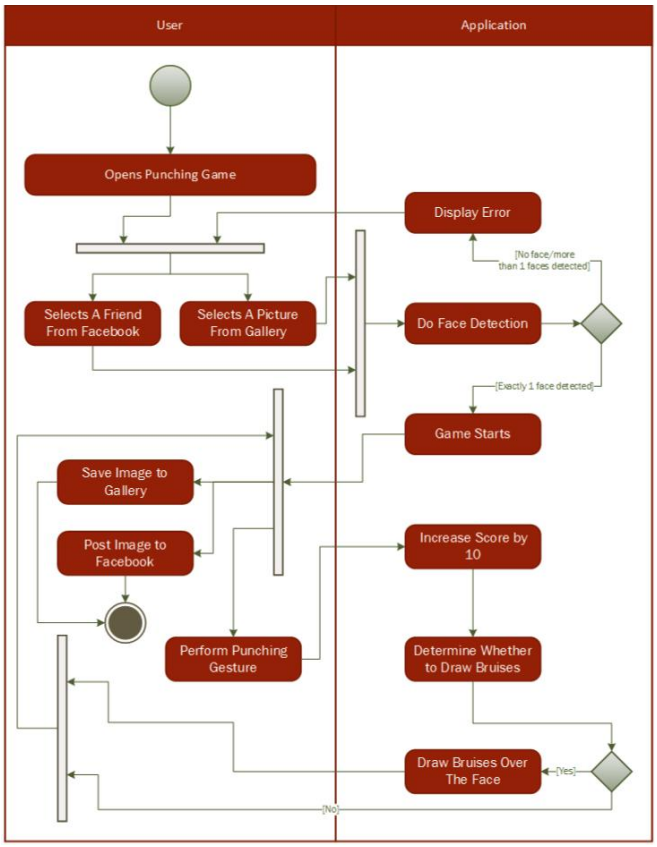

Fig. 3 Activity Diagram for the Game

To make the performance better, all the image processing logic outside the main thread was implemented. All tasks will have its own AsyncTask to do the process in the background. For the image loading, a library called Picasso was used. This library is well known for its simple usage and its automatic image caching feature. When the game first loads, the application will load all the bruise images and resize them accordingly. Since the image has been preloaded, the application will not need to load the bruise images again by the time it needs to draw the bruises.

When an image of a person has been chosen by the user, the application will attempt to perform a face detection algorithm on the image. To perform such a task, the FaceDetector API that is provided from the Android operating system was used. The values returned from the API are the centre coordinates of the face and the distance between the person's eyes. An estimation algorithm was used to estimate the face area from a person's image for the bruise to be drawn over.

Another feature that was implemented is the Facebook functionality. The official Facebook Android SDK comes with several helper classes included. However, these classes were customized to make them suitable for the application.

We show some figures to show the results of the application implementation. Fig. 4 shows the application's start screen where the users can choose from the three application parts. Fig. 5 shows the screens to try the human-computer interaction demonstration or to do the debugging on the magnetometer sensor. Fig. 6 shows the start screen of the punching game. After the user chooses the image, the application will detect the face automatically as shown in Fig. 7. If the detection is successful then the users can start the punching motion in front of their mobile phone. The effects of these punches are shown in the right screen of Fig. 7. After finishing punching the image, users can post the bruised face image to his or her Facebook page as shown in Fig. 8.

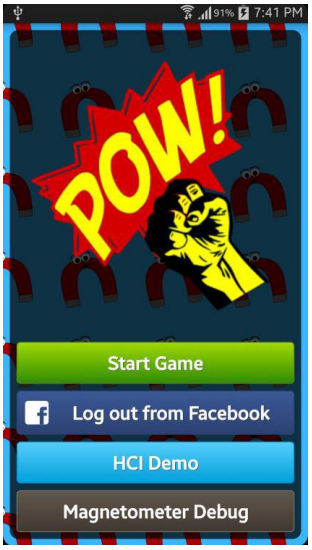

Fig. 4 The start screen of the application

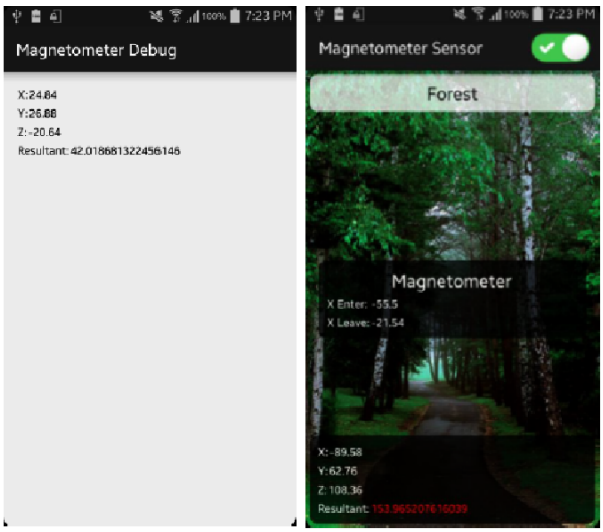

Fig. 5 Magnetometer debugging and HCI demo screen

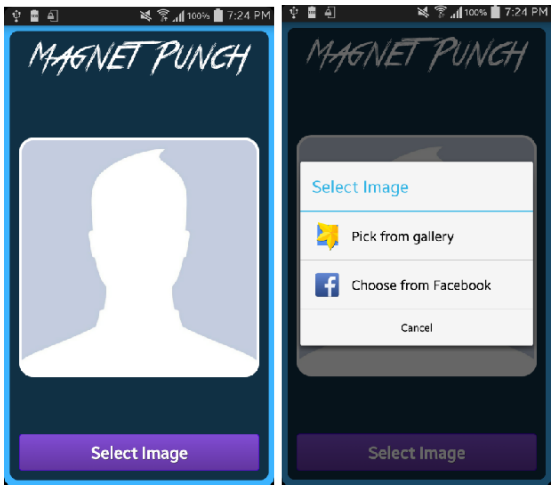

Fig. 6 Game initial screen and image selection screen

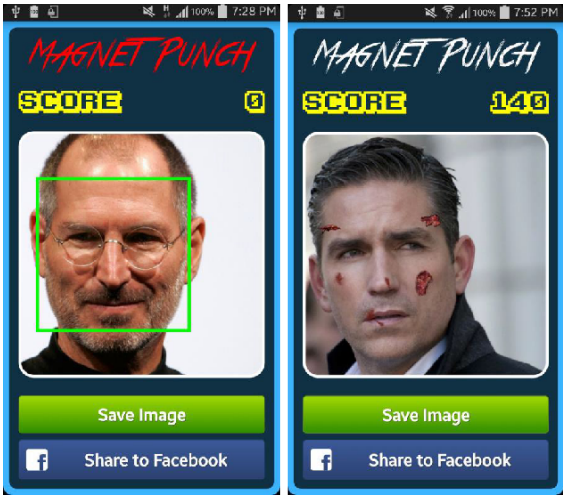

Fig. 7 Face detection overlay and bruises drawn on the face

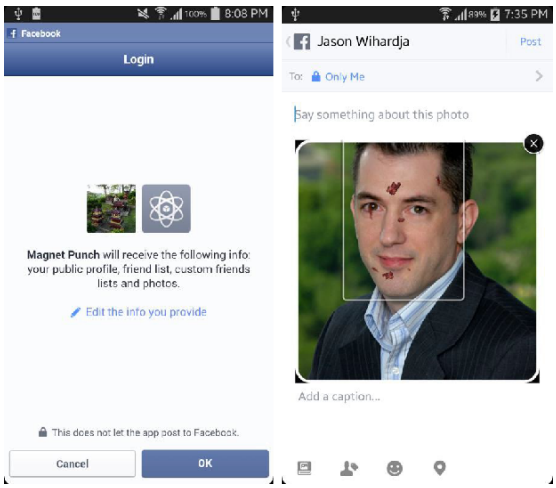

Fig. 8 Facebook login and image posted to Facebook

B. *Experiment Results*

Arguably, the most widely used touchless interaction technology that is still in use today is using the phone's front camera. Therefore, we compared the magnetometer solution against the camera solution to achieve touchless interaction in the following aspects: power consumption, accuracy and responsiveness in detecting gestures.

Fig. 9 shows the reduction in the battery level of the mobile phone for the two solutions. In the first 2 minutes, the battery dropped 1% in the magnetometer solution and dropped 2% in the camera solution. For the next 5 minutes, the battery on magnetometer solution only drops 3% further. In contrast, the battery dropped 7% in the camera solution.

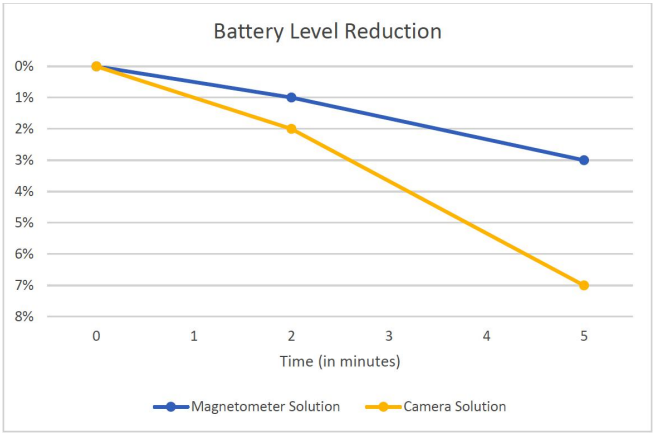

Fig. 9 Power Consumption Comparison

So the relative battery level reduction for the magnetometer solution in the first two minutes is 1%, while the relative battery level reduction for the camera solution is 2%, and the relative battery level reduction for the magnetometer solution from two to five minutes is 2%, while the relative battery level reduction for the camera solution from two to five minutes is 5%. Thus, in term of power consumption, the camera solution consumes a significantly higher power compared to the magnetometer solution.

Fig. 10 shows the results of the accuracy and the responsiveness comparison. In accuracy test, the camera solution detects all 20 inputs that were performed, which is equivalent to 100%, and slightly better than the magnetometer solution, which only detected 18 out of 20 inputs or 90% accuracy.

In responsiveness tests, the camera solution performs much better compared to the magnetometer solution, detecting 19 out of 20 inputs, which is equivalent to 95%. The magnetometer solution only managed to detect 14 out of 20 inputs, which is equivalent to 70%. This might have something to do with the magnetometer sensor itself, which was not sensitive enough to detect such quick changes. An improvement of the magnetometer may make this technology become a mainstream technology for touchless interaction.

Unfortunately, we were not able to conduct a CPU usage test. At the time when the experiments were conducted, Android only supports running one application at a time. As soon as another application comes to the foreground, the background application is suspended. As a result, after a CPU usage monitoring tool was used and started running the solution application to be tested, the CPU

usage monitoring program is immediately suspended and stops running. However, judging from the algorithm complexity, which contributes a lot to the CPU usage, the magnetometer solution clearly has significantly simpler logic compared to the camera solution. The magnetometer solution algorithm only contains some if else branches. The camera solution must run an image processing logic, which is known to be computationally expensive.

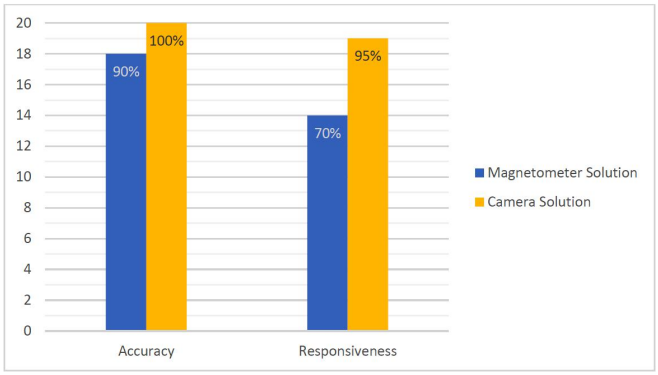

Fig. 10 Accuracy and Responsiveness Comparison

In summary, it can be concluded that the magnetometer solution consumes significantly less CPU resource and power, while the camera solution is more accurate and responsive.

C. Discussion

The first limitation of this solution is the fact that a magnetometer sensor can only detect changes in magnetic fields. Therefore, it would be impossible to detect the presence of multiple magnets. When multiple magnets are used, the magnets will interfere with each other, creating an incoherent magnetic field around the device. In other words, if two or more magnets are used, the sensor can only detect the results of the changes of all the magnets.

While testing the application, it was found that that magnetometers were not originally intended to detect quick changes. Originally, it was set to poll changes every 0.6 seconds. However, this rate is too slow for fast paced applications such as games, which leads to some undetected punch gestures. When the polling rate was increased to 0.2 seconds, this action had some side effects. Sometimes the sensor was getting stuck, stopped polling changes, returned wrong values to the sensor listener, and in the worst case, even crashed the application. Currently we have not found any solutions to this problem.

In addition, we sometimes experienced a condition where the magnetometer lost its ability to detect magnets and required a recalibration to return it back to normal. This commonly happens with many smart phones today. Another similar experiment done in [9] also indicated this issue.

The strength of the magnet will also need to be taken into consideration. While developing the application, we use a refrigerator magnet with a diameter of about 2 cm. When using a smaller magnet with a diameter of about 1 cm, many inputs were undetected. However, we did not do the experiment with larger magnets.

V. Conclusion

This paper explored the use of a magnetometer sensor as an alternative way for conducting touchless interaction. To demonstrate the concept, an application and a punching game that utilized this principle were developed. We attempted to map some common gestures into something that can be understood by a magnetometer sensor. Using the application, users are able to perform common swiping gestures using magnetometer sensor, such as left swipe, right swipe, and selection or punching gestures.

The application's proof of concept successfully demonstrated the idea of implementing a touchless interaction using magnetometers. According to the test results, the proposed magnetometer solution has some advantages and disadvantages over the current camera solution. However, knowing the fact that magnetometer are still underused, especially as a human-computer interaction method, and considering that the price of a magnet and a magnetometer sensor is relatively cheap, we think that this technology might have a promising future.

Finally, we hope that this research can provide an interesting alternative to the touchless interaction area and also expect that this research can serve as a reference for others who are doing research in the same area.

Some possible future work is as follows. The current magnetometer that is equipped with most smartphones today is only intended to be used in a compass application. As a result, its polling rate is relatively slow and thus, is not really suitable for detecting quick movements. A workaround that was implemented in this paper is unstable. Therefore, more research still needs to be done in this area. In addition, the magnetometer device for each phone model is also calibrated differently and as a result, different mobile phones might return different values. In the future, it would be better if the smart phone manufacturers could equip better magnetometer sensors in their smart phones.

Some common gesture mapping is still not standardized at the time of writing. It would be a lot better if there is a standard for the gestures that can be performed using magnetometer interaction in order to provide the same user experience across different applications.

References

[1] B. Priyantha, D. Lymberopoulos, and J. Liu. "Littlerock: Enabling energy-efficient continuous sensing on mobile phones." IEEE Pervasive Computing 10.2 (2011): 12-15.
[2] Samsung. Galaxy S4. Available: http://www.samsung.com/galaxy
[3] X. A. Chen, J. Schwarz, C. Harrison, J. Mankoff, and S.E. Hudson. "Air+ touch: interweaving touch & in-air gestures." Proceedings of the27th annual ACM symposium on user interface software and technology. ACM, 2014.
[4] B. Kellogg, V. Talla, and S. Gollakota. "Bringing Gesture Recognition to All Devices." NSDI. Vol. 14. 2014.
[5] Supermagnete, "What is the safe distance that I need to keep to my devices?," Available: http://www.supermagnete.de/eng/faq/What-is-the-safedistance-that-I-need-to-keep-to-my-devices.
[6] L. Pei, J. Liu, R. Guinness, Y. Chen, H. Kuusniemi and R. Chen. "Using LS-SVM based motion recognition for smartphone indoor wireless positioning." Sensors 12.5 (2012): 6155-6175.
[7] Y. Cai, Y. Zhao, X. Ding, and J. Fennelly. "Magnetometer basics for mobile phone applications." Electron. Prod.(Garden City, New York) 54.2 (2012).
[8] J. Quasdorf, "A Case Study: MR vs. Hall Effect for Position Sensing," Sensors, pp. 15-20, November 2005.

[9] A. Bianchi and I. Oakley, "Designing Tangible Magnetic Appcessories," Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction, pp. 255-258, 2013.