Performance Characterization on Mobile Phones for Collaborative Augmented Reality (CAR) Applications

Víctor Fernández, Juan M. Orduña, Pedro Morillo*
Dept. Informática - Universidad de Valencia - SPAIN

ABSTRACT

Collaborative Augmented Reality (CAR) systems allow multiple users to share a real work environment including computer-generated images in real time. Currently, the hardware features of most mobile phones not only provide excellent multimedia services, but it also includes wireless network capabilities that provides a natural platform for CAR systems. However, the wide variety of these hardware features can have important effects on the performance of the mobile CAR applications.

This paper presents the experimental characterization of CAR applications for mobile phones in regard to well-known performance metrics in distributed systems. Characterization results show that the most time consuming stage in a CAR application is the marker detection stage. Moreover, the rendering stage is decoupled on some devices, depending on the operative system used. This decoupling process allows avoiding low refresh rate, facilitating the collaborative work. These results can be used as the basis for an efficient design of CAR systems and applications.

Keywords: Collaborative Augmented Reality, Mobile Phones, Performance Evaluation

1 INTRODUCTION

Since the beginning of AR systems, the potential of collaborative AR (CAR) systems was exploited for different activities like Collaborative Computing or Teleconferencing [3]. Wearable devices were used to provide CAR systems where a wearable AR user could collaborate with a remote user at a desktop computer [9].

On other hand, a lot of devices comprising a computing embedded system pervade our daily life, and they have been used for CAR systems. One of these devices are mobile phones [5]. Effectively, current mobile phones have full color displays, integrated cameras, fast processors and even dedicated 3D graphics chips, and they have become an ideal platform for CAR systems. However, the wide variety of current mobile phones, with different graphic and processing capabilites, and different operating systems, can have significant effects on the performance of the CAR application. The design of an efficient CAR application must take into account these effects in order to fulfil the required specifications.

In this paper, we propose the characterization of different mobile phones for collaborative augmented reality applications. We have implemented a simple CAR application on a real system and we have measured the performance achieved with different mobile phones, considering operating systems (OS): Android OS and iOS. The results show that the most time consuming stage in a CAR application is the marker detection stage. Therefore, any improvement of the CAR applications (categorized from the obtained results as CPU-intensive but not memory-intensive) should be addressed to improve the image acquisition process. Regarding the operating systems, the results show that the rendering stage is decoupled on devices using the Android OS, in such a way that it is executed with the other stages concurrently. The results also show that some recent mobile phones like IPhone 4 only works with high resolution images. As a result, these mobile devices need a lot of time for detecting the markers in the camera image plane.

*e-mail: {Juan.Orduna, Pedro.Morillo}@uv.es

2 CAR APPLICATIONS ON MOBILE PHONES

There are different kinds of mobile phones, with different operative systems (OS) and capabilities. The most extended OSs for mobile phones are Nokia Symbian, Google Android OS (commonly referred as Android), RIM/Blackberry, Apple iOS, Microsoft Windows Mobile/Phone 7 and Samsung Bada [1]. In this work, we are focusing on two of them, Android and iOS, because they share the vast majority of the current market [4].

CAR applications can be split into four stages: The first stage is denoted as image acquisition state and consists of obtaining an image from the camera's flow. In the second stage, markers are detected from the image obtained before. With the position of this markers, the third stage consists of drawing a 3D object on the image. Finally, in the fourth phase this information (for example, the position(s) of the mark(s)) is sent to the other application nodes through some kind of broadcast.

The first three phases are similar on any AR application [10], but the last one can be performed by using different technologies such as WiFi, 3G or Bluetooth . Although there are some classic CAR applications that uses Bluetooth as the AK-Phone project [2], usually the first two ones are used, because the use of Bluetooth severely limits the spatial range of transmission.

3 CHARACTERIZATION SETUP

We have tested two different mobile phones using Android and another two mobile phones using iOS operating system. For the Android we have considered the Motorola Milestone, with 550 MHz of CPU frequency, and Nexus One, with almost double CPU frequency (998 MHz). For the iOS operating system, we have considered the iPhone 3G, with 412 MHz of CPU frequency, and iPhone 4, with double CPU frequency (800 MHz). All have 5 MPixels of camera resolution, except iPhone 3G, which only has 2 MPixels.

On internet there are some implementations for Android and iOS OSs that are open source. We have used them as a starting point to obtain a CAR application. For the case of the implementation for Android we have used NyARToolKit[8]. NyARToolKit is a complete version of ARToolkit[6] that was exclusively written in Java. This makes it slower in execution than the original, but it also makes it completely independent of the processor architecture. As the original, NyARToolKit is a library of functions visual interpretation and integration of VR data into physical environments, including real-time camera vision functionality and 3D rendering of virtual objects. On the other hand, our iOS implementation has been based on the version developed by Benjamin Loulier [7], which in turn is based on ARToolKitPlus [10].

After getting the application, we did the same procedure as in the Android version: analyzing its stages, putting time marks and adding the sending stage, also with TCP sockets. In this case, we are showed an apple each time we found the mark. In contrast to Android, iOS only provides two camera resolutions: full or half. In

Table 1: Throughput (in terms of FPS) and RTT for each smartphone

	FPS	RTT (ms)
Milestone	1,43	14,14
Nexus One	5,98	5,54
iPhone 3G	2,51	15,42
iPhone 4	1,91	7,06

Table 2: Execution times (ms) for each considered mobile phone

Stages(ms)	Acq.	Detect	Render	Send	Total
Milestone	248,64	288,53	30,42	14,14	698,34
Nexus One	40,25	78,08	13,23	5,54	167,11
iPhone 3G	33,29	58,07	28,26	15,42	398,00
iPhone 4	17,66	182,17	23,34	7,06	523,26

half resolution it obtains the same resolution that in full resolution, but it only analyzes one of every two pixels. In order to make that fairest comparisons as possible, we used half of the resolution, with a resolution of 400x304 on iPhone 3G and a resolution of 1280x720 on iPhone 4.

4 Performance Evaluation

We have measured the system throughput, in terms of frames per second (FPS), and system latency, in terms of the round-trip-time (RTT) for each message sent to the server. Table 1 shows the performance evaluation, in terms of the average number of FPS achieved by each mobile phone, and the latency for the transmission stage of the application.

Table 1 shows that the Milestone is the handheld device with the poorest throughput, and the Nexus One is the one with the best throughput. Although the iPhone 3G provides better throughput than the iPhone 4G, this behavior is due to the fact that the iPhone 3G has to analyze much less amount of data than the iPhone 4, since the size of its image is much smaller. The right column in this table shows that the latency is quite similar in all the mobile phones considered, as it could be expected, because it depends on network features more than the computing capabilities of the considered mobile phone.

Table 2 shows the decomposition of the results shown in table 1. Concretely, the first four columns show the average duration of each stage per cycle for each device, and the rightmost column shows the total aggregated cycle time. The inverse of these times result in the number of frames per second shown for each device in table 1.

Each row in Table 2 shows the number of milliseconds that each stage needs to finish on each device. The Motorola Milestone provides the worst throughput because it is six times slower in obtaining images and almost twice slower in detecting a mark, than the other devices. Regarding the rendering and sending stages, there are also significant differences with other devices.

On Another hand, the behavior of iOS-based devices for AR purposes is different than the behavior of another mobile phone. In this sense, the image acquisition process is twice faster on a iPhone 4 than the same process performed on an iPhone 3G. However, the marker detection stage in an iPhone 4 is three times slower than in the iPhone 3G. Although the CPU of an Iphone 4 is twice as powerful as the CPU included in an Iphone 3G, the reason for this result is that the images processed in an iPhone 4 are six times bigger than the images processed by an iPhone 3G. When Android-based devices are compared to Apple's mobile phones in the analysis of the marker detection stage, Table 2 shows that the Android devices are faster than iOs phones because the images captured from the on-board cameras equipped in the Android devices are four times smaller than the images captured by an iPhone 4. I

Table 2 shows that if total execution time (included in the last column) is compared to the reciprocal value of the throughput (FPS in Table 1) both Android devices have almost a perfect matching. On the contrary, iOS smartphones need twice times to complete the cycle. These results seem to indicate that the Android devices render the final augmented reality scenes more often than the iOS devices. In order to confirm this result, we have measured the number of completed rendering stages compared to the rest of the threads of the CAR framework. The obtained averaged values are 6.28 renderings per cycle in HTC Milestone and 5.55 renderings per cycle in the case of the HTC Nexus One.

Additionally, we have measured the CPU and the memory utilization for the considered mobiles phones in the experiment. The obtained results show that all the considered smartphones are close to reach the saturation point, in terms of CPU usage since (100% CPU usage), when the CAR application was executed. CAR applications (as an enhanced type of AR applications) can be considered as CPU-intensive since they demand the maximum microprocessor resources available in a coupled (iOS) or a decoupled (Android) mode of operation. On the other hand, the results obtained with the considered smartphones show that Android-based smartphones need more memory than the iOS-based mobile phones when the same CAR application is executed on them. The reason of this memory overhead is that the management of the memory in Android smartphones is transparent to the application developers and is based on a Virtual Machine (VM).

5 Conclusions

In this paper, we have proposed a performance characterization of mobile phones oriented to the provide a robust and an efficient design of for Collaborative Augmented Reality (CAR) Applications. The results show that when the same CAR application is executed on different mobile phones, the best throughput, measured in frames per second (FPS), is obtained for smartphones based on Android operative platforms.

The results show that the most time-consuming stage in a CAR application is the marker detection stage, followed by the image acquisition stage, the rendering stage and finally, the transmission stage. Therefore, any improvement of the CAR applications should be oriented to enhance the image acquisition process. Regarding the execution of CAR applications on mobile phones in different operating systems, the results show that the rendering stage is decoupled on devices using the Android OS, in such a way that it is executed with the rest of the stages concurrently. However, this stage can be programmed to work as in iOS operating system in ad-hoc implementations.

Finally, the results also show that some recent mobile phones like IPhone 4 only work with high resolution images. As a result, these mobile devices achieves the most visual quality at the expense of needing a lot of time for detecting the markers in the camera image plane.

Acknowledgements

This work has been jointly supported by the Spanish MICINN and the European Commission FEDER funds under grants Consolider-Ingenio CSD2006-00046 and TIN2009-14475-C04-04.

References

[1] T. Ahonen. *T.A. Phone Book 2010*. T.A. Consulting, 2010.

[2] M. Assad, D. J. Carmichael, D. Cutting, and A. Hudson. Ar phone: Accessible augmented reality in the intelligent environment. In *In OZCHI2003*, pages 26–28, 2003.

[3] M. Billinghurst and H. Kato. Real world teleconferencing. In *Proc. of the conference on Human Factors in Computing Systems (CHI 99)*, 1999.

[4] S. P. Hall and E. Anderson. Operating systems for mobile computing. *J. Comput. Small Coll.*, 25:64–71, December 2009.

[5] A. Henrysson and M. Ollila. Umar: Ubiquitous mobile augmented reality. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, MUM '04, pages 41–45, New York, NY, USA, 2004. ACM.

[6] D. H. Kato. Artoolkit, 2011. Available at http://www.hitl.washington.edu/artoolkit/.

[7] B. Loulier. Augmented reality on iphone using artoolkitplus, 2011. Available at http://www.benjaminloulier.com/.

[8] Nyatla. Nyartoolkit:artoolkit class library for java/c#/android, 2011. Available at http://nyatla.jp/nyartoolkit/.

[9] W. Piekarski and B. H. Thomas. Tinmith-hand: Unified user interface technology for mobile outdoor augmented reality and indoor virtual reality, 2002.

[10] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 125–134, Washington, DC, USA, 2008. IEEE Computer Society.