Proceedings of the 13th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2013
24–27 June, 2013.

On the Characterization of Markerless CAR Systems Based on Mobile Phones

Víctor Fernández[1], Juan Manuel Orduña[1] and Pedro Morillo[1]

[1] *Departamento de Informática, Universidad de Valencia - SPAIN*

emails: Victor.Fernandez-Bauset@uv.es, Juan.Orduna@uv.es, Pedro.Morillo@uv.es

Abstract

Collaborative Augmented Reality (CAR) systems based on mobile phones have experienced a huge expansion last years, since the hardware features of most mobile phones provide excellent multimedia services and wireless network capabilities. In previous works, we improved the performance of CAR systems based on mobile phones that use a marker detection mechanisms. However, CAR systems based on feature vectors have just emerged, changing the way in which Augmented Reality applications work.

In this paper, we propose the characterization and performance improvement of CAR systems based on markerless tracking when using mobile phones. The characterization of client devices show that they work faster with markerless tracking than with fiducial marker tracking, regardless of the phone model and operating system considered. The characterization of the whole CAR system shows that the response times provided by the CAR system remain interactive, except when the system enters saturation. At that point, the system does not drop any message, at the cost of exceeding an interactive response time (250 ms.). As the system enters deep saturation, more messages are dropped and as a consequence the response time is reduced, providing interactive response times again. Thus, we have achieved simulations of CAR system with 1000 client devices for all the considered types of mobile phones without significantly exceeding interactive response times.

Key words: Collaborative Augmented Reality, Mobile Phones,Performance Evaluation, Qualcomm, Vuforia

1 Introduction

Since the beginning of AR systems, the potential of collaborative AR (CAR) systems was exploited for different activities like Collaborative Computing [6] or Teleconferencing [5].

 ISBN: 978-84-616-2723-3

Wearable devices were used to provide CAR systems where a wearable AR user could collaborate with a remote user at a desktop computer [16]. One of these devices are mobile phones [11, 14]. 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 [10]. 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, in terms of system latency, frames per second or number of supported clients with certain latency levels.

Augmented Reality superimposes multimedia content - 3D object, text, sound, etc - on real world through a display or screen. In order to locate digital contents on a specific image of the real world point, some references within the image are needed. These references are known as markers, and two methods are usually used to track them: natural feature tracking and fiducial marker tracking. The former method uses some actual objects of the environment as a target to be tracked in order to place 3D virtual objects. The latter method uses fiducial markers to find a specific position of real world.

In previous works, we characterized the behavior of different mobile phones for Augmented Reality marker tracking, and we also proposed some improvements for CAR systems based on fiducial markers and mobile phones as client devices [3, 2, 7]. However, the advent of Vuforia [17], released by the ARM-processor company Qualcomm at the end of the 2011, has allowed the widespread use of markerless-based CAR applications worldwide. The reason for its popularity is that, unlike another approaches such as NyARToolkit [15], the VuforiaSDK supports for more than 400 different smartphones and tablet models, and it tracks real objects on the current frame with a impressive fluidityand reliability.

In this paper, we propose the characterization and performance improvement of CAR systems based on markerless tracking when using mobile phones. The characterization of client devices show that they work faster with markerless tracking than with fiducial marker tracking, regardless of the phone model and operating system considered. The characterization of the whole CAR system shows that the response times provided by the CAR system remain interactive, except when the system enters saturation. At that point, the system does not drop any message, at the cost of exceeding an interactive response time (250ms.) [9]. As the system enters deep saturation, more messages are dropped and as a consequence the response time is reduced, providing interactive response times again. Thus, we have simulated CAR system with 1000 client devices for all the considered types of mobile phones. Unfortunately, since the Vuforia library is not open source code, we cannot find the reason for that behavior.

The rest of the paper is organized as follows: Section 2 shows some background about CAR applications on mobile phones. Section 3 describes the characterization setup, and Section 4 shows the characterization results and the performance obtained with this new values in our CAR server. Finally, Section 6 presents some conclusions remarks.

 ISBN: 978-84-616-2723-3

2 Background

Any CAR application needs a device equipped with an onboard camera, CPU and display. The most common devices used for CAR applications are Tablet-PCs or 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 [8].

Augmented reality superimposes multimedia content - 3D object, text, sound, etc - to real world through a display or screen. In order to locate digital contents on a specific image of the real world point, some references within the image is needed. These references are known as markers. The Augmented Reality marker tracking process in CAR applications can be split into four stages: The first stage is denoted as image acquisition stage, and it consists of obtaining an image from the camera's flow. In the second stage, markers are detected from the image obtained before. Using the position of this markers, the third stage consists of drawing a 3D object on the image. Finally, in the fourth phase some information (for example, the position(s) of the mark(s)) is sent to the other application nodes through some kind of broadcast communication.

Two methods are usually used in CAR applications for the marker detection stage: natural feature tracking and fiducial marker tracking. The former method uses interest point detectors and matching schemes to associate 2D locations on the video with 3D locations [22]. This process can be grouped in three big phases: interest point detection, creation of descriptor vectors for these interest points, and comparison of vectors with the database [13]. The latter method uses fiducial markers to find a specific position of real world. This process can be divided in three phases: edge detection, rejection of quadrangles that are too large or too small, and checking against the set of known patterns [22].

There are few solutions based on fiducial marker tracking over mobile phones. In 2003, ArToolKit [12], one of the most well-known software libraries for developing Augmented Reality (AR) application, was released for Windows CE, and the first self-contained application was developed for mobile phones [23]. This software evolved later as the ArToolKitPlus tracking library [22]. A tracking solution for mobile phones that works with 3D color-coded marks was developed [14], and a version of ArToolKit for Symbian OS was developed, partially based on the ArToolKitPlus source code [10]. The research teams behind these works have worked on fiducial marker tracking, but not from the collaborative point of view. Also, there are many other works that focus on natural feature tracking [22], [19], [20], [21].

 ISBN: 978-84-616-2723-3

3 Characterization setup

We propose the characterization of each of the stages of a CAR application over different mobile phones. For characterization purposes, the application performs the sending of the positions of the marks. In the CAR server side, we have developed a multithreaded CAR server that supports simulated clients (simulated mobile devices) with the behavior measured in the characterization part, as we did for marker-based CAR systems [3, 7]. We have time-stamped every message generated within this CAR system, in order to measure the performance of every device. The system configuration consists of one server, and a certain amount of mobile devices that are scanning the visual space of their video camera, looking for a marker that will be converted into a 3D object in their display. The action cycle performed by each client is composed of the steps illustrated by Figure 1, which can be described as follows: first, it performs one new image acquisition followed by a marker detection stage. Then, the client waits until the cycle period (determined by the action frequency, a system parameter) finishes. Next, if the acknowledgments from all the neighbors have been received, then a new message with the new marker location is sent. If all the acknowledgments have not been received, then it waits for a maximum waiting threshold of 20 seconds, and then a new round of messages (with the latest marker location) are sent to the neighbors through the server. The neighbors simply return an ACK message to the sender device through the server. The server simply forwards the messages to the corresponding destination clients.

We have tested two different mobile phones using Android and another two mobile phones using iOS operating system. We have selected the same phone models for comparison purposes with our previous studies for CAR systems based on fiducial markers. Table 1 shows the main features of these mobile phones. For the Android operative system, we have considered the Motorola Milestone, with 550 MHz of CPU frequency, and HTC Desire, with almost double CPU frequency (998 MHz). For the iOS operating system, we have considered the iPhone 3GS, with 412 MHz of CPU frequency, and iPhone 4, with double CPU frequency (800 MHz). All of them include a 5 megapixels (5 MP) resolution camera, except iPhone 3GS, which equips a 2 megapixels (2MP) resolution camera.

OS	Android		iOS	
Model	Milest.	Desire	iPh 3GS	iPh 4
CPU (MHz)	550	998	412	800
RAM (MB)	256	512	128	512
Camera (MP)	5,02	4,92	1,92	4,92

Table 1: Hardware features of the considered mobile phones

The implementations we have used are those given in the Qualcomm site for both the

 ISBN: 978-84-616-2723-3

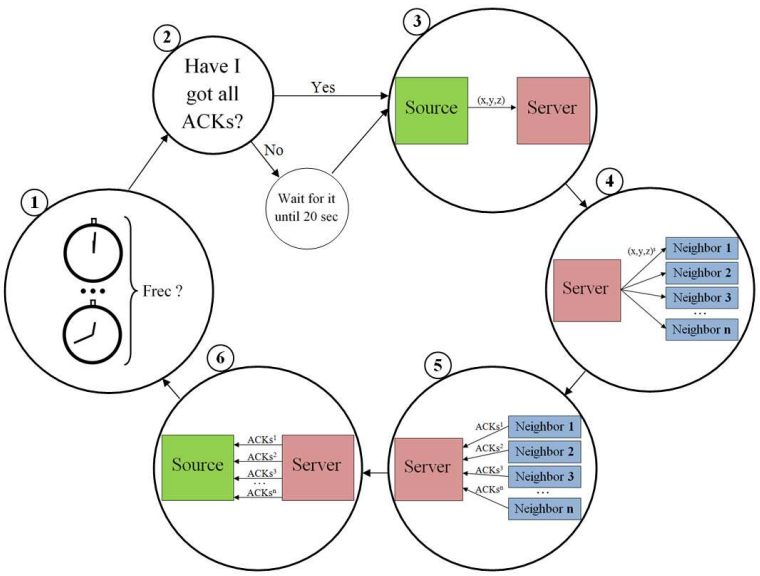

Figure 1: A description of the most common stages in the AR marker tracking process

Android and iOS [18] operating systems. They are very similar (in fact, they first implemented the Objective-C version, and then they used JNI to get the Android version). We added the sending step described above to these implementations, converting the Augmented Reality(AR) application into a CAR application. This step involves a simple socket used to send to the server the position obtained each cycle from the real world.

4 Clients Characterization

The first step in improving the performance of CAR systems based on markerless detection is the characterization of client mobile phones, in terms of both the time required by the system to complete each CAR stage, and the time required by each smartphone to complete a CAR cycle. Since we already performed a characterization with an implementation using fiducial markers [3, 7], a performance comparison is made. In order to do so, table 2 shows the same results when using fiducial markers (the upper part of the table, labeled with the name of the libraries used, NyARToolKit ARToolKit Plus) and the results obtained with the markerless implementation (labeled with the name of the library used, Qualcomm).

Each row in Table 2 presents the results for a mobile phone. The first two rows show mobile phones using Android OS (representative examples high-end and middle-end devices), and the last two rows show mobile phones executing iOS (idem). The same devices

 ISBN: 978-84-616-2723-3

NyARToolKit & ARToolKit Plus					
Phases(ms)	Camera	Detection	Render	Sending	Total
Milestone	248,64	288,53	30,42	14,14	698,34
Desire	40,25	78,08	13,23	5,54	167,11
iPhone 3GS	33,29	58,07	28,26	15,42	398,21
iPhone 4	17,66	182,17	23,34	7,06	523,26
Qualcomm					
Phases (ms)	Camera	Detection	Render	Sending	Total
Milestone	19,58	3,95	2,38	15,06	86,41
Desire	9,97	18,03	1,76	8,48	56,68
iPhone 3GS	11,06	8,23	14,29	17,38	145,53
iPhone 4	8,15	6,21	17,85	9,68	105,91

Table 2: Execution time (ms) per stage for each considered mobile phone for both systems

are used for markerless tracking, in the lower part of the table. The first four columns represent each phase of the CAR applications, and the last one represents the total time needed to complete a cycle. All values are represented on milliseconds (ms.), and they are averaged values. As it can be seen, the system based on markerless tracking is much faster than the system based on fiducial markers in each CAR stage, except the in sending stage, where both systems show similar times. This is due to the fact that the sending step exclusively depends on the network parameters and status,and the same network with the same configuration were used for both characterizations.

Table 2 shows that the time required for the image acquisition stage in the markerless tracking system is at least half of the one required by the fiducial marker system (the case of iPhone 4 case), while for the case of iPhone 3GS it works here three times faster, from 33'29ms. to 11'06ms. However, the best improvement is achieved for the Milestone, passing from the 248'64 ms. required by the fiducial marker tracking to only 19'58ms required by the markerless tracking.

Regarding the second stage, the performance differences are even larger. The time required for completing this stage by the markerless implementation are at least one fourth of the time required by the fiducial markers implementation. The reason for this behavior is difficult to find, since Qualcomm does not provide the source code of the implementation, but the compiled library. It is also worth mention that each considered device provides different image resolutions, and the size of these resolutions is very different from Android devices to iOS devices. As shown in our previous works [3], it is not the same to analyze a image of 320x480 pixels searching a mark that doing the same for a image of 1280x720 pixels, because the last one needs too much time to be analyzed. Regarding the third stage, table 2 shows that the markerless implementation does not improve this stage for all the

 ISBN: 978-84-616-2723-3

considered devices, as it is the case for the iPhone 4, which requires a similar time in both implementations.

Finally, the last column represents the time (in milliseconds) required by each device to complete a CAR cycle when using a given implementation. Comparing the two implementations, we can conclude that all devices work at least three times faster when using the markerless implementation. Therefore, the markerless implementation is better for any kind of device, either high-end or middle-end. One reason that can contribute to these results is that the fiducial marker tracking implementations were not designed explicitly for the current mobile phones. However, the main reason is the efficient markerless implementation, although we cannot deeply analyze its performance because it is not an open source code.

In order to show the practical effects of the performance achieved by both implementations, Table 3 shows the same values shown in table 2, but expressed in performance parameters like Frames Per Second (FPS), that is, the amount of CAR cycles done in one second, and the Round Trip Time (RTT). As before, each row represents the devices we are using. The FPS are shown in the first column. The RTT are represented in the second column (in ms.), and it corresponds to the values in the fourth stage showed in table 2.

	NyARToolKit		Qualcomm	
	FPS	RTT (ms)	FPS	RTT (ms)
Milestone	1,43	14,14	11,57	15,06
Desire	5,98	5,54	17,64	8,48
iPhone 3G	2,51	15,42	6,87	17,38
iPhone 4	1,91	7,06	9,44	9,68

Table 3: Throughput (in FPS) and RTT for each smartphone in both systems

Table 3 shows that the slower device when using the Qualcomm implementation is the iPhone 3GS, working almost over 7 FPS. This "slowest" FPS on Qualcomm is faster than any of the performances obtained with NyARToolKit or ARToolKit Plus, whose fastest device was the HTC Desire, with a FPS of almost 6. These results show that markerless tracking provide better results for CAR applications executed on mobile phones.

5 CAR System Performance

After the characterization of client devices, we simulated a CAR system with client devices with that behavior, in order to analyze and improve the performance provided by CAR systems using these devices. Like in the case of fiducial marker tracking [4], we used a server based on UDP sockets . We performed tests on the CAR system varying the number of client devices (from 100 to 1000), and the number of neighbors inside the workgroup

 ISBN: 978-84-616-2723-3

(5, 10, 20 and 25). Due to space limitations, we will only show here the results for CAR systems using one of the intermediate client devices, the iPhone 3GS, as a representative example of slow devices (145.53ms. per cycle).

Table 4 shows the CAR system performance when all the client devices are iPhone 3GS devices. This table show four subtables, for the cases of 5, 10, 20 and 25 devices per working group. A working group is the subset of neighbor client devices that are collaborative working on the same task (displaying the same scene), and therefore they are the subset of neighbor devices to which the marker location should be sent in each cycle. The greater working group size, the higher number of messages should be sent and acknowledged per each cycle. In order to obtain comparable results with the case of fiducial marker tracking [4], the messages containing the marker location are sent through the server (that is, it sends the location update message to the server, and then the server re-sends the message to the appropriate clients). For performance evaluation purposes, the destination clients return an acknowledgment message (ACK) to the server, which, in turn, forwards it to the source client.

The first (most left) column in each subtable Table 4 shows the number of simulated devices in the CAR system. The next five columns, from left to right, show the average system response time for all the clients (measured in milliseconds), labeled as "RT", and its standard deviation (labeled as "Dev"). This response time is measured as the time elapsed since the origin client device sends a new position to the server until the instant when the origin device receives all ACKs from its neighbors in the workgroup. The third column shows the percentage of CPU utilization in the system server during the simulation. The fourth column represents the server's response time, measured as the time elapsed since the server sends a message to a client until the instant when the server receives the answer to that message. Finally, the last column represents the percentage of lost packets (this server works with UDP sockets) in regard to the total number of messages exchanged.

Table 4 shows that for a working group size of 5 devices the latency remains almost constant, around 10 milliseconds. The standard deviation increases slightly as the population increase, from 3'53ms. to almost 21ms. The CPU consumption also increases but does not reach 75%. The response time in the server remains constant too and no packets are lost. That is, when all the clients in the CAR system are iPhone 3GS terminals and the working group size is 5, then the system works under a low workload, even when supporting one thousand clients. The worst response time is obtained in the workgroup of 10, obtaining values slightly higher than the interactive times (250 ms. [9]), with 257ms of response time for the case of 1000 devices. Here the response time grows linearly with the population. The standard deviation grows higher than before, reaching 65'86ms.. Like the CAR systems based on fiducial marker tracking, the system saturation is reached when the percentage of CPU utilization reaches 85%, since no CPU utilization values higher than 85,3% are reached. Nevertheless, the system does not enter deep saturation even for 1000

 ISBN: 978-84-616-2723-3

iPhone 3GS										
WG	5					10				
Disp.	RT	Dev	CPU	RT_S	%L	RT	Dev	CPU	RT_S	%L
100	6,43	3,53	19,20	3,10	0,00	15,73	5,82	18,60	7,81	0,00
200	7,54	5,85	21,63	3,46	0,00	13,46	9,23	32,93	6,09	0,00
300	9,44	7,31	29,10	4,29	0,00	20,67	17,12	47,13	8,86	0,00
400	6,04	7,24	34,30	2,84	0,00	30,13	24,75	58,80	12,80	0,00
500	8,96	12,66	42,63	4,26	0,00	42,12	33,58	73,47	17,36	0,00
600	12,11	18,52	49,00	5,04	0,00	101,09	52,79	83,83	49,49	0,00
700	12,10	20,22	59,63	5,14	0,00	174,63	50,59	84,50	85,83	0,00
800	6,87	13,53	63,23	3,07	0,00	202,13	50,49	85,23	99,40	0,00
900	8,14	20,92	72,83	3,41	0,00	228,80	56,26	85,30	112,64	0,00
1000	7,03	16,51	74,83	2,71	0,00	257,06	65,86	85,20	126,32	0,00
WG	20					25				
Disp.	RT	Dev	CPU	RT_S	%L	RT	Dev	CPU	RT_S	%L
100	13,08	8,82	72,17	5,69	0,40	9,25	6,42	71,63	3,82	0,82
200	22,37	16,12	76,20	10,12	0,31	20,65	13,46	78,57	9,34	0,90
300	26,84	29,76	79,10	12,23	0,29	28,53	37,83	80,53	13,03	0,94
400	48,68	37,38	81,57	23,01	0,39	39,97	28,63	82,00	18,83	0,93
500	66,94	49,25	82,77	31,55	0,38	53,52	35,90	83,90	25,16	0,92
600	84,72	65,54	84,47	40,99	0,43	71,21	45,91	84,10	33,87	0,94
700	102,73	77,29	82,93	49,71	0,43	77,87	55,10	84,57	37,50	0,98
800	139,70	79,44	84,60	68,12	0,46	89,79	66,15	85,03	43,50	0,98
900	165,61	86,26	85,30	81,11	0,50	102,51	72,44	84,73	49,56	1,03
1000	178,73	87,04	85,23	87,05	0,54	124,03	83,96	84,73	59,93	0,99

Table 4: System performance for the iPhone 3GS

client devices, since no packets are dropped. When this point is reached, the response time in the server, reaching 126'32ms. for 1000 devices.

The results shown by Table 4 for the working group size of 20 devices are not worse (in terms of system response times) than the ones shown for a working group size of 10 devices. Moreover, the response times obtained for a working group size of 25 devices show shorter response times, although the percentages of CPU utilization are higher. The reason for that behavior is that some messages are dropped, and as a result the average response time significantly decreases.

Although they are not shown here due to space limitations, the results for another faster devices were similar, although the average response times were slightly higher. These results show that when using natural feature tracking, the response times provided by the CAR system remain interactive, except when the system enters saturation. At that point, the system does not drop any message, at the cost of exceeding the interactive response time.

 ISBN: 978-84-616-2723-3

As the system enters deep saturation, more messages are dropped and as a consequence the response time is reduced, providing interactive response times again. Unfortunately, since the Vuforia library is not open source code, we cannot find the reason for that behavior.

6 Conclusions

In this paper, we propose the characterization and performance improvement of CAR systems based on markerless tracking when using mobile phones. The characterization of client devices show that they work faster with markerless tracking than with fiducial marker tracking, regardless of the phone model and operating system considered. The characterization of the whole CAR system shows that the response times provided by the CAR system remain interactive, except when the system enters saturation. At that point, the system does not drop any message, at the cost of exceeding an interactive response time (250ms.) [9]. As the system enters deep saturation, more messages are dropped and as a consequence the response time is reduced, providing interactive response times again. Thus, we have simulated CAR system with 1000 client devices for all the considered types of mobile phones. Unfortunately, since the Vuforia library is not open source code, we cannot find the reason for that behavior.

Acknowledgements

This work has been jointly supported by the Spanish MICINN and the European Commission FEDER under grants TIN2009-14475-C04-04 and TIN2011-15734-E.

References

[1] Ahonen, T.: TomiAhonen Phone Book 2010. TomiAhonen Consulting (2010)

[2] Bauset, V.F., Orduña, J.M., Morillo, P.: On the characterization of car systems based on mobile computing. In: Proc. of IEEE 14th Int. Conf. on High Performance Computing and Communication (HPCC-ICESS),. pp. 1205 –1210 (june 2012)

[3] Bauset, V.F., Orduña, J.M., Morillo, P.: Performance characterization of mobile phones in augmented reality marker tracking. In: Proceedings of the 12th International Conference on Computational and Mathematical Methods in Science and Engineering. CMMSE '12, vol. 2, pp. 537–549. La Manga, Spain (July 2012)

[4] Bauset, V.F., Orduña, J.M., Morillo, P.: How large scale car systems based on mobile phones should be implemented (2013)

 ISBN: 978-84-616-2723-3

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

[6] Billinghurst, M., Poupyrev, I., Kato, H., May, R.: Mixing realities in shared space: an augmented reality interface for collaborative computing. In: IEEE International Conference on Multimedia and Expo (ICME 2000). vol. 3, pp. 1641–1644 (2000)

[7] Fernández, V., Orduña, J.M., Morillo, P.: How mobile phones perform in collaborative augmented reality (car) applications? The Journal of Supercomputing pp. 1–13 (2013), http://dx.doi.org/10.1007/s11227-013-0925-8

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

[9] Henderson, T., Bhatti, S.: Networked games: a qos-sensitive application for qos-insensitive users? In: Proceedings of the ACM SIGCOMM 2003. pp. 141–147. ACM Press / ACM SIGCOMM (2003)

[10] Henrysson, A., Billinghurst, M., Ollila, M.: Face to face collaborative ar on mobile phones. In: Mixed and Augmented Reality, 2005. Proceedings. Fourth IEEE and ACM International Symposium on. pp. 80 – 89 (October 2005)

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

[12] Kato, D.H.: Artoolkit (2011), available at http://www.hitl.washington.edu/artoolkit/

[13] Lee, S.E., Zhang, Y., Fang, Z., Srinivasan, S., Iyer, R., Newell, D.: Accelerating mobile augmented reality on a handheld platform. In: Computer Design, 2009. ICCD 2009. IEEE International Conference on. pp. 419 –426 (October 2009)

[14] Mahring, M., Lessig, C., Bimber, O.: Video see-through ar on consumer cell-phones. In: ISMAR'04. pp. 252–253 (2004)

[15] Nyatla: Nyartoolkit:artoolkit class library for java/c#/android (2011), available at http://nyatla.jp/nyartoolkit/

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

[17] Qualcomm: Vuforia sdk 1.5. Available at http://www.qualcomm.com/solutions/augmented-reality (2012)

 ISBN: 978-84-616-2723-3

[18] Qualcomm: http://www.qualcomm.com/solutions/augmented-reality

[19] Srinivasan, S., Fang, Z., Iyer, R., Zhang, S., Espig, M., Newell, D., Cermak, D., Wu, Y., Kozintsev, I., Haussecker, H.: Performance characterization and optimization of mobile augmented reality on handheld platforms. In: Workload Characterization. IISWC 2009. IEEE International Symposium on. pp. 128 –137 (2009)

[20] Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., Schmalstieg, D.: Real-time detection and tracking for augmented reality on mobile phones. Visualization and Computer Graphics, IEEE Transactions on 16(3), 355 –368 (May-June 2010)

[21] Wagner, D., Schmalstieg, D., Bischof, H.: Multiple target detection and tracking with guaranteed framerates on mobile phones. In: Mixed and Augmented Reality. ISMAR 2009. 8th IEEE International Symposium on. pp. 57 –64 (2009)

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

[23] Wagner, D., Schmalstieg, D.: First steps towards handheld augmented reality. In: Proceedings of the 7th IEEE International Symposium on Wearable Computers. pp. 127–135. ISWC '03, IEEE Computer Society, Washington, DC, USA (2003)

[24] Wagner, D., Schmalstieg, D.: Artoolkitplus for pose tracking on mobile devices. In: 12th Computer Vision Winter Workshop. CVWW'07 (Feb 2007)

 ISBN: 978-84-616-2723-3