Examining the Practical Challenges of an Augmented Reality Cyber-Infrastructure Framework

Chenjie Wei, Chiu Ni Wang, Rajiv Ramnath, Jay Ramanathan
The Ohio State University, Columbus
Department of Computer Science and Engineering
395 Dreese Laboratories, 2015 Neil Ave.
Columbus, OH 43210, U.S.A
{weic, wangch, ramnath, jayram}@cse.ohio-state.edu

ABSTRACT

In this paper, we present issues to be addressed and practical solutions for these issues in an Augmented Reality (AR) cyber-infrastructure framework currently being developed. This framework provides services to AR application developers and users for creating, sharing, publishing, and consuming AR content. It also helps in managing different content datasets, including user-generated AR content, internal and external content repositories. The practical development challenges can be categorized into three groups: computation challenges, sensor capability challenges, and challenges associated with development and deployment of the platform. Specifically, these challenges include finding the optimal update frequency for tracking a user's motion, working with delays in the GPS, thread synchronization, adverse interactions with the hosting device, as well as debugging and testing of a pose computation in AR application. Limitations and practical solutions for both an AR mobile device and AR frameworks are discussed, along with an analysis of characteristics of AR frameworks.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *Artificial, augmented, and virtual realities.*

General Terms

Performance, Design, Human Factors.

Keywords

Augmented Reality, Cyberinfrastructure, Content Sharing, Modifiability, Software Frameworks.

1. INTRODUCTION

Augmented Reality (AR) is a technology that enhances the physical worldview by overlaying information on top of the objects of interest. The augmented information is computer generated sensory inputs, such as sound and graphics superimposed on top of the user's display view. Milgram and Kishino defined the term "Reality-Virtuality Continuum" to explain different forms of digitally enhanced worldviews [16]. According to them, augmented reality is a special kind of mixed reality, where there is a presence of both real world and virtual objects. Essentially, AR is an extension of reality achieved through computing.

The availability of sensors and hardware features in smartphones has made it viable to deploy AR-based applications on mobile devices [11]. However, although developers have been successful in doing so, there are limitations imposed by the platform.

In this paper we will briefly describe a cyberinfrastructure framework for building commercial applications using AR technology. This framework provides services to various types of users – including developers, authors and curators of AR content, and end users that consume the content. The specific focus of this paper will be on the practical and technical issues that arise in the development of mobile AR apps, a discussion of possible solutions within the framework and in the possible application design to address these challenges. We discuss the framework in the context of a game called Game Acre, a pose-computation AR application developed for an industry partner that develops consumer-oriented, pervasive, location-based games for use within sports stadiums, university campuses and similar places. Game Acre is loosely based on a scavenger hunt, with rules and a storyline. Game Acre is intended to be a means of brand promotion, with the virtual elements being related to a particular brand, and with prizes being merchandise of the brand. The Game Acre application has been initially targeted at the iPhone^{TM} and iOS^{TM}, and currently employs an open source library called the iPhone^{TM} ARKit (http://www.iphonear.org/).

The rest of this paper is structured as follows: Related work is presented in Section 2. Our AR framework is introduced in Section 3, and the Game Acre application is described in Section 4, so that we can lead into a detailed discussion on challenges in Section 5. Section 6 further discusses the characteristics of this framework. Finally, Section 7 concludes this paper and presents future work.

2. RELATED WORK

Gotow, Zienkiewicz, White and Schmidt [3] present and discuss the challenges of AR mobile applications in their work focused on providing lightweight algorithms and solutions to address the computational demands of AR.

AR platform design research is presented in [1][4][12]. This work targets building frameworks for rapid content creation and consumption by providing a platform for non-programmers to generate and share content. The authors of [5] have attempted to unify different AR systems. In general, the approach adopted in framework development has been to identify the common components of an AR system in order to provide them in a baseline framework (along with extension points for adding additional capabilities).

There are several existing authoring tools for developing AR desktop and mobile phone applications, all of which can serve as starting points for application development. Note that in each tool, additional code must be added in order to create a complete application [4]. ARToolKit [6] is an example of a low-level computer vision-tracking library, while Studierstube [7] is a high-level programming library. Layar, wikitude and Junaio have been evaluated in [13], which are pose computation AR applications.

Schmalstieg and Reitmayr derive their model from direct analysis of the infrastructure required for ubiquitous computing in [10]. In this model, a knowledge environment based on cyberinfrastructures [8] (also called the *Cyberinfrastructure Information Ether*) is not confined to desktops and laptops but includes "smart" objects spread out in the physical environment.

The vision of a cyberinfrastructure is captured in the Atkins report [8][9] as "Applications are enabled and supported by cyberinfrastructure, which incorporates a set of equipment, facilities, tools, software, and services that support a range of applications."

In the research done at the Nokia Research Center, for Mixed Reality Solutions [1], Belimpasakis, You and Selonen proposed an infrastructure for rapid content creation. The framework design discussed in the paper [5] is based on flexible and modular software reusable components.

In this paper, we present a cyberinfrastructure for AR applications where different kinds of users can participate in content creation, sharing, and consumption, and application development. The focus is an investigation of the *practical* development challenges. They include finding the optimal update frequency for tracking users' motion, working with delays in the GPS, thread synchronization, adverse interactions with the hosting device, and debugging and testing of a pose computation AR application.

3. A CYBERINFRASTRUCTURE FRAMEWORK FOR AUGMENTED-REALITY APPLICATIONS

To begin with, our comprehensive requirements analysis showed that there are several user roles [1] that must be supported by the AR framework. They are (shown in Figure 1) as follows:

- ∞ Users: These only consume AR content
- ∞ Power Users: These consume and create AR content
- ∞ Small Business Users: These focus on local business, and provide *commercial* AR content, to which they seek to attract local or passerby consumers; their uploaded content is mainly collected by the AR system's internal repository
- ∞ Enterprise Users: These provide globally accessible external repositories of business records for AR applications, which focus on AR content renewal of all local chain stores
- ∞ Developers: These develop AR applications based on the AR components and shared content, provided by the roles above
- ∞ Service Providers: These maintain and manage the services of the AR infrastructure.

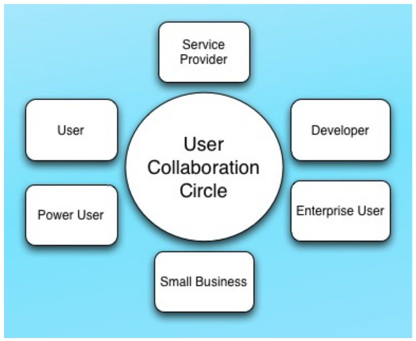

Figure 1. User collaboration circle.

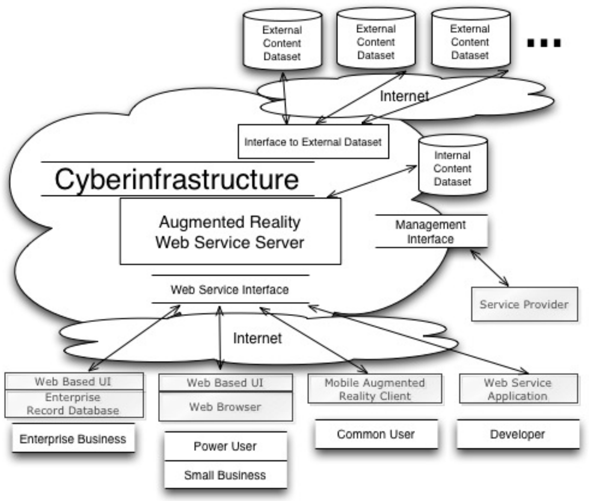

Figure 2. High-level system architecture for cyberinfrastructure ([1]).

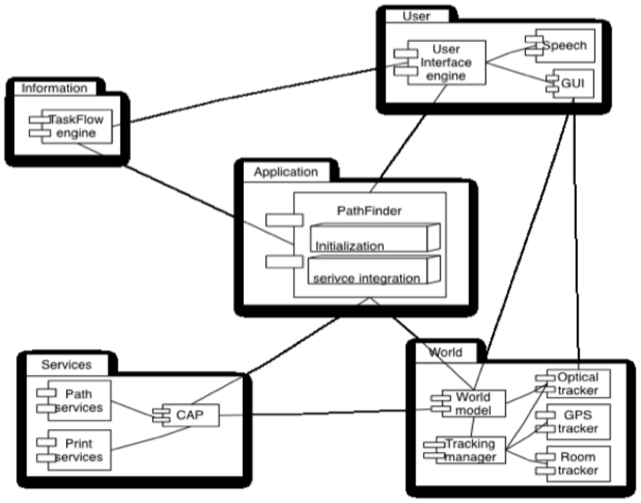

Figure 3. Services within the Cyberinfrastructure architecture for application development ([5]).

We next describe the architecture of the AR cyber-infrastructure framework (see Figure 2). This framework is responsible for managing access to the various datasets containing AR content. The datasets include either user generated content or external content hosted by third-party servers. (Users are provided with standard web interfaces for publishing, sharing consuming and creating content. Normal users and small business users can link their web accounts hosted on different sources like Flickr, Twitter, Facebook, etc. to create and share AR content). Developers use an application-programming interface (API) for creating custom AR applications.

As shown in Figure 3 the API is divided into four functional areas, Services, Information, World and User. The Services component represents the external services that are not part of the cyber-infrastructure. The World component handles the modeling of the physical space; for example, in a marker-less AR application, natural features of the real objects and optical tracking are used. The User component defines how a user interacts with an AR system. Finally, the Information component handles access to the information provided by the server.

4. GAME ACRE APPLICATION ARCHITECTURE

Game Acre is a “scavenger hunt” like multi-player game, targeted at small businesses and companies looking for a new channel to deliver digital advertising. So that the application can be sold to multiple organizations, the software architecture is designed to allow customization (in other words, modifiability). In this paper, Game Acre is the example used to concretely illustrate practical challenges in developing AR applications.

The architecture framework designed for Game Acre consists of two main layers – the AR framework and the game layer. The AR framework is made up of the fundamental components (such as tracking and image registration) required for developing AR applications. It is layered on the open source iPhone ARKit library, and is composed of three main modules, namely AR object, AR view controller and Data (see Figure 6), as follows:

- AR object is used for defining the attributes of a custom virtual object. For Game Acre, the attributes include geo coordinates (Latitude, Longitude), distance label (for displaying distance between the user and nutria), inclination and azimuth. It is basic information used by the tracking system to render the images on the camera view.
- AR view controller holds the main logic for pose computation (tracking) and image rendering.
- Data module serves as a repository for the graphics and the coordinate values of the virtual object.

The baseline Game Acre application involves the capturing of imaginary creatures called “nutria” placed in an outdoor “game field”. Figure 4 shows screen shots of Game Acre as it initializes after startup. Figure 5 shows when the user is approaching the nutria, and its capture.

The Game layer provides the means for implementing the customized game logic and graphics. It consists of the Game Set-up, Game-in-action, and Game Score modules. A configuration file specifies the initial game set up and the images to be used. Currently, game rules and behavior are implemented using code, which makes up the “glue logic” that connects the different components.

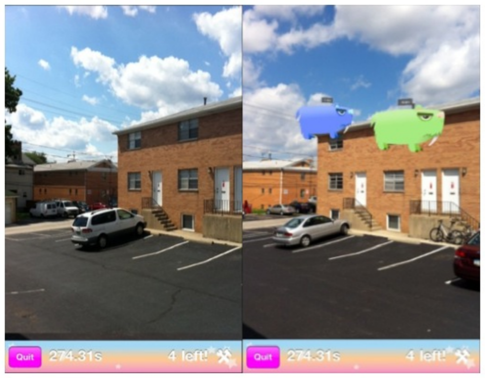

Figure 4. Game setting (left: game start; right: initial nutrias).

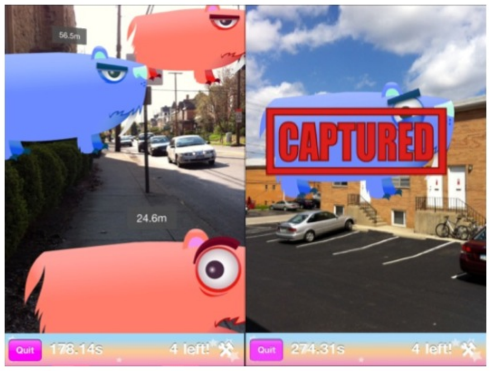

Figure 5. Game setting (left: close to nutrias; right: nutria captured).

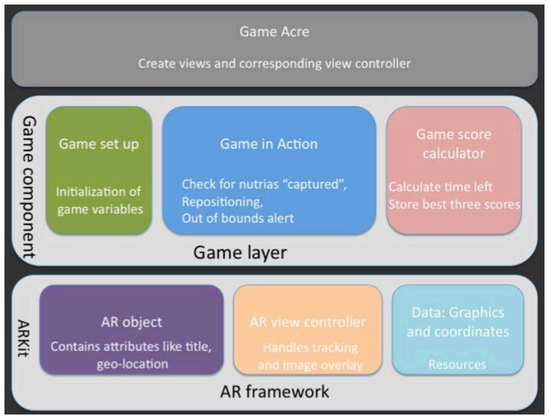
Game Acre
Create views and corresponding view controller
Game component
Game set up
Initialization of game variables
Game in Action
Check for nutrias "captured", Repositioning, Out of bounds alert
Game score calculator
Calculate time left
Store best three scores
Game layer
ARKit
AR object
Contains attributes like title, geo-location
AR view controller
Handles tracking and image overlay
Data: Graphics and coordinates
Resources
AR framework

Figure 6. Component based view of the architecture framework.

Game Acre is the first part of a larger effort within our research center for developing engaging, massive-scale applications on distributed cyberinfrastructures. Lessons learned from the Game Acre implementation will feed into this larger research program.

5. CHALLENGES FOR AN AUGMENTED REALITY APPLICATION WITHIN MOBILE DEVICE

In this section we detail the practical challenges of developing AR applications. We categorize these challenges into three parts:

computational challenges, sensor limitations and development and deployment platform challenges.

5.1 Computation Challenges

Real-time pose estimation of view port is computationally demanding

AR is all about displaying information at the right time and at the right place. This means that with the change of position, orientation and heading, image transformation and/or rendering must be reflected on the camera preview almost instantly (i.e. in real time). Therefore, computation of the view port must be done almost every fraction of a second, in order that the field of view stays synchronized with the phone's orientation. Note, however, in reality, the sensory inputs from the phone accelerometer and magnetometer are constantly changing by small values even if the user is not moving physically. This is because, it is almost impossible for the user to hold a phone (or any object) completely still. In addition, noise inputs from the surrounding environment introduce errors in the readings of the sensors. Thus, even as the computation must be carried out frequently, care must be taken to filter out sensor noise. Tracking for updates in the position, orientation and heading must be quick enough for the user to not be able to detect jitter in the images overlaid on the view from the camera. The optimal value of the calibration frequency should be maintained such that the real time image placement and removal will appear smooth, minimizing potential errors.

All of this must be done while conserving computation for other application tasks, such as fetching information from the servers. One way to minimize computation is by tailoring the speed of viewport recalculation. Note that the upper bound of the view port recalculation rate is the frame rate (frames per second – fps) of the phone's camera. Setting the update checks to any frequency higher than the frame rate is a clear wastage of CPU cycles because any change will not be reflected on the camera view due to the slower speed of the camera.

Next, the refresh rate may also be no faster than the speed of human perception. Note once again that the camera on iPhone 4G has a frequency of 30 fps. Hence, the upper bound of the update frequency in Game Acre is 1/30 second. The optimal frequency was then obtained through a small experimental study. Users were asked to play Game Acre at different refresh frequencies. The frequency levels ranged from 1 sec to 1/30 of a second (i.e. the maximum). The users were asked to select the minimum comfortable level at which they experienced no jitter on the camera view. The result obtained through the experiment was 1/20 sec. This period of time relates to the principle of perception of motion, where the brain needs to comprehend reality rather than just the persistence of an image [14]. Using these two criteria, we set the optimal frequency of the refresh rate so that CPU cycles were not wasted.

Synchronization problems arise if several timers are used to fire events that share the same variables

Timers are generally used to trigger events (at intervals) that are not generated by user interaction. In game applications, there are typically multiple actions that must occur at different times. In Game Acre, an image had to be set to fade after every 5 minutes, and an alert message had to pop up every 7 minutes. A separate timer must be used for each periodic event. Each timer executes on a separate thread; hence when several timers are used, multiple threads are forked. The use of several timers often led to synchronization issues and race conditions when actions triggered by several timers manipulated the same resources/variables. Two threads modifying the same resource can affect each other in unintended ways [17] that are hard to track and debug. Also, addressing the errors post facto usually requires a significant overhaul of the underlying coding assumptions.

As with any synchronization problem, the solution was to identify critical sections in the code and use in-built or user-defined synchronization functions to prevent inconsistencies. In addition it was important to check the values of variables and their states before performing any operations on them. Certainly, the most important step in assuring thread safety was in having a good program design that minimized the number of interactions between shared resources.

In the guidelines for thread safe design from Apple, one is asked to find the right balance between safety and performance. For example, the code could be designed to avoid threads altogether [17], or by having each task manipulate only its own local copy of data so that synchronization would not be necessary. Note that using locks and other synchronization techniques can also impede performance. If there is a high contention among the threads for a specific resource then the threads may end up waiting for a long time before they can perform any useful actions. On the other hand, creating local copies of data has storage overhead as well as consistency issues.

In Game Acre application, three timers were used. Each time a timer was fired, the corresponding thread manipulated the shared variables holding the virtual object's location and the image overlays. To resolve a class of synchronization challenges, we moved a set of computations that did not require local user information but were intended to realize collaboration between users (such as services that dealt with the location and capture state of the nutria), to the server. In Game Acre, the client is now only responsible for requesting information about the AR content from the server, and then rendering it. The client also has the functionality to *cache* previously rendered situations. Thus, if the mobile device only moved within a limited distance, say, a meter, the rendering is not redone; instead, cached screen renderings are reused without decreasing the quality of game experience.

5.2 Sensor Capability Challenges

Geomagnetic sensor noise causes jitter and inaccurate positioning of virtual objects

The magnetic sensor (compass) provided by iPhone is similar to a magnetic needle in a compass. The accuracy of this sensor can be affected by stray magnetic fields in the surrounding environment. These "noisy" inputs may come from nearby computers, television monitors, and other electronics components [16]. There are three common sources of magnetic fields, namely, permanent ferro-magnets (for example, those in speakers and buzzers), induced fields within ferro-magnetic materials (such as sheet steel) and electromagnetic fields (usually generated by electric circuits) [2]. This magnetic interference causes severe problems for AR applications that rely on detecting orientation using the magnetometer. The lowest value of horizontal magnetic field strength to be detected by a smartphone compass is 10μT (in northern Canada and in Russia) [2]. Thus, for an accuracy of 0.05 radians or 3 degrees in compass heading readings, the tolerable error in estimating geomagnetic field is no more than 0.5μT.

Although the calibration software running within any e-compass is able to remove interference to achieve accuracy better than 0.5μT, it is still not error-free enough to provide accurate readings (not to mention that these are also computationally intensive, especially when run continuously, because they use regression and other statistical methods). Standard algorithms used to remove noise interference are pattern recognition and smoothing filters [15]. A lightweight and portable algorithm has been proposed in the paper [3]. It is an extension of Finite Impulse Response filters with added statistical analysis for data exclusion and outlier analysis. The algorithm uses statistical methods that are computationally expensive during the initialization phase but may be optimized henceforth to run in constant time by keeping track of only the current sum and variation.

Delay in geolocation determination by GPS displaces the central point of reference
GPS satellites continuously transmit information (its location and current time) to the earth's surface. The GPS receiver on obtaining information from different satellites performs a calibration to determine the user's location. As a result of this process, GPS devices normally take about a minute or two to get the location from satellites. Thus, any application that depends on GPS must deal with an initialization delay before the correct location of the user is found.

We addressed this problem in the Game Acre application by making a call to get the user's location in advance (at the start of the application). This solution was possible because the spherical coordinates were required only for the camera view in the application. Having the correct location at the launch of the camera view was important for Game Acre application because the user's spherical coordinates were used as the origin around which the nutria were populated. The above solution prevented erroneous out of bounds (i.e. outside game field) reports, as well as application jitter.

Altitude problem
The GPS takes even longer to return the correct altitude value. As a result of this the frame of reference is disrupted. The iPhone uses three ways to find the user's location: local Wi-Fi signals, cell towers and satellites. However, the altitude value may only be obtained from the GPS satellite. If there is a strong signal reception then the altitude value may be returned quickly but if the signal is weak, the phone may not find the value until the satellite signal has been received several times.

This challenge was addressed by noting that the application characteristics were such that the altitudes of the users were only slow time varying. Thus, once an altitude value is calculated it is cached for reuse until the location manager returns an updated altitude. Note that when the altitude was required merely for pose estimation of the device, the application simply used zero as the datum and added to each virtual object a height that is proportional to its distance from user's current location. We chose distance-height relationship in Game Acre to achieve an elevated view of virtual objects so that complete overlapping of object would be avoided.

In general, sensor accuracy will be improved if we can consolidate information obtained from multiple resources. For instance, Wi-Fi hotspots, signal towers, caches of previous values, accelerometer and compass readings to estimate distances, by dead reckoning, can be used to calculate a possible locus.

5.3 Development and Deployment Challenges

Debugging and testing
The testing and debugging of applications that use the GPS can be a challenge because such applications are not able to test the code that handles location updates unless the application is in actual use, and the user is moving around. The iPhone emulator is not of much use because it does not provide support for hardware features. Applications can be deployed on the phone directly in the debug mode; but it is still a problem because the debugger tool requires the phone to be plugged into the computer at all times.

In order to overcome this hurdle, we needed a module that will simulate GPS inputs to the application without actual physical movement. FTLocationSimulator [18] is an open source project that may be used to simulate core location updates in the iPhone simulator. We recorded movement from game initialization to completion, and used these recordings to generate traces.

Long periods of inactivity cause the phone to sleep
Since the main user interaction in an AR application is observation through the camera, it is essential for the proper functioning of an AR application. However, when the digital cameras are held open for a long time, the device automatically goes into sleep mode *because observation through the camera is not recognized as user interaction with the application* by the device's operating system. Further, before the smartphone goes into the sleep mode and auto-locks the screen, it takes a snapshot of the state of all the running application, so when the user unlocks the phone the same application resumes. For some applications, this default feature is simply undesirable.

Game Acre is an application in which time plays an important role. The game rule requires the player to catch all nutrias as fast as they can. The default nature of auto locking is not desired here as it could frustrate the player who is close to winning a game, or worse, lead to potential misuse of the system by a player. A player could intentionally run to the approximate location of the next nutria to be grabbed while the phone is in sleep mode thus saving time and falsely gaining a higher score. Forcing the device to remain ON throughout the lifetime of Game Acre is also unpractical, because a major power conservation tactic is no longer available. Therefore, we programmed a server side application to detect whether a user is active. As long as the server receives significant device movements, it will send out orders to force the mobile phone to be ON. If within a certain amount of time the mobile device has no apparent activities, the server will set the mobile device's mode back to normal.

6. CHARACTERISTICS OF AUGMENTED REALITY FRAMEWORK

We close this paper with a discussion on the requirements of an AR framework – in particular those beyond the services identified (i.e. ones that support only the functional requirements of an AR application) in the papers described in the related work. These requirements are as follows:

- *Real-time processing and data transfer*
  The framework must enable and support the need for AR applications to respond in real-time to changes in location, the application state and the device state (for e.g. the orientation and direction of the device). Caching and sharing snapshots of computation – such as of the virtual and real environment, sensor values at specific locations within the environment, and

so on - in order to save on re-computation. This is an essential capability that must be supported.

- *Manage users and roles and provide services that support the spectrum of AR user roles*
  The AR cyber-infrastructure framework must provide services and API for management, authoring and development.
- *Management of Heterogeneous Datasets and External Content Repositories*
  The AR framework must manage a geo-tagged internal content repository, generated by various users, plus external datasets, as well as relationships between datasets.
- *Support for debugging and testing*
  AR applications are notoriously hard to test and debug without some kind of simulated data generation or data capture and replay capability.
- *High Availability*
  Given that the AR framework will be used to develop applications for commercial use, high availability is a highly desirable requirement.

7. CONCLUSION

Augmented Reality technology has expanded its reach from research and professional-grade equipment to commodity handheld devices operated by lay consumers of technology. In this paper, several of the major technical challenges imposed by the use of a commodity mobile device are presented and discussed with a view of giving guidance to future AR application developers. We are developing an extensible cyber-infrastructure framework that supports the customization of an AR application as outlined in this paper. Requirements of this framework have been described in this paper, to serve as a guide to future work.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation's Industry-University Cooperative Research at the Centre for Experimental Research in Computer Systems at Georgia Tech Institute and The Ohio State University. The paper is based on a project sponsored by Songwhale Inc. The sponsor was very helpful and provided us a lot of resources for writing this paper.

REFERENCES

[1] Belimpasakis P, You Y and Selonen P. Enabling Rapid Creation of Content for Consumption in Mobile Augmented Reality. *Mixed Reality Solutions, Nokia Research Center, Finland. IEEE 2010.*

[2] Ozyagcilar T. Layout Recommendations for PCBs Using a Magnetometer Sensor. *Freescale Semiconductor. Application Note. May 2011.*

[3] Gotow JB, Zienkiewicz K, White J, and Schmidt DC. Addressing Challenges with Augmented Reality Applications on Smartphones. Vanderbilt University, Nashville, TN USA. *Mobilware 2010.*

[4] Schmalstieg D, Langlotz T, and Billinghurst M. Augmented Reality 2.0. *Virtual Realities 2011.*

[5] Bauer M, Bruegge B, Klinker G, MacWilliams A, Reicher T, Rib S, Sandor C and Wagner M. Design of a Component–Based Augmented Reality Framework. In *Proceedings of the IEEE and ACM International Symposium on Augmented ISAR 2001.*

[6] Kato H. and Billinghurst M., Marker Tracking and HMD. Calibration for a video-based Augmented Reality Conferencing System. In *Proc. of the 2nd International Workshop on Augmented Reality (IWAR 99),* pp. 85-94, USA, 1999.

[7] Szalavari S, Schmalstieg D, Fuhramann D, and Gervautz M. "Studierstube" - An Environment for Collaboration in Augmented Reality. *Virtual Reality - Systems Development and Applications.* Vol. 3, No. 1, pp. 37–49. Springer, New York 1998.

[8] Gennari J, Harrington M, Hughes S, Manojlovich M, and Spring M. Preparatory Observations Ubiquitous Knowledge Environments: The Cyberinfrastructure Information Ether. June 2003.

[9] Atkins DE. Cyberinfrastructure And The Next Wave Of Collaboration. In *Proceedings of EDUCAUSE Australasia, Auckland, New Zealand,* April 5-8, 2005.

[10] Schmalstieg D and Reitmayr G. The World as a User Interface: Augmented Reality for Ubiquitous Computing. *Central European Multimedia and Virtual Reality Conference,* 2005.

[11] Schmalstieg D and Wagner D. Mobile Phones as a Platform for Augmented Reality. Graz University of Technology. *Symposium A Quarterly Journal In Modern Foreign Literatures* (2008), Vol 1, pp. 43-44.

[12] D. Mizell. Augmented Reality Applications in Aerospace. In *Proceedings of ISAR 2000*, Munich, 2000.

[13] Marimon D, Sarasua C, Carrasco P, Álvarez R, Montesa J, Adamek T, Romero I, Ortega M and Gascó P. *MobiAR: Tourist Experiences through Mobile Augmented Reality.* Telefonica Research and Development, Barcelona, Spain; Visual Interaction Communication Technologies Vicomtech-IK4, San Sebastian, Spain; LabHuman, Valencia, Spain, Indra Software Labs, Madrid, Spain, Brainstorm, Valencia, Spain; 2010.

[14] Wertheimer M. Experimentelle Studien über das Sehen von Bewegung. *Zeitschrift für Psychologie 61*, pp. 161–265. 1912.

[15] Savitzky A, Golay M.J.E.. "Smoothing and Differentiation of Data by Simplified Least Squares Procedures". Analytical Chemistry, 1964.

[16] Milgram P, Kishino F. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transaction on Information Systems,* Vol E77-D, No.12 December 1994.

[17] http://developer.apple.comlibrary/mac/#documentation/Cocoa/Conceptual/Multithreading/ThreadSafety/ThreadSafety.html

[18] https://github.com/futuretap/FTLocationSimulator