CloudRidAR: A Cloud-based Architecture for Mobile Augmented Reality

Zhanpeng Huang† Weikai Li† Pan Hui† Christoph Peylo‡
† HKUST-DT System and Media Laboratory,
Hong Kong University of Science and Technology, Hong Kong
{soaroc, weikaili, panhui}@ust.hk
‡ Telekom Innovation Laboratories, Berlin, Germany
christoph.peylo@telelcom.de

ABSTRACT

Mobile augmented reality (MAR) has exploded in popularity on mobile devices in various fields. However, building a MAR application from scratch on mobile devices is complicated and time-consuming. In this paper, we propose CloudRidAR, a framework for MAR developers to facilitate the development, deployment, and maintenance of MAR applications with little effort. Despite of advance in mobile devices as a computing platform, their performance for MAR applications is still very limited due to the poor computing capability of mobile devices. In order to alleviate the problem, our CloudRidAR is designed with cloud computing at the core. Computational intensive tasks are offloaded on cloud to accelerate computation in order to guarantee runtime performance. We also present two MAR applications built on CloudRidAR to evaluate our design.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Augmented and Virtual Realities

General Terms

Performance, Design, Algorithm

Keywords

Mobile Augmented Reality, Cloud Computing, Mobile Platform, Software Architecture

1. INTRODUCTION

With the rapid development of mobile devices comes the ability to create new experience that enhances the way we acquire, interact with and display information within the world that surrounds us. Mobile technology improvement in built-in camera, embedded sensors, computational resources and power of cloud-sourced information has enabled augmented reality (AR) on mobile devices. We are able to blend information from our senses and mobile devices in myriad ways that were not possible before. MAR has the potential to allow users to interact with information without getting distracted from the real world. The way to supplement the real world other than to replace real world with an artificial environment makes it especially preferable for applications such as tourism, navigation, entertainment, advertisement, and education. MAR is widely regarded as one of the most promising technologies in the next ten years. The mobile advertising market for MAR-based apps is estimated up to $732 million by 2014 [6] and MAR technology within mobile application will lead to almost 1.4 billion annual downloads worldwide by 2015 [7].

Most MAR applications are task-centered and well-structured. These monolithic and highly specialized applications normally choose specified architectures that are so much different. Hence, it is quite difficult to reuse the technologies in these architectures even many basic functional modules, such as tracking and rendering, are essentially the same. It is important to design an architecture that can be used for a variety of different MAR applications. Several MAR software frameworks have been developed to reduce developer's workload, but these frameworks have several drawbacks including low performance, hardware platform incompatibility, and poor flexibility. In addition, despite the advance in mobile devices as a computing platform, the performance for real-time application is still very limited. MAR applications are still constrained to the poor computational capability of mobile devices. These software frameworks cannot fully leverage the advantage of emerging cloud computing technology, making them ill-suited for increasingly computational intensive and power-hungry MAR applications.

In this paper, we propose CloudRidAR, a cloud-based mobile augmented reality framework for prototyping, development, and deployment of various MAR applications with little effort. The framework is designed with consideration of requirements from perspective of both application and framework developers. It enables developers to focus on high-level application logic rather than low-level implementation. To obtain best runtime performance with limited computing capability on local mobile devices, we integrate the cloud computing with our framework. Computational intensive tasks are outsourced to the cloud for computing acceleration and runtime performance improvement.

MobiSys 2014 Workshop for Mobile Augmented Reality and Robotics-based technology systems Bretton Woods, NH, USA

2. RELATED WORKS

Numerous MAR applications have been developed for various purposes. Columbia University built several prototypes for MAR-supported tourism [8]. Information including labels of buildings, iconic flags for important information, and virtual presentation of former buildings on the sites that have been demolished are superposed on users' current view. Elmqvist et al. [3] proposed a three-dimensional virtual navigation application to support both path finding and local features hinting. Hand gesture is used to facilitate interaction with virtual world. To improve user experience in computer games, Piekarski and Thomas [18] proposed MAR ARQuake game. The game enable users to play virtual games in physical world. Players are able to walk around outside and kill virtual 3D monsters with physical props. MAR is also used in training and education applications. The Naval Research Lab (NRL) developed a Battlefield training system [11] to train soldiers for military operations in different environments. The battlefields are augmented with virtual 3D goals and hazards that can be authored and deployed beforehand or dynastically at run time. Freitas and Campos [4] developed SMART, an education system to teach low-level grade students. Virtual 3D models such as car and airplane are overlaid on real time video to demonstrate concepts of transportation. MAR is also widely used for assembly and maintenance purposes. Billinghurst et al. [2] proposed a MAR assembly system to help users refer to a 3D view of step-by-step guidance for an assembly task in real world. A virtual 3D view of next step is overlaid on current view so that users can decide which part to use and where to place it in the next step. Henderson and Feiner [5] developed a prototype to track users and components in a maintenance task. Dynamic and prescriptive instructions are overlaid on HMD to response to users' activity. Virtual 3D arrows, labels, and aligning dash lines are displayed to facilitate the operations. So many MAR applications are developed during past years. Interested readers are referred to detailed surveys [9].

It is complicated and time-consuming to build a MAR system from scratch. Most MAR systems share several basic functions such as tracking, graphical rendering, and human-computer interactions. Many efforts have been directed towards MAR architecture to help developers relieve their workload. Reitmayr and Schmalstieg developed Studierstube ES [19], an AR framework that includes several modules such as OpenTracker for tracking, Muddleware for client-server communication, and APRL for authoring. However, the framework is only supported by Windows and Android phones. Moreover, it is not available for the public. KHARMA [10] is a MAR framework based on KML, a variation of XML used to describe geo-referenced multimedia. The framework is only suitable for development of geospatial MAR applications. Piekarski and Bruce [17] proposed an object-oriented framework named Tinmith-evo5. Data flow is divided into several layers with sensor data as input and display device as output. Layers are abstracted as class objects. All objects in the system are allocated in an object repository to support persistent storage and run-time configuration. The framework does not support handheld computers yet. AndAR [1] is an open project to enable MAR on Android platforms, but it is only tested on very few mobile phones and the project has ceased since 2010. Metaio [14] is a commercial toolkit that has been used to development many MAR applications. It includes a Metaio Cloud module to support apps and contents storage on cloud, but it can not use cloud for computation acceleration and developers cannot control where their tasks to run. ARtoolkit [12] is a widely used tracking tool for AR/MAR applications, but it is software library rather than a framework. There are also several commercial AR glasses shipped with development frameworks such that are specifically tailored for their products. Most frameworks run on local mobile devices. Some are built based on client-server architecture, but the server is only used as database to store data.

Multitier model is widely adopted in the design of MAR. Schmalstieg et al. [20] used 3-tier model in their MAR system. The first tier is a database. The second tier links the database and applications. It converts raw data to the third tier where applications reside. Nicklas and Mitschang [16] divided the system into a client device tier, a federation tier and a server tier. The federation tier connects the server tier to the client tier using a register mechanism. Tonnis [21] adopted a 4-tier model with bottom tier providing connectivity and communication services, second tier including basic functional components for MAR, third tier with high-level modules comprised of basic components, and top layer responsible for user interaction.

3. ARCHITECTURE DESIGN

Our primary goal is to develop a both research-oriented and application-oriented platform for MAR applications. In addition to research purpose, we follow Mizell's idea [15] to bring well-established MAR technologies into practise with a few assumptions: 1) A prosperous MAR market in next few years will come up with massive MAR applications; 2) Wireless network continues to improve on speed and broadband connectivity; 3) Cloud infrastructure and service providers continue to deploy innovative services at low cost. 4) Apart from cloud computing resource, we only utilize components on mobile devices to make the application self-contained and little intrusive. With these assumptions we try to achieve following objects in our CloudRidAR framework:

1. Easy to use for developers: CloudRidAR includes ready-to-use functional modules that provide uniform and brief interfaces for developers. Predefined task flow simplifies workload and dynamical registration mechanism guarantees flexibility to any changes.

2. Real-time performance for users: Based on cloud architecture, computational intensive tasks are outsourced to cloud for computing acceleration. It also eliminates harsh requirements of hardware and software on mobile devices. A side benefit using cloud computing is to save power energy and prolong battery lifetime.

3. Scalable across different mobile platforms: CloudRidAR introduces an abstract layer to encapsulate diverse hardware on mobile devices. Upper modules are built on hardware abstraction layer to make applications portable across various mobile platforms.

For framework developers, we parallelize development tasks to reduce both cost and time, but it requires that parallel tasks should not be overlaid and coupled. We adapt component design principles to realize it. Component technology is widely used in software design to improve reusability

and integration of local and external resources. Framework breaks down into individual components with well-defined interfaces according to MAR functional requirements. As components and their interface are defined, they can be implemented in a parallel way. We can even make it more flexible by combining different components to meet various scenarios as long as we design them as general as possible.

From application developers' point of view, they need to take care of both the low-level implementation and high-level application logic if they have to build a MAR application from scratch. Our framework should be able to encapsulate the low-level functional units that developers can directly reuse to define high-level logic in a flexible way. We group the components according to reusable elasticity. The least elastic components are common functional units including tracking, display, and data storage, which are essential parts and reusable for all MAR applications. The less elastic are components to comprise the basic workflow of a typical MAR applications and user interface. The most elastic components are those to define application logic. Application logic heavily depends on application scenarios. For instance, application logics of a maintenance system and a tourism system are apparently different.

The reusable components flexibly connect to compose three high-level layers: hardware abstraction layer, basic flow layer, and application container as illustrated in Figure 1. We will give full descriptions of each layer in the following section.

4. SYSTEM IMPLEMENTATION

4.1 Hardware Abstraction

A typical MAR application is a combination of several necessary external hardware such as computing resources, tracking devices, and display devices. These hardware components vary with different mobile platforms. For instance, mobile devices may have display screens with different size and scale. A fixed field of view may cause rendering distortion. Several mobile devices do not support native floating-point computing. We have to use software emulation to guarantee numerical accuracy. Traditional MAR applications built on specific hardware platform make them failure on other platforms. A hardware abstraction layer is required to make applications independent on hardware and portable to various hardware platforms.

It is difficult to abstract hardware because physical limitations, such as resolution of sensors and temporal update frequency, may vary with different hardware, not to mention the rapid change of hardware over time. Many hardware components are not standardized yet in MAR systems, so we adopt a least common denominator model for each type of hardware with several common interface. For instance, various sensors including gyro, accelerator and magnetometer are used for tracking purpose in MAR. Although the sensors may have different characteristics such as resolutions, update frequencies, measurement ranges and drift, an abstraction to encapsulate tracking sensors may only requires a few interfaces of initialization, calibration, and data acquisition. Upper components require orientation and position information from tracking sensors, so that the data acquisition interface should be able to return pose information. Each hardware device has a XML-based configuration file to describe its characteristics and functions. The configuration file can be modified to enable our framework scalable on emerging hardware. A typical hardware configuration file is listed in Figure 2. Users can define additional specifications for new hardware devices. They are able to call specific functions equipped with hardware driver using predefined function prototypes.

```
<hardware type = "ACCELERATOR" id="acc1" >
  <specification>
    <accuracy> "0.0015 0.0015 0.0015" </accuracy>
    <resolution> "0.001 0.001 0.001" </resolution>
    <update rate> "300" </update rate>
    <lag> "0.03" </lag>
  </specification>
  <output id = "velocity" type = "vector3" default = "0 0 0">
  <output id = "acceleration" type = "vector3" default = "0 0 0">
  <interface>
    <function name="init" parameter = "void" output = "void"></function>
  </interface>
</hardware>
```

Figure 2: A XML-based hardware description file.

In addition to physical hardware devices, we supply a uniform interface to access various computing resource including local mobile CPU, mobile GPU, and cloud. We integrate our previous mobile cloud framework ThinkAir [13] to provide code offloading and dynamic task allocation services. Developers only need to add an annotation to indicate which parts could be offloaded if cloud is available. ThinkAir dynamically allocates annotated tasks on cloud for computing acceleration, which is totally transparent to developers.

4.2 Data-Driven Flow

A typical MAR application includes several subroutines such as tracking, rendering, and user interaction. These subroutines works together to complete the task. Compared to traditional MAR workflows, we employ ThinkAir to dynamically allocate tasks to different computing resources, which requires synchronization between offloading threads and the main thread in workflow. Our MAR flow for CloudRidAR framework is designed as shown in Figure 3. In initialization

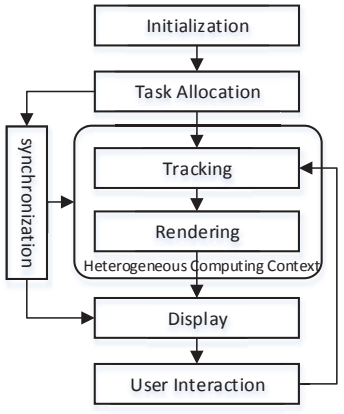

Figure 3: A brief workflow of our framework.

step, each registered hardware will automatically call predefined function in its hardware description file (see example in Figure 2) to initialize its status. The task allocation mechanism decides whether to run most computational intensive

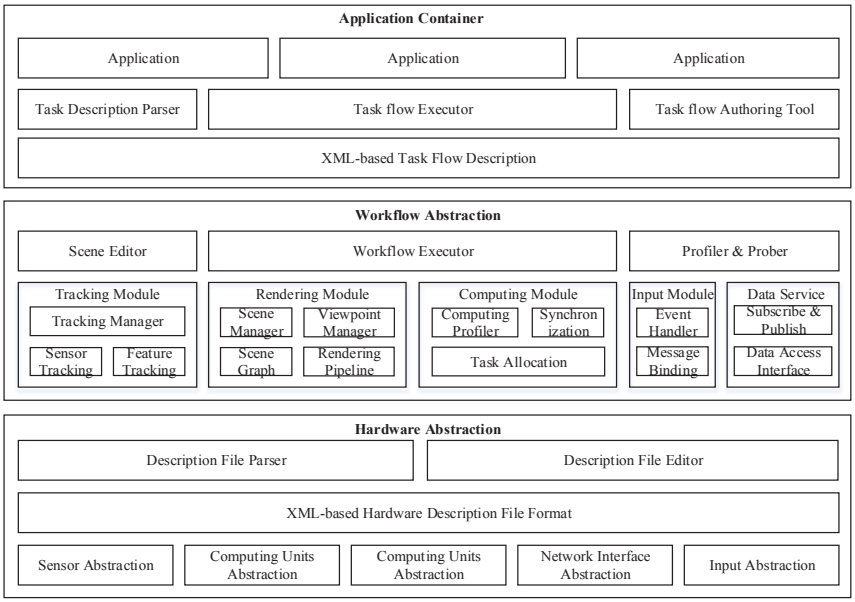

Figure 1: The software stack of our framework.

tasks on local device or remote cloud. As the system runs in a distributed computing environment, all outsourced tasks should be synchronized so that the next step can start only if all outsourced tasks have finished. Rendering results are mixed with real contents and ready for showing on display devices, which users can use to interact with the system in various ways such as gesture and voice control.

The basic workflow is abstracted as a data-driven engine, which is the representation of a sequence of actions that will be executed on data flow in predefined order. The data flow to drive the entire workflow includes pose information, virtual contents, and user inputs. Although the basic pipeline is determined, procedures in the workflow are flexible to configure to meet users' requirements. For instance, sensor-based and feature-based tracking are two typical tracking technologies for MAR applications. Generally speaking, feature-based tracking obtains higher accurate results at the cost of much heavier computational overhead comparing to sensor-based tracking. We integrate both tracking technologies in CloudRidAR framework. In addition, the feature-based implementation is able to be offloaded to cloud in order to guarantee runtime performance. The framework is responsible for underlying implementation and users just need to choose suitable version in terms of accuracy and runtime performance requirements and availability of cloud service for their applications. In user interaction implementation, CloudRidAR includes several input interfaces such as screen touch, gesture capture, and voice control. For gesture capture, the framework provides functions to recognize various hand gestures, which can be mapped to trigger actions predefined by users.

4.3 Application container

The application container is designed to be a run-time context that is tightly coupled with the application logic. As many MAR applications such as navigation, assembly, maintenance, training, and game are task-driven, we can employ abstract design method to reduce users' workload to develop applications for different purposes.

The task-driven applications can be abstracted as a finite state machine (FSM). Application is switched from current state to a new state when transition condition is satisfied. For instance, in a typical assembly system, a user should finish assembling current component in order to start next component. The user may have several predefined sequences to finish the task, which produces many transient states that can be transited to another state. A simply flow can be represent as FSM illustrated in Figure 4. Applications are abstracted as sequence of states connected by transition conditions. As applications have different intermediate states, it is up to users to define the application states. The application state includes the transition conditions and event handlers that can respond to user input and fire other events. A state may also have data sheet required to be displayed when the application transits into the state. The application container parses users' definition of application states and executes them according to predefined transient conditions. It also

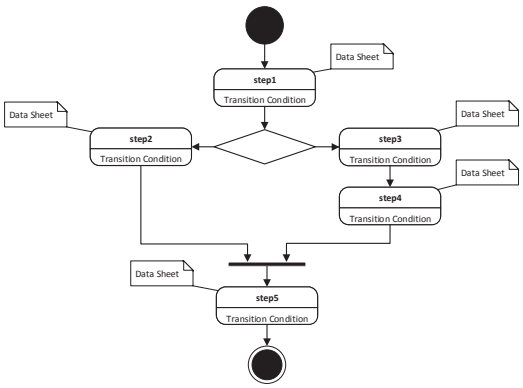

Figure 4: A simple flow represented with FSM.

displays any information contained in the states.

As application logic is separated from execution and display, it enables rapid development of applications with our framework. Dynamically parsing of the states allows applications to adapt to different scenarios and even change the task flow at runtime.

5. EXPERIMENTS AND RESULTS

We developed two MAR prototypes based on CloudRidAR. Attributed to basic functions provided by our framework, we are able to develop two prototypes within a few hours. Our prototypes are built on SAMSUNG Galaxy Nexus with 1.2GHz PowerVR SGX540.

Our first prototype is a car running application as illustrated in Figure 5. A virtual car on mobile phone can run on a real path. It is an interactive game that users can plan path dynamically to control where and how the car runs. In

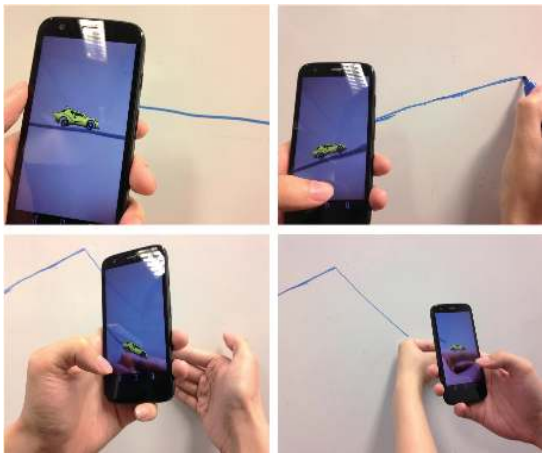

Figure 5: A prototype of virtual car running on real path.

this demonstration, the virtual car is driven along a real line drawn by user dynamically. It is also able to run on a user's hand. In fact, the path can be any line features extracted from view images of real world. As feature extraction requires heavy computational overhead, it normally cannot achieve real-time performance on mobile phones that lacks computing capability to finish the computation in a short period. By leveraging the cloud computing architecture, we offload the feature extraction on cloud for computing acceleration. The coordinates of recognized line features are sent back to mobile phone and assembled as geometry paths to guide car's movement. The real-time feature extraction on the cloud allows user to interact with car in an interactive way.

We also implement a collaborative MAR application. Multiple users are able to play the game in a collaborative way. Players can use any physical rectangle planes as bats to hit the virtual Ping-Pong ball as illustrated Figure 6. In this prototype, we are required to track pose information of both players and "bats". Players' information is estimated by tracking position and orientation of mobile devices. The virtual Ping-Pong ball is rendering from players' current poses so that players can see the virtual ball from current viewpoints on their own mobile phones. Pose information of physical bats are tracked to locate pre-created 3D geometry

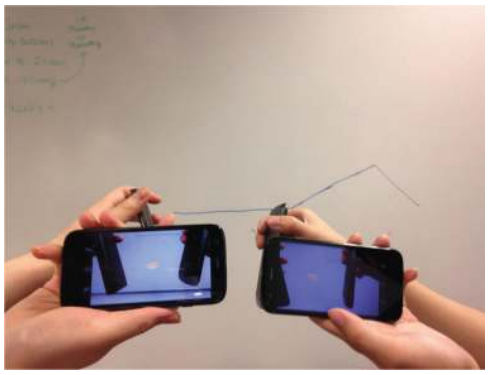

Figure 6: A collaborative Ping-Pong game.

objects in virtual world. The virtual geometry objects are only used for collision detection with virtual ball to simulate the hitting and rebounding of Ping-Pong ball with bats. As geometry objects are not required for visual rendering, we only create coarse models to represent the bounding boxes of physical bats. To simulate motions of virtual Ping-Pong in a real way, we estimate velocity of physical bats and calculate the rebounding velocity of virtual ball using the kinetics and momentum conservation principle. As collision detection and physical rigid body simulation are time-consuming, we outsource both tasks to cloud in order to improve runtime performance. The position and velocity information of virtual Ping-Pong ball are sent back for animation and rendering. As both local applications request the results from the same physical numerical simulator on the cloud, it guarantees spatial coordinate consistence of virtual ball among client applications. In the development of both applications, the CloudRidAR framework allows us to prototype the applications in a fast and direct ways. Even the application logics are totally different, we can still share several basic functions and services provided by our framework. In our original design, we did not consider about the physical simulation. We developed a simple physical engine for rigid body simulation, which cost us additional time for development. In the two applications, we employed feature tracking for pose estimation. Thanks to the powerful cloud computing capability, we are able to achieve real-time performance even on mobile phones. Figure 7 gives a compassion to show the time cost of tasks on mobile phone and cloud. Each application is comprised of several major tasks including rendering and feature tracking. Others are some trivial tasks such as camera access and user input response. The second application includes specific tasks of rigid body simulation. As shown in the figure, feature tracking is the major performance bottleneck on mobile device. By offloading the feature tracking task to the cloud, the total time cost drops greatly to guarantee real-time performance. In both prototypes, virtual content is relevant simple and does not require too much computation, so we implement the rendering task on mobile phone rather than cloud. In the second prototype, we run collision detection and rigid body simulation on the cloud.

6. CONCLUSION

In this paper we present CloudRidAR, a cloud-based framework for MAR application development. The framework is

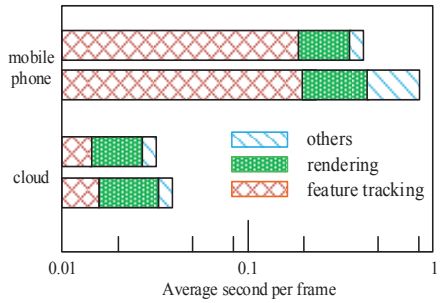

Figure 7: Tasks performance measurements on mobile device and cloud.

designed from a high-level viewpoint. It is comprised of three layers with abstraction of hardware, basic flow and application logic to facilitate development and deployment of MAR applications.

There are several modules that we will integrate into our CloudRidAR framework in future. In most MAR applications, virtual scenes are relative simple comparing to pure 3D applications such as video games and virtual world. As 3D rendering is less likely to be the performance bottleneck, it is reasonable to design a mobile 3D engine for local rendering. While we believe that a MAR framework for future applications may also require powerful rendering capability when large-scale environmental model of an entire city or big data from Internet of Things (IOT) is ready for use. In order to meet future requirement of intensive rendering, we will design a cloud rendering subsystem that leverages the cloud for rendering acceleration. As rendering and display are separated on cloud and local mobile device, we have to solve the problem of remote user interaction and rendering result retrieval. A feasible solution is to compress the rendering image on cloud and decompress it on local device for display. User interaction is parameterized, encoded, and uploaded to cloud for updating rendering. In addition, a scene editor will also be developed to help users design and deploy virtual scene in a what-you-see-is-what-you-get way.

Another future feature of CloudRidAR is authoring tool for task flow design. We will provide several most widely used task templates for users to implement their own application logics. Users can select a predefined template or design a new one with template to instantiate a task element. The task elements are then visually linked by corresponding interfaces to represent a task flow, which will be parsed and executed by task flow engine in runtime.

We will develop more MAR applications to evaluate our design in future. In addition, we can prototype and verify any new idea about MAR with our framework. CloudRidAR can serve as a useful foundation for both research-oriented and application-oriented MAR systems, which we believe will help to boost MAR applications in relevant fields.

7. REFERENCES

[1] AndAR. https://code.google.com/p/andar/, 2010.

[2] M. Billinghurst, M. Hakkarainen, and C. Wodward. Augmented assembly using a mobile phone. In *Proceedings of MUM*, pages 84–87, 2008.

[3] N. Elmqvist, D. Axblom, and J. Claesson. 3dvn: A mixed reality platform for mobile navigation assistance. In *Proc. CHI*, 2006.

[4] R. Freitas and P. Campos. Smart: a system of augmented reality for teaching 2nd grade students. In *Proceedings of British Computer Society Conference on Human-Computer Interaction*, pages 27–30, 2008.

[5] S. J. Henderson and S. K. Feiner. Augmented reality in the psychomotor phase of a procedural task. In *Proceedings of ISMAR*, pages 191–200, 2011.

[6] W. Holden. http://www.juniperresearch.com/viewpressrelease.php pr=166, 2009.

[7] W. Holden. http://www.juniperresearch.com/viewpressrelease.php pr=232, 2011.

[8] T. Hollerer, S. Feiner, and D. Hallaway. User interface management techniques for collaborative mobile augmented reality, computers and graphics. *Modeling and Simulation Design*, 25(5):799–810, 2001.

[9] Z. P. Huang, P. Hui, C. Peylo, and D. Chatzopoulos. Mobile augmented reality survey: A bottom-up approach. Technical report, HKUST Technical Report, 2013 (cite as arXiv:1309.4413).

[10] J. Irizarry, M. Gheisari, and G. Williams. Infospot: A mobile augmented reality method for accessing building information through a situation awareness approach. *J. Autom. Constr.*, 33:11–23, 2012.

[11] S. Julier, Y. Baillot, L. M, and D. Brown. Bars: Battlefield augmented reality system. In *NATO Information Systems Technology Panel Symposium on New Information Processing Techniques for Military Systems*, 2000.

[12] H. Kato, M. Billinghurst, R. Blanding, and R. May. Artoolkit, 1999.

[13] S. Kosta, A. Aucinas, and P. Hui. Thinkair: Dynamic resource allocation and parallel execution in cloud for mobile code offloading. In *Proceedings of INFOCOM*, pages 945–953, 2012.

[14] Metaio. http://www.metaio.com/, 2014.

[15] D. Mizen. Augmented reality applications in aerospace. In *Proceedings of ISAR*, 2000.

[16] D. Nicklas and B. Mitschang. A model-based, open architecture for mobile, spatially aware applications. In *Proceedings of Advances in Spatial and Temporal Databases*, pages 392–401, 2001.

[17] W. Piekarski and H. T. Bruce. An object-oriented software architecture for 3d mixed reality applications. In *Proceedings of ISMAR*, pages 247–256, 2003.

[18] W. Piekarski and B. H. Thomas. Arquake: The outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[19] G. Reitmayr and D. Schmalstieg. Collaborative augmented reality for outdoor navigation and information browsing. In *Proceedings of ymposium Location Based Services and Tele Cartography*, pages 53–62, 2004.

[20] D. Schmalstieg, G. Schall, and D. Wagner. Managing complex augmented reality models. *IEEE Computer Graphics and Applications*, 27(4):48–57, 2007.

[21] M. Tonnis. Data management for augmented reality applications. Master's thesis, Technische Universitat Munchen, 2003.