Lecture 8:
Location-based Mobile Augmented Reality Applications
Challenges, Examples, Lessons Learned

Philip Geiger, Marc Schickler, Rudiger Pryss, Johannes Schobel, Manfred Reichert

COMP 30025J

Dr. Abraham Campbell

Abstract

- Design and Implementation of a mobile AR application on both IOS and Android
- Lessons learnt and discussing implementing a mobile business application

6 Summary and Outlook

- Insights into development of the core AREA framework.
- AREA engine abilities outlined especially its locationview technique
- Modularity importance demonstrated by the LiveGuide integration.
- POI design issues highlighted such as when they are placed on top
- Indoor tracking requirements in the future.

1 Introduction

- New types of business application are possible with AR
- Mobile devices offer new abilities but developers need to cope with new design issues such as battery and low CPU resources.
- Maintain similar functionalities between different mobile OS's such as android and IOS

1.1 Problem statement

- The problem : Develop the core of a location-based Mobile Augmented reality engine for Mobile operating systems.
- This paper is structured more as a software engineering report
- Its structure should help you think about your final year project report next semester.

1.1 Problem statement : what challenges are addressed

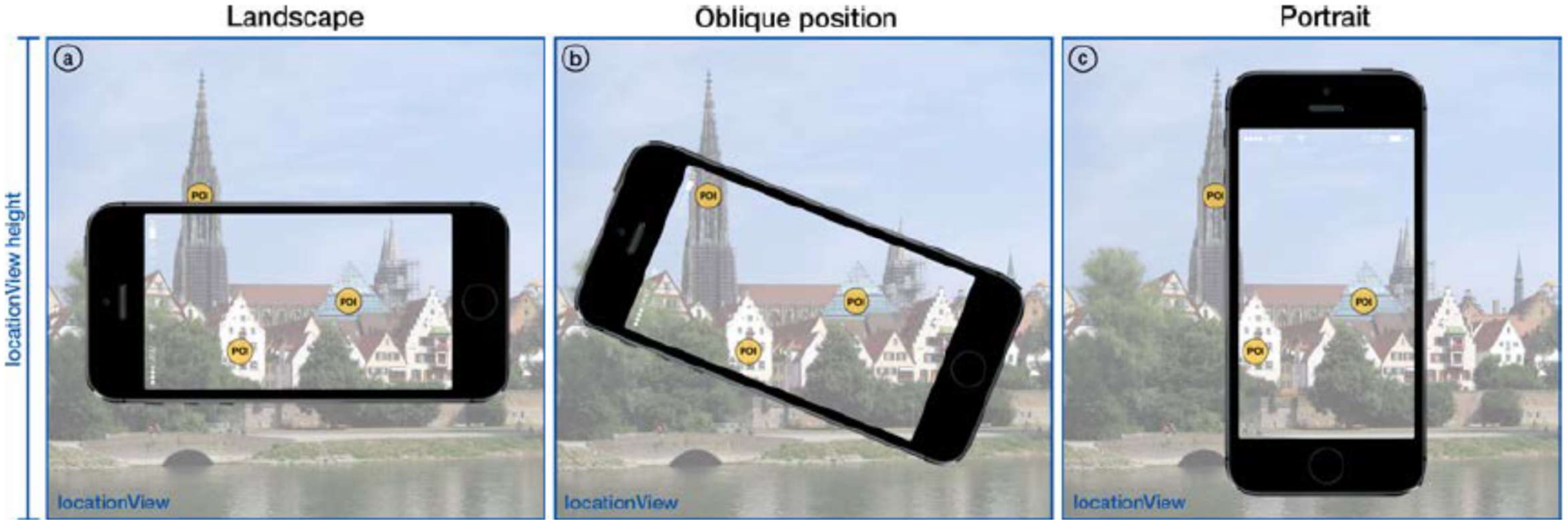
- In order to enrich the image captured by the smart mobile device's camera with virtual information about POIs in the surrounding, basic concepts enabling location-based calculations need to be developed.
- An efficient and reliable technique for calculating the distance between two positions is required (e.g., based on data of the GPS sensor in the context of outdoor location-based scenarios).
- Various sensors of the smart mobile device must be queried correctly in order to determine the attitude and position of the smart mobile device.
- The angle of view of the smart mobile device's camera lens must be calculated to display the virtual objects on the respective position of the camera view.

1.2 Contribution

- Creation of AREA engine discussed and outlined how it can work in multiple mobile operating systems.
- Section list gives an outline of what the contributions will be
  - This is well done in terms of forward and backword links
  - Section 2: Outlines the core concepts
  - Section 3: lessons learnt
  - Section 4: Implementing a business application with the engine
  - Section 5: Related work
  - Section 6: Conclusions

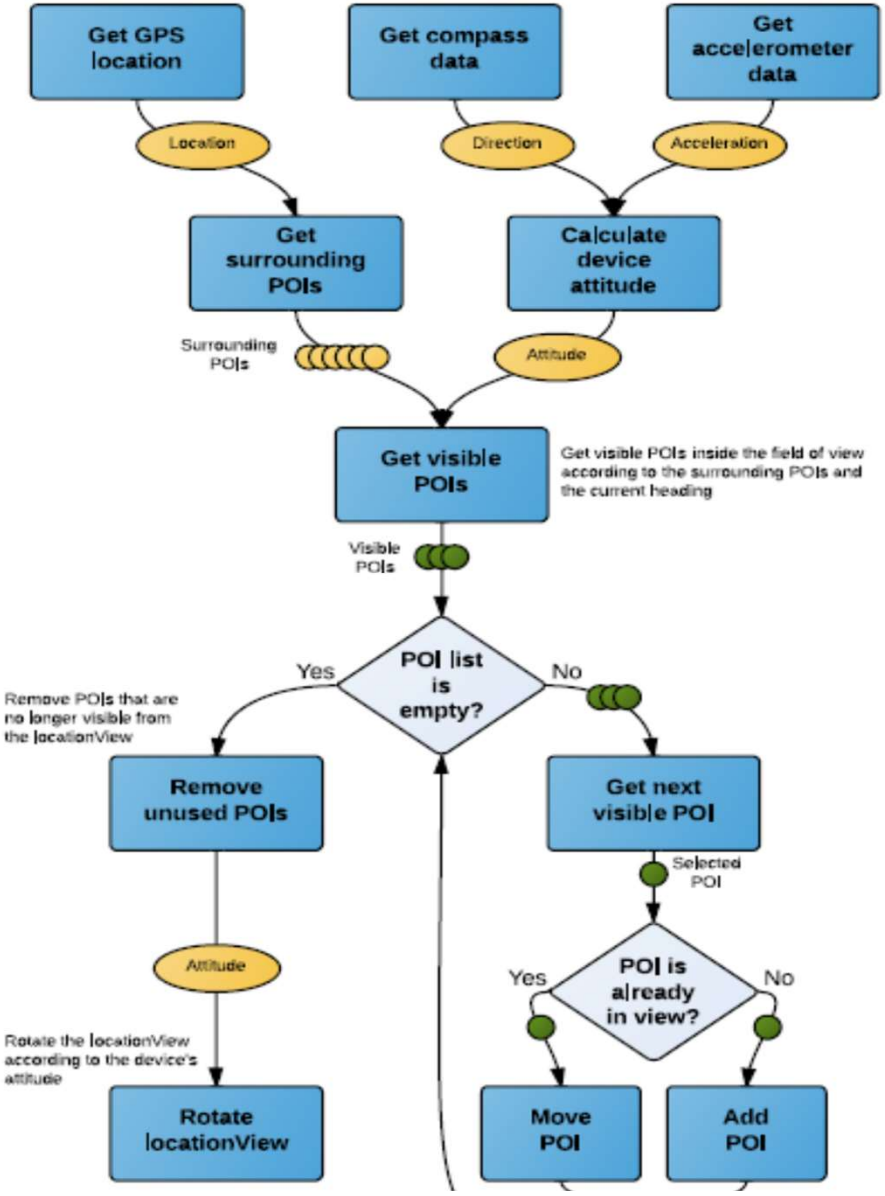
2 Area Approach and 2.1 The location View

- Point of interested displayed as yellow markers
- Area concept is based on $\sqrt{\text{width}^2+\text{height}^2}$



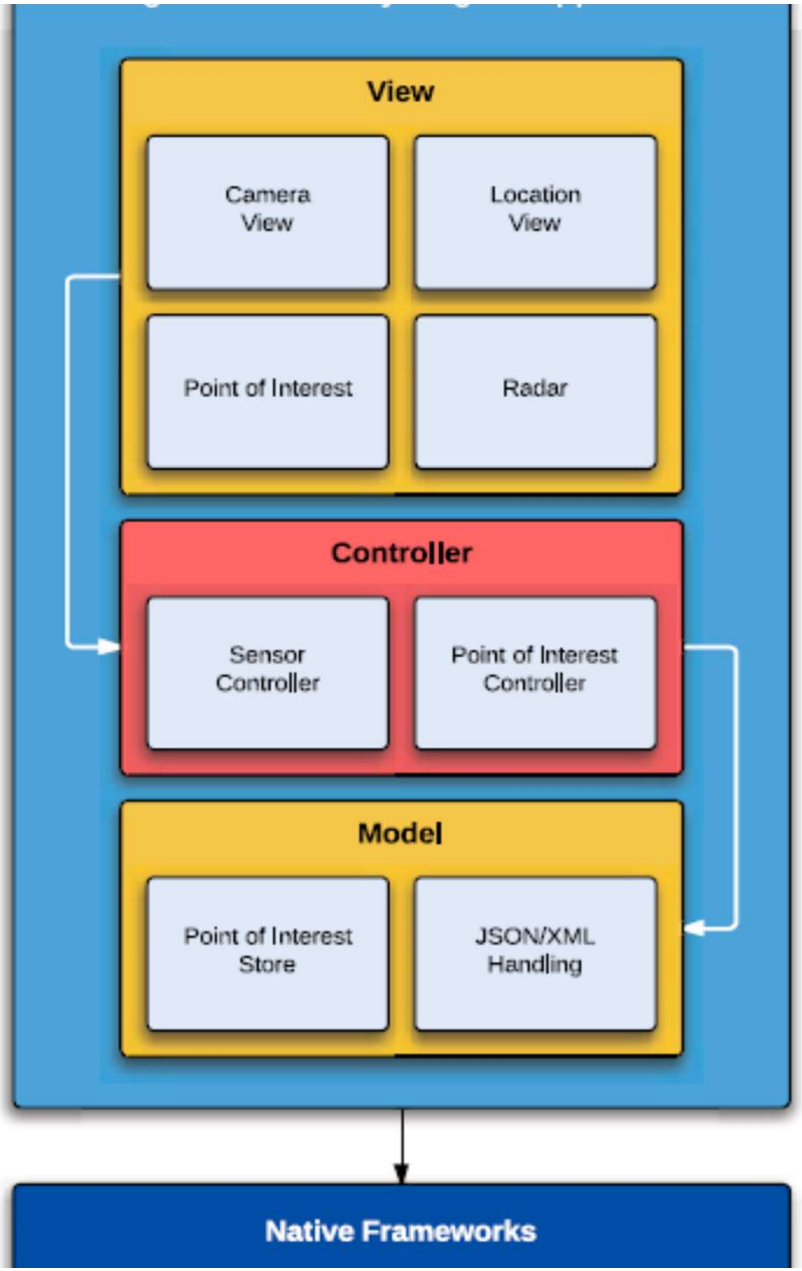
2.2 Architecture

- Model / View / Controller pattern used
- Model
  - POI defined in XML or JSON
  - ARML scheme used
- Controller
  - Sensor controller
  - Point of interest Controller
- View
  - Camera view
  - LocationView



3 EXPERIENCES WITH IMPLEMENTING AREA ON EXISTING MOBILE OPERATING SYSTEMS

- Implement the same system on different mobile platforms
- Using Phonegap would be too slow and lacked enough fine grain control so they required native implementations.



3.1 iOS MOBILE OPERATING SYSTEM & 3.1.1 Sensor Controller

- Xcode used
- Corelocation framework used
- Updates based on 10 meter differences
- CoreMotion framework used for accelerometer with low pass filters
- Constantly need to reconfigure true north to make sure the application adjust for any rotation.

3.1.2 Point of Interest Controller

- Sensor controller lead to two events being called in POI controller
  - Detecting a new location
  - Gathering a new heading as well as other accelerometer data being updated.
  - Using their LocationView concept , this information was then used to place POI markers on the screen.
  - Abstracting out the need for any changes to the screen due to rotation is due to relaying on the locationView concept.

3.2 ANDROID MOBILE OPERATING SYSTEM & 3.2.1 Sensor Controller

- Similar implementation but used android equivalents
- Android.location and android.hardware
- Android implementation uses rotation matrix to achieve the same effect (Also the right way of doing it in my opinion)
- Sensors on android are polled rather than pushed as in the IOS design philosophy. (Again from an energy saving consumption perspective a much better approach)

3.2.2 Point of Interest Controller

- Layouts on android can actually be rotated so this paper is incorrect, but it requires some very nasty native code so for all intents and purposes the paper is right.
- AREA directly rotates the POI's around the centre of the loactionView.

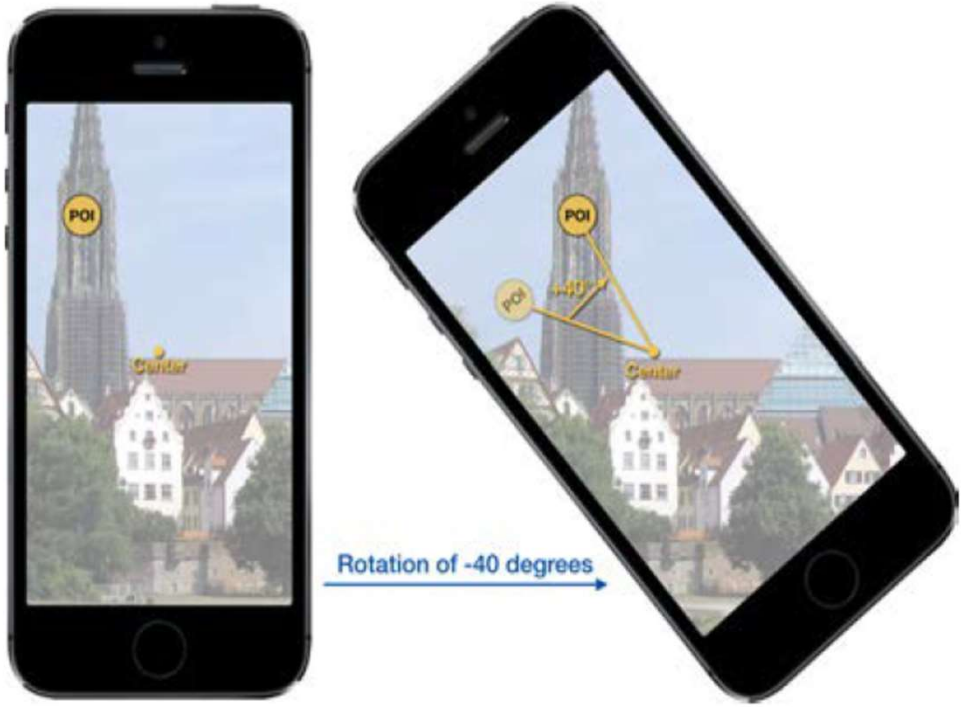


Figure 7: Android specific rotation of POI and field of view.

3.3 Comparison

- First of all, it is noteworthy that the features and functions of the two implementations are the same

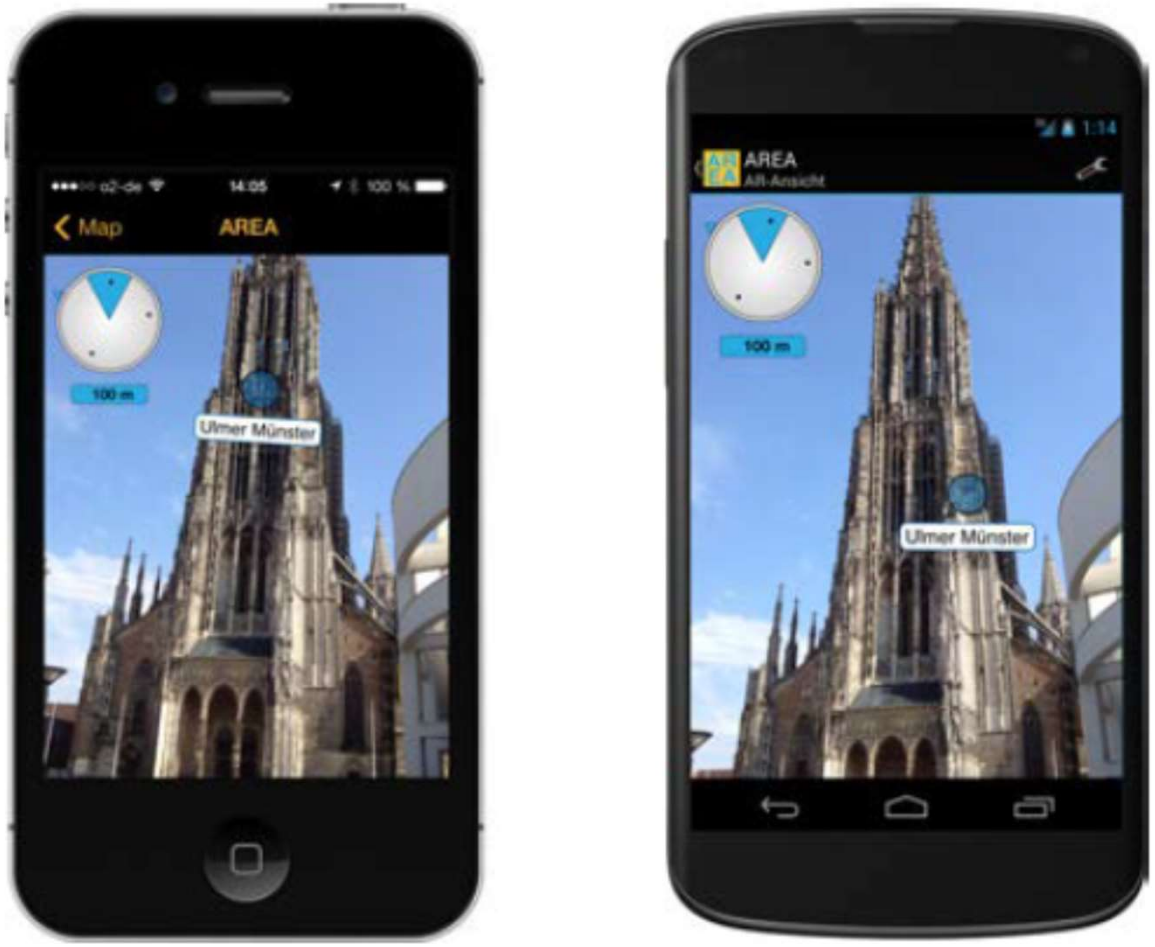


Figure 8: AREA's user interface for iOS and Android.

3.3.1 Realizing the locationView

- Difference between android and IOS
  - Android implementation did not fully match the concept but it could be adapted
  - Android version worked with NEXUS 4 but not NEXUS 5 so supporting multiple version of android became difficult.
  - On going issues with Android standardization lead to 4.4 (this project only supported 4.2)

3.3.2 Accessing Sensors

- IOS philosophy gives develops a final polished answer in terms of the sensors
- Android give more control but requires the developer to do more to combine different sensors together
- Higher version of IOS improved the application as they updated their sensors more than previous versions.
- Android pushing work to the developer leads to the developer needing to handle electrical interference from other hardware.

4 VALIDATION & 4.1 DEVELOPING BUSINESS APPLICATIONS WITH AREA

- LiveGuide used the AREA engine.
- Adaptation for POI displays
- Underlying data model was changed
- These steps where made easier by the initial model/view/ controller pattern used in the AREA development.

LiveGuide
Ditzingen

AREA
Core Engine

4.2 LESSONS LEARNED & 4.2.1 Updates of Mobile Operating Systems & 4.2.2 POI Data Format

- Android and IOS updates lead to the need to constantly update your software
- Applying their own XML scheme was need as their need customized data
- Ideally in the future they would also support ARML

5 RELATED WORK

- Mentions early work by Feiner and ,Kooper and MacIntyre
- Mentions its competitors in the for of Wikitude, Layar and Junaio.
- Mentions that most of these examples did not disclose enough details about how their designs were developed.

References

- 1 self reference but its a technical report going into more in depth detail
- Feiner, S., MacIntyre, B., H¨ollerer, T., and Webster, A. (1997). A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. Personal Technologies,1(4):208–217.
- Kooper, R. and MacIntyre, B. (2003). Browsing the real-world wide web: Maintaining awareness of virtual information in an AR information space. Int'l Journal of Human-Computer Interaction,16(3):425–446.
- Reitmayr, G. and Schmalstieg, D. (2003). Location based applications for mobile augmented reality. Proc of the Fourth Australasian user interface conference on User interfaces, pages 65–73.

Next Week papers

- Next weeks paper will be examining how cheap VR headsets suddenly appeared in 2012 , there quite short so I'll add abit of a lecture in to supplement .
- There will be 2 papers and 1 poster.
  - **A Design for a Smartphone-Based Head Mounted Display by** Olson et al
  - A poster of the paper above
  - **A Design of Low Cost Head-Mounted Display Using Android Smartphone** by Surale & Shinde