RESULTS OF AN EVALUATION OF AUGMENTED REALITY MOBILE DEVELOPMENT FRAMEWORKS FOR ADDRESSES IN AUGMENTED REALITY

Victoria Rautenbach[1], Serena Coetzee and Danie Jooste

Centre for Geoinformation Science, Department of Geography, Geoinformatics and Meteorology, University of Pretoria, Hatfield, Pretoria, South Africa
Email: victoria.rautenbach@up.ac.za, serena.coetzee@up.ac.za, dfjooste@yahoo.com

ABSTRACT

Addresses displayed on dwellings and buildings play a key role in society. Amongst others, they are used for deliveries, in household surveys, to navigate, or to find friends. Sometimes, address signs are destroyed, displaced or illegible, for example, as a result of vandalism, disasters, or poor maintenance. In augmented reality, computer-generated information is superimposed onto a live view of the real world. When address signs are not available, displaying the address in augmented reality could be immensely useful. The research presented in this article is part of a larger research endeavour to investigate augmented reality for addressing. This article presents the results of an evaluation of augmented reality mobile development frameworks for the implementation of a mobile application that displays addresses in augmented reality. Firstly, the requirements for addresses in augmented reality were identified. Three use cases informed these requirements: disaster relief, e.g. address signs are destroyed by an earthquake; household surveys, e.g. locating dwellings in informal settlements or rural areas where addresses are not assigned in any specific sequence and signs do not exist; and address data quality management, e.g. validating digital address data against addresses displayed in the physical world. Due to procurement challenges in the use cases, open source licensing and integration with open source products was identified as an important requirement. The internet was searched and a list of augmented reality mobile development frameworks was compiled. Based on the requirements, the list was shortened to seven frameworks, which were evaluated against a set of criteria informed by the requirements. The evaluation results can guide developers in choosing a framework best suitable for their specific needs and/or for integration with open source products.

KEYWORDS: addressing, augmented reality, mobile applications, development frameworks, open source

1. INTRODUCTION

Addresses play a vital role in society. They are used for deliveries, in household surveys, by utility companies, to navigate, or to find friends [1]. Sometimes, address signs, such as street names and house numbers, are destroyed or displaced as a result of vandalism, disasters or poor maintenance. Replacing the address signs takes time and is expensive. In augmented reality, computer-generated information is superimposed onto a live view of the real world. When address signs are not available, superimposing address information onto a live view of the real world could be a viable alternative.

The research presented in this article is part of a larger research endeavour on the display of geocoded address data in augmented reality. As part of this endeavour, a mobile

[1] Corresponding author: Victoria Rautenbach; email: victoria.rautenbach@up.ac.za;
Mailing address: Geography building 3-11, University of Pretoria, Hatfield, Pretoria, South Africa, 0002;
Work no: +27 12 420 3489; Cell no: +27 72 569 8916

application will be developed. This article presents the results of an evaluation of augmented reality mobile development frameworks.

In augmented reality a live view of the real world is superimposed with computer-generated information, such as text or images. Azuma [2] defines augmented reality as the real-time combination of the physical world with virtual objects. Augmented reality enhances our understanding and interaction with the physical world [3-4]. Augmented reality has proven to be useful in a variety of application fields, such as medicine [2, 5-6], education [7-8], navigation [9-10] and planning [11-12].

Allbach *et al.* [12] evaluated augmented reality applications for urban planning and design. They concluded that at the time it was not possible to recommend a single augmented reality browser, but rather commented on the shortcomings of augmented reality applications, such as limited precision of the GPS, size of the mobile device. i.e. information might be too dense to be displayed on a small screen, and the need for internet connectivity which is not always available. However, with the rapid development of augmented reality and mobile technology these shortcomings are fast disappearing [7, 9]. Leebmann [13] suggested augmented reality as a solution for disaster relief. Examples are analysing rescue routes for collapsed buildings and performing analyses from safe distance [13-14]. A drawback is that such applications need a large amount of data, e.g. 3D laser scans and site plans, in order to be useful.

Amin and Govilkar [3] compared six augmented reality software development kits (SDKs), three of them (Metaio, Wikitude and iPhone ARToolkit) are also evaluated in this paper. They compared the license type, platform support, marker generation, tracking functionality and overlaying capability. The choice of the SDKs was not justified. In this paper, mobile development frameworks for augmented reality are evaluated with the specific requirement of superimposing address information on a live view of the real world. At present, case studies of augmented reality applications where the main focus is on augmenting address data could not be found in literature. The remainder of this article is structured as follows: in section 2, three use cases are described and requirements for a mobile application based on the use cases are identified; in section 3, evaluation criteria, derived from the requirements, are described; results are presented in section 4 and discussed in section 5. We describe a proof of concept custom solution in section 6 and then conclude in section 7.

2. REQUIREMENTS FOR ADDRESSING IN AUGMENTED REALITY

Three use cases informed the requirements for the display of addresses in augmented reality: 1) disaster relief, e.g. address signs are destroyed by an earthquake; 2) household surveys, e.g. locating dwellings in informal settlements or rural areas without any address infrastructure; and 3) address data quality management, e.g. validating digital address data against addresses displayed in the physical world. In this section, the three use cases are presented and subsequently, requirements for the mobile application, based on these use cases, are described.

2.1 Use case 1: Disaster relief

In the disaster relief use case, dwellings with the house numbers and street names signs have been damaged or destroyed. A tsunami, an earthquake or fires could be the cause of such a disaster. Emergency workers are informed that there may be survivors at a specific address. Assuming that the backbone for internet (and mobile) connectivity has been destroyed, how do the emergency workers locate the site? A backup of geocoded address data was recovered from an off-site location. However, address maps are of little use as buildings, streets and signs have been destroyed. Emergency workers are equipped with smartphones connected to a satellite network, but data connectivity via satellites is expensive. Relief efforts are coordinated from a disaster management centre where a server has been set up.

a) Damage caused by the earthquake on 4 September 2010 in Christchurch, New Zealand (Photo: www.foxnews.com)

b) Damage caused by the tsunami on 11 March 2011 in Kesenuma, Japan (Photo: http://www.dailymail.co.uk)

Figure 1. Damaged or destroyed street name signs and house numbers after a disaster

2.2 Use case 2: Household surveys in rural areas

In this use case, a random sample of dwellings has been selected for a survey. Using aerial photography as a backdrop, a unique number was assigned to each dwelling without an address. Subsequently, a random number generator was used to select the sample of dwellings, based on the unique numbers assigned to each dwelling. Enumerators, i.e. people doing the interviews, have to interview the household at each of the dwellings in the sample. Some of the dwellings are in rural areas, others in an informal settlement. In both cases there is no address infrastructure: there are no street signs, no house numbers, and an intricate web of footpaths connects dwellings to each other. Paved roads connect one village or settlement to another; smaller roads beyond are typically nameless dirt roads. Dwellings in the villages or settlement are generally scattered, not necessarily arranged in a fixed pattern. In the rural areas, dwellings are sometimes interspersed with agricultural fields. See Figures 2 and 3. Figures 4 and 5 show dwellings in an informal settlement.

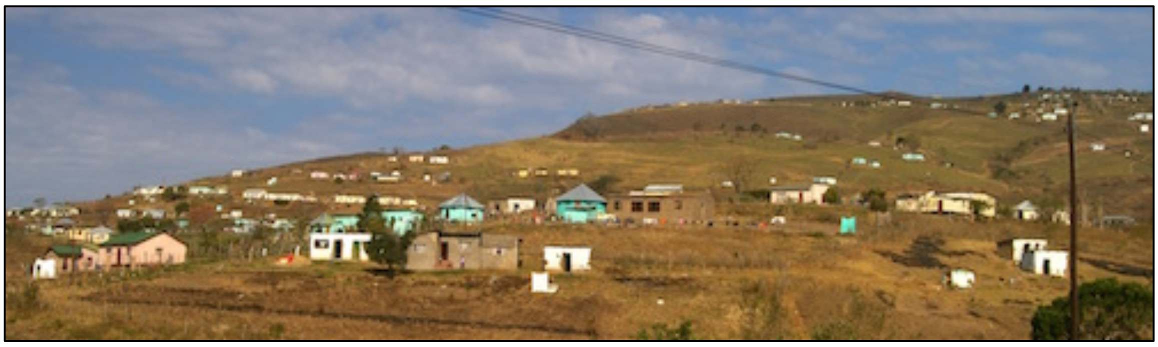


Figure 2. Dwellings in a rural village in the Eastern Cape, South Africa (Photo: Serena Coetzee)

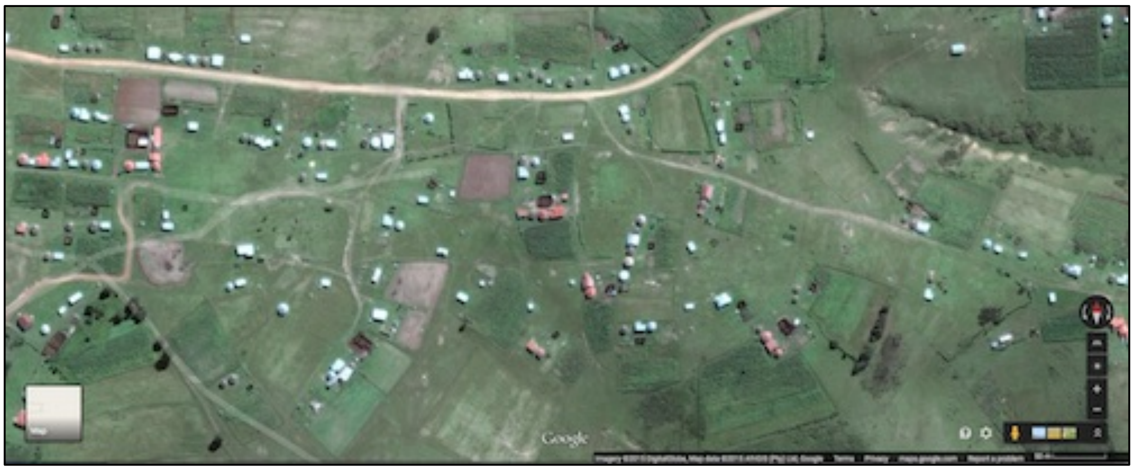


Figure 3. A rural village in the Eastern Cape, South Africa (Image from maps.google.com)

Surveys are planned and coordinated from a head office with access to ample bandwidth and internet connectivity. In the rural villages, internet connectivity is not necessarily available. Survey responses are captured on tablets and/or smartphones. Imagine an enumerator had to visit three dwellings on the hill displayed in Figure 2: How does the enumerator find the dwellings without any street signs or house numbers?

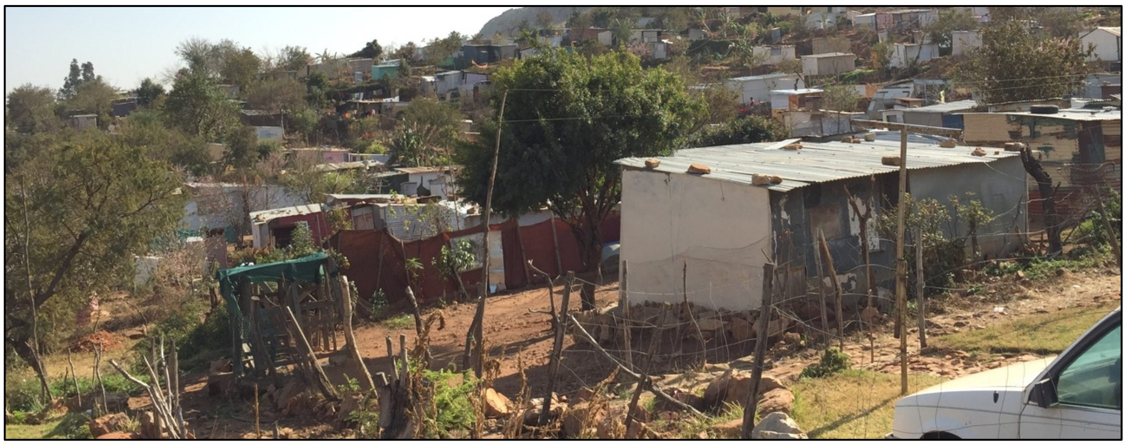


Figure 4. Dwellings in an informal settlement in the City of Tshwane, Gauteng, South Africa (Photo: Victoria Rautenbach)

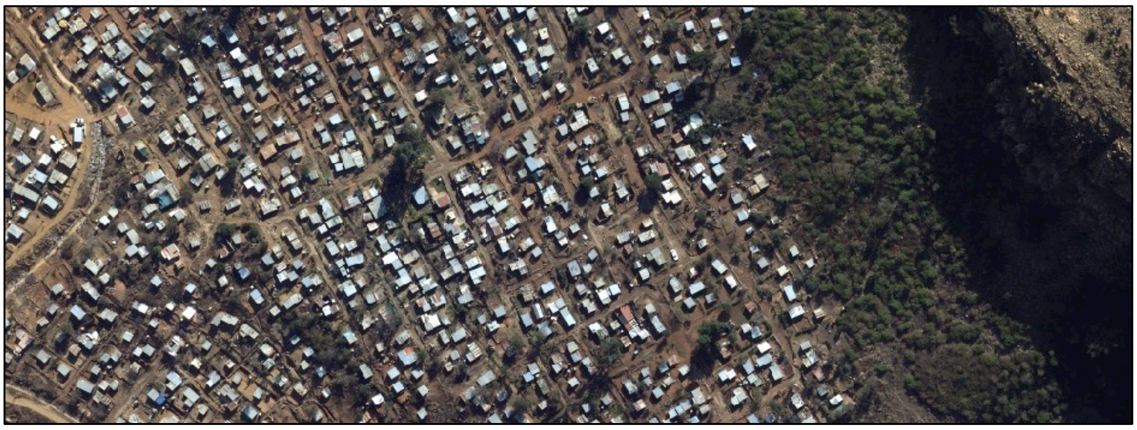


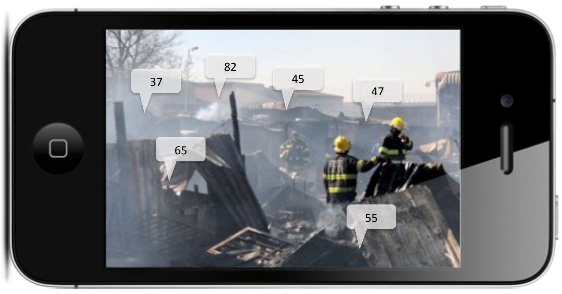
Figure 5. An informal settlement in the City of Tshwane, Gauteng, South Africa (Image from the City of Tshwane Metropolitan Municipality)

2.3 Use case 3: Address data quality management

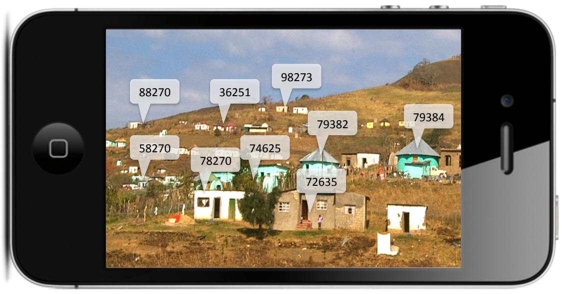
In this use case, a field worker compares geocoded address data with address signs in the real world. The local authority assigns house numbers to dwellings and buildings when building plans are approved, i.e. before the buildings are constructed. Geocoded house numbers are stored in a geospatial database. House numbers are not verified after the buildings have been erected and when buildings are altered or extended, one does not have to apply for a house number again. As a result, owners and occupants may put up house numbers, which are not reflected in the geospatial database at the local authority. Ultimately, this may lead to returned mail and service delivery interruptions, e.g. when bills by the local authority do not reach the owner. Therefore, from time to time, the local authority needs to compare its digital address database against address signs in the real world in order to harmonize the digital representation with the real world.

2.4 Requirements for addresses in augmented reality

In all three use cases, superimposing digital address data onto a live view of the real world could solve the problem at hand: to locate the address where survivors need assistance; to visit specific dwellings in a rural village or informal settlement without an address infrastructure; and to compare digital address data with house numbers in the real world. See Figure 6.



a) Use case 1: Disaster relief (Photo: www.citizen.co.za)



b) Use case 2: Household surveys in rural areas

Figure 6. An example of addresses displayed in augmented reality in a rural village setting

In two of the three use cases, internet connectivity is available at a coordinating centre, but not in the field. Therefore, the display of address information in the augmented reality application must be available, even if the device is offline.

In all three cases, geocoded address data is available. In the augmented reality view, the address should be superimposed as close as possible to the actual location of the address and one has to be able to distinguish an address from its neighbouring address. Appropriate precision is therefore important; 'appropriate' because in rural areas dwellings are spaced further apart, requiring less precision; whereas in densely populated informal settlements, better precision is required. It would also be useful to know the distance between the smartphone and the address. For example, in the household survey use case, this would allow the enumerators to plan their route of interviews around the village.

In the disaster use case, any delay for procurement processes is not an option: as many licenses as may be needed for the relief exercise have to be available immediately in order to save lives. The use case of rural villages and informal settlements without an address infrastructure is often found in developing countries with financial constraints and plagued with corrupt and/or lengthy procurement processes. Therefore, ideally, the mobile application should be available free of charge.

Nice-to-have requirements include navigation and/or wayfinding; a map view in addition to the augmented reality view; and the capability to edit or update address data, or any other information linked to the address, e.g. survey responses or notes about the dwelling. Table 1 lists the functional and non-functional requirements for addresses in augmented reality.

Table 1. Functional and non-functional requirements

Description	Type of requirement
Download and install the app (immediate availability)	Non-functional
No procurement process (to avoid corruption)	Non-functional
Offline availability of address data	Non-functional
Appropriate precision, depending on the density of addresses	Non-functional
Display distance between the smartphone and address	Functional
Digital address data superimposed onto a live view of the real world	Functional
Navigation and/or wayfinding functionality	Functional (nice-to-have)
Map view	Functional (nice-to-have)
Functionality to edit/update the address data	Functional (nice-to-have)
Functionality to edit/update information linked to an address	Functional (nice-to-have)

3. METHODOLOGY

In this section, we describe the two-phase approach followed to evaluate the augmented reality mobile development frameworks. They were evaluated in the context of an augmented reality solution for addresses that meets the requirements identified in Section 2. Refer to Figure 7 for an overview of the two-phase evaluation.

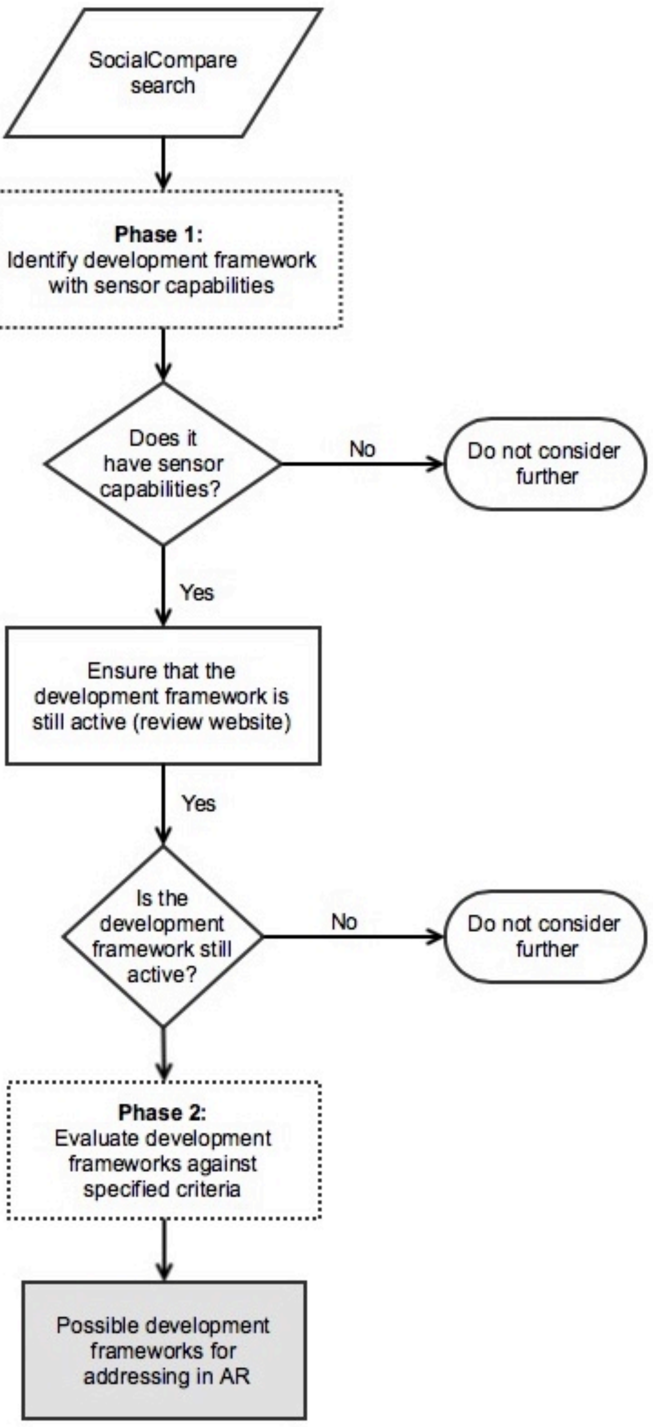

SocialCompare search
Phase 1: Identify development framework with sensor capabilities
Does it have sensor capabilities?
No
Do not consider further
Yes
Ensure that the development framework is still active (review website)
Yes
Is the development framework still active?
No
Do not consider further
Phase 2: Evaluate development frameworks against specified criteria
Possible development frameworks for addressing in AR

Figure 7. Overview of the two-phase evaluation

We consulted SocialCompare (http://socialcompare.com/en), a collaborative online comparison tool, where a comparative list of augmented reality SDKs and frameworks is actively maintained[2]. Guided by the comments posted on the SocialCompare list, two additional development frameworks were included, namely iPhone ARToolkit and Layar iPhone ARToolkit was added to the SocialCompare list between the time of the evaluation and the writing of this paper. This resulted in a list of 68 SDKs and frameworks.

In the first phase of the evaluation, development frameworks were disqualified if they did not support GPS and inertial measurement unit IMU) sensors, which are required to locate and correctly display geocoded address data in augmented reality. Thereafter, we reviewed the websites of each of the remaining 12 frameworks. Inactive and irrelevant frameworks, e.g. frameworks mainly aimed towards gaming, were disqualified. This left seven frameworks for evaluation in the second phase.

During phase two, each development framework was evaluated in more detail against a set of evaluation criteria and also tested to ensure that it complied with the requirements for the display of addresses in augmented reality. Table 2 describes the three categories of evaluation criteria: general, functional and non-functional. The criteria are based on requirements identified in Section 2.

Table 2. Evaluation criteria for the display of addresses in augmented reality

	Criteria	**Criteria meaning**
1. General criteria	1.1. Platform	Platforms and/or operating systems supported by the development framework, such as Windows Mobile, Android or iOS. Development frameworks that support the implementation of cross-platform mobile applications are desirable.
	1.2. Programming language	Which programming languages does the development framework supports? The programming language and the supported platform (1.1) are closely related. For example, Swift is only available for iOS but Java can be used for both Android and Windows Mobile. Widely used programming languages are desirable.
	1.3. License	Refers to whether the license under which the development framework is available, e.g. open source, freeware or proprietary. The licensing affects the manner in which the development framework may be used and also how derived implementations may be distributed. As few constraints as possible are desirable.
	1.4. Implemented standards	The implementation of standards contributes to the interoperability and modularity of an implementation. Various standards may contribute to interoperability, for example, data encoded in eXtensible Markup Language (XML) facilitates data exchange from different sources. Recently, the Open Geospatial Consortium (OGC) published the Augmented Reality Markup Language (ARML) that uses XML to describe the location and appearance of augmented visual objects.

[2] http://socialcompare.com/en/comparison/augmented-reality-sdks, last updated 31 July 2015

	Criteria	Criteria meaning
	1.5. Offline availability	The connectivity required by the augmented reality application in the field. It is desirable to have all functionality available in the field without any connectivity.
2. Functional criteria	2.1. Data source	The flexibility of the development framework to allow access to various data sources. Least desirable is a dedicated source, e.g. via a quick response (QR) code, hosted by a provider or organization.
	2.2. Data display	The capability to adjust the visual representation of the data being superimposed on the camera feed. For example, can the text size be altered? Flexibility in configuring the visual representation is desirable.
	2.3. Object behaviour	The ability to add behaviour to the object, e.g. by adding triggers or events to the object. For example, if the user clicks on the object, additional information is displayed.
	2.4. Display radius	The flexibility to adjust the amount of content that is displayed in the camera feed based on the distance between the user and objects (e.g. addresses or dwellings). For example, specify that only addresses that are within a 50m radius from the user are displayed.
	2.5. Visual search	The ability to recognize a specific object based on additional information, such as a photo or light detection and ranging (LiDAR) data.
3. Non-functional criteria	3.1. Ease of integration with other applications	Built-in capabilities to facilitate integration with other applications, such as PostGIS, QuantumGIS or ArcGIS. For example, integration with PostGIS will allow seamless access to a database without the need for additional code or third-party products.
	3.2. Ease of extending the framework	Refers to the effort required to add additional functionality to the development framework and to port the solution to different platforms.
	3.3. Usability	Refers to the user friendliness framework and ease of installation of the development framework.
	3.4. Documentation and support available	The documentation and additional avenues available to developers for support, such as forums and mailing lists.

4. EVALUATION RESULTS

4.1. Phase 1

The list of 68 development frameworks compiled from the SocialCompare source was reduced to 12 by disqualifying development frameworks that do not support GPS or IMU sensors. After reviewing the websites of these 12 frameworks, the following were eliminated as they did not meet the requirements for the use cases: ARmedia[3], BeyondAR[4], LibreGeoSocial[5], Total Immersion[6] (previously known as D'Fusion), and Xloudia[7].

LibreGeoSocial is an open source project that looked promising initially. However on closer inspection, it was eliminated, as it has not been updated for quite a number of years. The LibreGeoSocial website was also recently removed after being hacked. Total Immersion and Xloudia are more geared towards proprietary solutions and provide little flexibility. For example, these frameworks cannot easily be customized or extended to serve a different purpose than originally intended. Lastly, ARmedia and BeyondAR were eliminated as they focus on superimposing 3D objects. This is useful for gaming or advertising purposes, but not for superimposing textual information as required in the use cases. The remaining seven development frameworks were evaluated further in phase 2.

4.2. Phase 2

The following development frameworks were included in the phase 2 evaluation: ARLab, iPhone ARToolkit, DroidAR, Layar, Metaio, PanicAR and Wikitude. A brief overview of each of the evaluated frameworks follows.

1. **ARLab**[8]
   ARLab is developed by a commercial company that focuses on augmented reality solutions. They have divided their products into the following modular units: the AR Browser, image matching, 3D engine, image tracking, object tracking and virtual buttons. The AR Browser is a SDK that allows the development of a location-based augmented reality application for points of interests (POIs) within minutes.

2. **iPhone ARToolkit**[9]
   iPhone ARToolkit is a GNU GPL v3 open source library developed for Apple iOS. iPhone ARToolkit was not originally included, but was added to the list after being mentioned in comments on the SocialCompare page. iPhone ARToolkit has not been updated since 2013, but there are various blog posts on how to fix errors due to deprecated functions in newer versions of iOS. This shows that iPhone ARToolkit still has strong community support. The framework makes use of MapKit and UIKit, two standard Apple iOS libraries.

3. **DroidAR**[10]
   DroidAR is an open source framework available under the GNU GPL v3 license. It focuses on location and marker based augmented reality. Even though the code has not been updated in the GitHub repository since 2013, there seems to still be an active user community. With small adjustments in the code, DroidAR runs on the latest Android operating systems. It is not clear why the project stagnated, as it averaged about 1500

[3] http://dev.inglobetechnologies.com
[4] http://beyondar.com/platform
[5] http://www.libregeosocial.org
[6] http://www.t-immersion.com/products/dfusion-suite
[7] http://www.xloudia.com/xloudia-imerico/
[8] http://www.arlab.com/arbrowser
[9] https://github.com/nielswh/iPhone-AR-Toolkit
[10] http://droidar.blogspot.com

website hits per month in 2013. The DroidAR website is currently out-dated and the development status of the framework is unclear.

4. **Layar**[11]
   Layar was not included in the SocialCompare list. Based on comments posted on this list, it was included in the evaluation because it is one of the leading augmented reality frameworks available today. Layar provides developers with an open platform to publish and discover augmented reality layers. 'Reality layer' is the Layar term for data. Layar employs representational state transfer (REST) services to serve POIs facilitating, the integration of Layar with other Layar applications.

5. **Metaio**[12]
   Metaio is a commercial company providing augmented reality solutions for a variety of application fields. Metaio offers six products that cover all the requirements of the augmented reality value chain, ranging from development tools to out of the box solutions. Metaio was included in the evaluation, but was acquired by Apple Inc. in May 2015, after this evaluation had been completed. Metaio does no longer offer any subscriptions for purchase.

6. **PanicAR**[13]
   PanicAR is a SDK that is fully customizable and focused on location-based AR. PanicAR is fully integrated with the Apple MapKit, allowing the developer to visualize POIs on a 2D map, in addition to the augmented reality view. PanicAR states that it is fully white label, i.e. it can be completely re-branded.

7. **Wikitude**[14]
   Wikitude was the first openly available location-based augmented reality application, and it has won numerous awards, such as the Android Developers Challenge in 2008. Wikitude provides a multifunctional framework that includes numerous features, for example, location-based augmented reality, 3D model rendering, and image recognition and tracking.

The Apple iOS and Android **platforms** are supported by most of the development frameworks (refer to Table 3). DroidAR is the only framework that does not support Apple iOS and iPhone ARToolkit does not support Android. Apple iOS requires high subscription fees for iOS development and implementation of the framework on iOS requires conversion of the code to Objective-C. Wikitude has the widest platform support: apart from Android and iOS, Google Glasses, Blackberry and Windows Mobile are supported. As mentioned in Table 1, the supported platform and the **programming language** are closely related. The primary supported programming languages are Java (used on Android) and Objective-C (used on iOS). The majority of frameworks are available under a proprietary **license**, but offer a free development framework option (freeware) with limited functionalities and a watermark. Among the evaluated frameworks, iPhone ARToolkit and DroidAR are the only open source frameworks.

[11] https://www.layar.com
[12] http://www.metaio.com
[13] http://panicar.dopanic.com
[14] http://www.wikitude.com

The OGC ARML 2.0 standard [15] is currently the only augmented reality **standard** published in the geospatial industry. ARML is composed of XML for describing locations and appearance of virtual objects, and of ECMAScript for dynamic access to the properties of the objects. Layar, Metaio and Wikitude implement ARML. Other standards, such as OGC web services, e.g. the web feature service or web processing service, and encoding standards, such as XML and JSON, contribute to the modularity and integration of the framework with other applications. At present, no framework implements OGC services. However, all the frameworks rely on standard encodings.

Table 3. Overview of the results of the general criteria

			ARLab	**iPhone ARToolkit**	**DroidAR**	**Layar**	**Metaio**	**PanicAR**	**Wikitude**
1. General criteria	**1.1. Platform**	*Android*	X		X	X	X	X^+	X
		Blackberry				X			
		iOS	X	X		X	X	X	X
		Other					X		X
	1.2. Programming languages	*Java*	X		X	X	X	X	X
		Objective-C	X	X		X	X	X	X
		Other				X	X		X
	1.3. License	*Open Source*		X	X				
		Freeware				X	X		X
		Proprietary	X			X	X	X	X
	1.4. Implemented standards	*OGC ARML*				X	X		X
		OGC web services							
		Encoding standards	X	X	X	X	X	X	X
	1.5. Offline availability	*Yes*		X	X				
		No	X*			X*	X*	X*	X*

Offline availability is crucial when working in rural areas or in a disaster relief situation where connectivity is limited. Since iPhone ARToolkit and DroidAR are open source, users can implement the application in such a way that it reads the data from a local source, for example from a file in comma-separated values (CSV) format or from a JavaScript object notation (JSON) message. With the proprietary frameworks, the default method of accessing information is via a service that requires connectivity. However, if the format of the response is known, the developer can create a file in the same format and the application can then read the data from this file.

In all frameworks, the primary **data source** is a web service. Other options are directly from a database or through native code (refer to Table 4). Layar and Metaio access the data through a proprietary web service that is available as a ‘black box’ to the user. With Metaio, the user accesses the web service through a channel identifier. The channel is the entry point to the Metaio Cloud backend from where the information is requested. The channel identifier can be distributed in two ways: by QR code or by publishing the channel identifier. When using Layar, the user publishes the data on the Layar service and the data is then seen as a layer. The format of the layer is not known. The user can then only access the layers via this Layar service. All frameworks, except Layar, allow the user to make use of native code to acquire data from custom sources. However, with the proprietary frameworks, this is a challenge as the data request has to be replaced with custom code. For example, with Metaio the data can only be accessed via a channel identifier. The user has to reproduce the Metaio service data format and inject it into the implementation.

Table 4. Overview of the results of the functional criteria

			ARLab	**iPhone ARToolkit**	**DroidAR**	**Layar**	**Metaio**	**PanicAR**	**Wikitude**
2. Functional criteria	**2.1. Data source**	*Web service*	X	X	X	X*	X*	X	X*
		QR code					X		
		Database	X	X	X			X	X
		Native code	X	X	X		X	X	X
		Other					X		
	2.2. Data display	*The visual representation can be altered?*	X	X	X	X		X	X
		User can swap between which information is displayed?	X	X	X	X	X	X	X
	2.3. Object events	*Does the framework implement event triggers?*	X	X	X	X	X	X	X
	2.4. Display radius	*The display radius can be altered?*	X	X	X	X	X	X	X
	2.5. Visual search	*Photo*	X^{+}				X		X
		LiDAR	X^{+}				X		X
		Other, e.g. 3D objects	X^{+}		X		X		X

Augmented reality superimposes information (**data display**) on a live feed, for example a camera feed. Various types of data or information can be superimposed, such as text, images or videos. For addressing, the display of text is important. In all frameworks, the fields to be displayed from a table or database, can be configured. However, adjusting the visual representation can be tricky, especially with proprietary frameworks that do not allow rebranding. iPhone ARToolkit relies on the UIKit to adjust the visual representation of the objects. The Metaio API specifies the access of POIs and also events (**object events**), but does not document how to change the visual representation of the POIs on the live feed. This code is hidden from the user. In contrast, PanicAR promotes itself as being a white label software, i.e. it can be completely be rebranded by the user. All frameworks implemented object behaviour in the form of events and/or triggers.

The **display radius** is a filtering mechanism based on distance from the user, for example, one can specify that only objects within a 100m radius are displayed. All evaluated frameworks provided the users with a method of adjusting or specifying this radius. However, the display can still get crowded with information if there are many points within close proximity of the user. Due to limited options for setting the visual presentation of information in augmented reality (see previous paragraph), the way in which label overlap is avoided (or not) cannot always be set.

Visual search provides added intelligence to the application by not only relying on the location of the object only, but also on additional information, such as a photo, LiDAR data or 3D model. ARLab provides modular solutions, therefore the AR Browser does not implement these functionalities, but they are available in other packages developed by ARLab, such as the AR image matching or 3D engine. Wikitude provides the widest range of functionalities, including support for Google Glasses to recognize objects in the wearer's view based on image recognition. Metaio follows closely behind Wikitude. DroidAR does not support image or photo recognition or LiDAR, but does support gesture recognition that can be used to develop virtual reality applications to complement the augmented reality applications. Visual search functionality is less important for the use cases described in this paper.

At present, none of the frameworks have built-in **integration with other products**, such as ArcGIS, QuantumGIS or PostGIS (refer to Table 5). Although this is not essential, integration with other products would make it easier to access or exchange information. For example, the application could directly access information from a PostGIS database and display it in the augmented reality application.

It is generally not **easy to extend** proprietary frameworks, as the code is not available and licensing constraints prohibit the user from extending the framework. iPhone ARToolkit and DroidAR are the only applications that can easily be extended, as they are open source frameworks. Open source frameworks encourage the modification and extension of the code to produce higher quality frameworks, and also to add new functionalities.

All the frameworks were found to be very **usable**. Frameworks either used the Android Studio with additional libraries or a stand-alone SDK that could be installed using a one-click installer. Tools typically provided auto completion and error checking that assisted with fast and effective coding.

All frameworks, except DroidAR, provide a variety of avenues of **support**. They provide extensive, well-structured and up-to-date documentation with numerous examples and code snippets. Additionally, instructional videos, issue trackers, forums and mailing lists are provided. Official documentation for DroidAR is fairly limited: the majority of the documentation is provided by the user community in the form of non-official documentation, instructional videos, and basic examples on GitHub.

Table 5. Overview of the results of the non-functional criteria

			ARLab	**iPhone ARToolkit**	**DroidAR**	**Layar**	**Metaio**	**PanicAR**	**Wikitude**
3. Non-functional criteria	**3.1. Ease of integration with other GIS applications**	*Built-in integration with any GIS software product?*							
	3.2. Ease of extending the framework	*Can the framework easily be extended?*		X	X				
	3.3. Usability	*The framework is easy to install?*	X	X	X	X	X	X	X
		The framework is generally easy to use?	X	X	X	X	X	X	X
	3.4. Documentation and support available	*Documentation is clear and up to date?*	X			X	X	X	X
		Forums and mailing lists are available to interact with developers and user community?	X	X	X	X	X	X	X
		Support desk	X			X	X	X	X

5. DISCUSSION OF RESULTS

Our aim was to identify and evaluate existing development frameworks that could be used for the development of a mobile application that displays addresses in augmented reality. The requirements are based on three use cases in disaster relief, household surveys and address data quality management respectively. All the frameworks that were evaluated in phase two are available on the Android platform. Android has various advantages over other operating systems, such as iOS and Windows Mobile, primarily, because Android is an open source Linux-based mobile operating system. Other advantages of developing Android applications include integration with other Java based software, and also wider adoption. At the time of writing, Android was the operating system used by the majority, more than 80%, of mobile phones on the market[15].

At the moment, only two open source frameworks (iPhone ARToolkit and DroidAR) satisfy the requirements for the use cases (see Section 2). Other open source frameworks, such as BeyondAR, LibreGeoSocial and Mixare[16], do not meet the requirements. We also found that the open source frameworks (including iPhone ARToolkit and DroidAR) were generally out-dated and the developers had moved on to other things. The need for an open source framework is apparent as proprietary frameworks do not allow extensions and could result in vendor lock-in, forcing users to make use of their web services at additional cost. An open source framework will also allow users to more easily integrate the application with other products. Such integration is important as it facilitates accessing information from a database in PostGIS or the serving of data as a feature service from GeoServer, for example.

ARML 2.0 was published in 2015 by the OGC [15]. It allows users to develop a XML style sheet that specifies the appearance of objects and their anchors. Anchors refer to the location or coordinates of the display item. Additionally, ARML defines ECMAScript bindings that allow the dynamic modification of the augmented reality scene subject to user input and behaviour. Wikitude originally developed ARML 1.0, and Metaio and Layar also implement ARML 2.0. Currently, none of these implementations conform completely to ARML 2.0. A likely reason is that the standard was only recently published and conformance testing is probably still in progress.

Offline availability of the data and application is critical when working in a rural area or a disaster relief situation where connectivity is limited. A current limitation of all the proprietary frameworks is extensive additional programming required to bypass the data acquisition method, for example, to bypass the data from a commercial web service in order to read from a local file. The limitation implies that the user has to know and understand the data format that is consumed by the application, so that native code can be written to request data in this format from a different source, either locally or online. Publishing data on a commercial server might also raise security or privacy concerns that might discourage the use of the augmented reality application.

A prototype of the augmented reality solution for addresses was implemented in iPhone ARToolkit (iOS) and Metaio (Android). iPhone ARToolkit was selected as it is an open source application, available on Apple iOS. Metaio provided a customizable solution, namely Junaio that could be used to test the framework. The prototypes were tested on the campus of the University of Pretoria, and successfully displayed addresses in augmented reality, i.e. an address could be correctly associated with a building. From a developer point of view, both frameworks were user friendly and the prototype was easy to implement. However, some functions in iPhone ARToolkit were deprecated and had to be fixed in the code. The iPhone ARToolkit community is very active and users will implement fixes if they are reported. However, such small issues might put-off beginners and cause them to rather look at proprietary options. With Metaio, customizing the data source to something other than a dataset or layer identified by a channel identifier is not possible. Configuring the visual representation of addresses in Metaio was not possible, as the Metaio branding cannot be changed through its library.

---

[15] http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[16] http://www.mixare.org

Even though the frameworks provided all the tools to develop an augmented reality application for addressing, the precision of a phone GPS might still cause challenges when the application is used in a densely populated environment. A phone GPS can have precision of approximately 5m horizontally. This level of precision may take about 2-5 minutes to achieve and is sufficient for sparsely populated (rural) areas.

The results of our evaluation highlight the respective strengths and weaknesses of the frameworks, which can guide developers to choose the framework best suited for their specific needs and requirement. The evaluation shows that all the evaluated frameworks meet the technical requirements for an augmented reality application for addresses identified for use cases in disaster relief, household surveys and address data quality management. However, restrictions on content due to the business model in the proprietary frameworks and the lack of maintenance and support of open source frameworks prompted us to look at alternatives.

Virtual globes or visualization frameworks, such as glob3mobile[17], have been suggested as a possible augmented reality application. Even though these applications are inherently spatial, they do not provide the desired functionality for an augmented reality solution for the address use cases described in this article. For example, these globes or frameworks are built for data sets that cover a large area. In contrast, displaying addresses in augmented reality requires pin-point precision spanning a much smaller area.

Each framework had its limitations and no framework satisfied all our requirements. A proof of concept application was therefore developed in Java making use of the Android SDK[18] and standard libraries. See Figure 8. With this we demonstrated that the framework limitations can be overcome. The proof of concept application reads addresses from a comma-separated values (CSV) file stored locally on the mobile phone. This allows a fieldworker to download the CSV file to the mobile device while connected to the internet. The address data is then available in the field without the need for connectivity. Non-standard functionality, such as coordinate system conversions, were performed externally for the proof of concept. In future versions, integration of the proj.4[19] library will be explored. JavaScript frameworks, such as AngularJS[20], enable the development of cross-platform mobile applications, e.g. for Apple iOS, Android and Windows Mobile. In future work, we plan to explore the use of such JavaScript frameworks for the address use cases.

[17] http://www.glob3mobile.com
[18] http://developer.android.com/sdk/installing/index.html
[19] https://github.com/OSGeo/proj.4/wiki
[20] https://angularjs.org

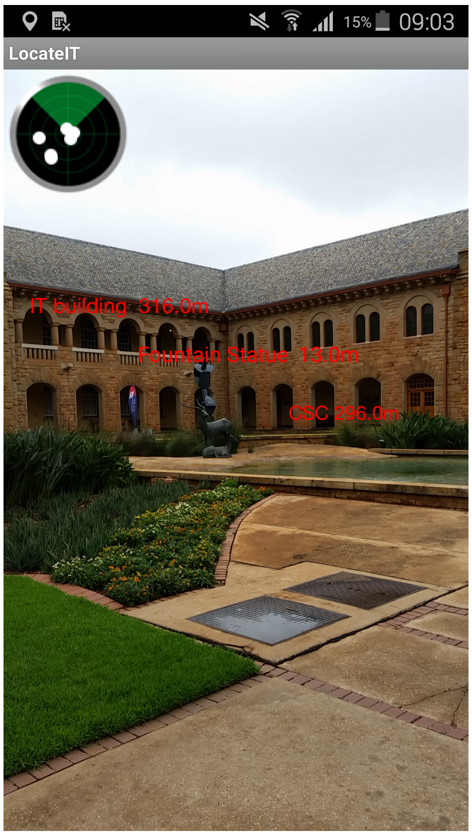


Figure 8. Screenshot of the proof-of-concept application making use of the Android SDK and standard Android libraries

6. CONCLUSION

In this paper, seven mobile development frameworks for augmented reality were evaluated. The evaluation was based on requirements for three use cases: 1) disaster relief, e.g. address signs are destroyed by an earthquake; 2) household surveys, e.g. locating dwellings in informal settlements or rural areas without any address infrastructure; and 3) address data quality management, e.g. validating digital address data against addresses displayed in the physical world.

A two-phased evaluation was followed. In the first phase, development frameworks were disqualified if they did not support GPS and IMU sensors. These are required to locate and correctly display geocoded address data in augmented reality. During phase two, seven development frameworks, namely ARLab, iPhone ARToolkit, DroidAR, Metaio, PanicAR and Wikitude, were evaluated against three categories of evaluation criteria: general, functional and non-functional.

Results show that most of the evaluated frameworks are available on Android and iOS, with Java and Objective-C the most widely supported programming languages on the respective platforms. Only two of the frameworks were distributed with open source licenses. Standards, such as XML and JSON, are used for encoding; the ARML standard is not widely implemented; and OGC web services are not implemented. Generally, the frameworks are designed to access data sources via the internet through a web service. Two features were available in all frameworks: specifying a radius for objects to be displayed in the augmented reality view, and specifying the textual information to be superimposed on the live view of the world. Manipulating the visual presentation, e.g. changing the text size or colour, is not widely available. The evaluated frameworks were found to be very usable and most of them provide a variety of avenues of support.

Two prototypes were implemented in iPhone ARToolkit on iOS and in Metaio on Android. The prototypes were tested on the university campus and successfully displayed addresses in augmented reality, i.e. an address could be correctly associated with a building. From a developer point of view, both frameworks were user friendly and the prototype was easy to implement after some tweaking of the code in one or two of the frameworks.

Based on the evaluation results, iPhone ARToolkit and DroidAR are most suitable for the three addressing use cases. Both frameworks seem to have active user communities who implement bug fixes in the case of iPhone ARToolkit and develop documentation in the case of DroidAR. However, infrequent updates to the code base are a concern. An alternative to using an existing development framework would be the implementation of augmented reality functionality from scratch. This is possible, for example, with the Android SDK and libraries. However, programming is required and this option does not follow the software engineering good practice of software re-use. Amongst others, a developer will have to (re-)implement functionality for the conversion between coordinate systems, for the calculation of distances between the user and anchor and for optimization the GPS precision.

Restrictions on content due to the business model in the proprietary frameworks and the lack of maintenance and support of open source frameworks prompted us to look at alternatives. Instead of using a framework, a proof of concept application was developed making use of the Android SDK and standard Android libraries. So far, results are promising and this avenue will be further explored in future work.

At present, the Open Source Geospatial Foundation (OSGeo) does not support any augmented reality development framework. The fact that open source frameworks exist but are not updated frequently, could suggest that they are in need of a structured support system, such as OSGeo, that would provide financial, organizational and legal support. An augmented reality framework that displays objects, sourced from spatial data layers in shapefiles or through web feature services on a server, could provide significant benefits to the free and open source for geospatial community in a variety of use cases.

The research presented in this article is part of a larger research endeavour on the display of geocoded address data in augmented reality. In future work, we plan to do empirical research to evaluate the use of augmented reality for addresses in each of the three use cases.

ACKNOWLEDGEMENTS

We would like to thank the Department of Science and Technology/National Research Foundation (DST/NRF) intern, Linda Khoza, who worked on the development of the augmented reality applications.

REFERENCES

1. Coetzee, S., and Cooper, A.K. (2007). What is an address in South Africa?. *South African Journal of Science*, 103, 449–458.
2. Azuma, R. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4), 355–385.
3. Amin, D., Govilkar, S. (2015). Comparative Study of Augmented Reality Sdk's. *International Journal on Computational Science & Applications*, 5(1), 11–26.
4. Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., Ivkovic, M. (2011). Augmented reality technologies, systems and applications. *Multimedia Tools and Applications*, 51, 341–377.
5. Kounavis, C.D., Kasimati, A.E., Zamani, E.D. (2012). Enhancing the tourism experience through mobile augmented reality: Challenges and prospects. *International Journal of Engineering Business Management*, 4, 1–6.
6. Dünser, A., Grasset, R., Billinghurst, M. (2008). A survey of evaluation techniques used in augmented reality studies. *ACM SIGGRAPH ASIA 2008 courses on - SIGGRAPH Asia '08*. New York, USA: ACM Press, 1–27.
7. Wu, H., Lee, S.W., Chang, H., Liang, J. (2013). Current status, opportunities and challenges of augmented reality in education. *Computers & Education*, 62, 41–49.
8. van Krevelen, D., Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *The International Journal of Virtual Reality*, 9(2), 1–20.
9. Wen, J., Deneka, A., Helton, W.S., Billinghurst, M. (2014). Really, it's for your own good...making augmented reality navigation tools harder to use. *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA '14*. New York, USA: ACM Press, 1297–1302.
10. Mulloni, A., Seichter, H., Schmalstieg, D. (2011). Handheld augmented reality indoor navigation with activity-based instructions. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*, 211–220.
11. Anagnostou, K., Vlamos, P. (2011). Square AR: Using augmented reality for urban planning. *Proceedings - 2011 3rd International Conferenceon Games and Virtual Worlds for Serious Applications, VS-Games 2011*. 128–131.
12. Allbach, B., Memmel, M., Zeile, P., Streich, B. (2011). Mobile Augmented City – New Methods for Urban Analysis and Urban Design Processes by using Mobile Augmented Reality Services. *Proceedings of RealCORP*, 6, 633–641.
13. Leebmann, J. (2006). Application of an augmented reality system for disaster relief. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-5/W10.
14. Leebmann, J. (2004). an Augmented Reality System for Earthquake Disaster Response. *Virtual Reality*, 35, 909–914.
15. Open Geospatial Consortium, (2015). *OGC Augmented Reality Markup Language 2.0 (ARML 2.0)*. http://docs.opengeospatial.org/is/12-132r4/12-132r4.html. Accessed 27 April 2016.

Option 1:

Figure	**Description**
Figure 1. Damaged or destroyed street name signs and house numbers after a disaster	This figure consists of two images that show examples of street name signs and house numbers that were destroyed during a disaster.

Figure 2. Dwellings in a rural village in the Eastern Cape, South Africa (Photo: Serena Coetzee)	Photo of a rural village in the Eastern Cape, South Africa
Figure 3. A rural village in the Eastern Cape, South Africa (Image from maps.google.com)	An aerial photograph of a rural village in the Eastern Cape, South Africa.
Figure 4. Dwellings in an informal settlement in the City of Tshwane, Gauteng, South Africa (Photo: Victoria Rautenbach)	Photo of the Alaska, informal settlement in the City of Tshwane, Gauteng, South Africa
Figure 5. An informal settlement in the City of Tshwane, Gauteng, South Africa (Image from the City of Tshwane Metropolitan Municipality)	An aerial photograph of the informal settlement of Alaska in the City of Tshwane, Gauteng, South Africa. The aerial photograph shows how densely populated the informal settlement is.
Figure 6. An example of addresses displayed in augmented reality in a rural village setting	Example of how address can be displayed in augmented reality applications.
Figure 7. Overview of the two-phase evaluation	Flow diagram depicting the two-phased process that was used during the evaluation.
Figure 8. Screenshot of the proof-of-concept application making use of the Android SDK and standard Android libraries	A proof of concept was developed using the Android SDK. This figure is a screen shot of when the application was tested on the University campus.

Option 2:

Figure 1. Damaged or destroyed street name signs and house numbers after a disaster

Figure 2. Dwellings in a rural village in the Eastern Cape, South Africa (Photo: Serena Coetzee)

Figure 3. A rural village in the Eastern Cape, South Africa (Image from maps.google.com)

Figure 4. Dwellings in an informal settlement in the City of Tshwane, Gauteng, South Africa (Photo: Victoria Rautenbach)

Figure 5. An informal settlement in the City of Tshwane, Gauteng, South Africa (Image from the City of Tshwane Metropolitan Municipality)

Figure 6. An example of addresses displayed in augmented reality in a rural village setting

Figure 7. Overview of the two-phase evaluation

Figure 8. Screenshot of the proof-of-concept application making use of the Android SDK and standard Android libraries