Idroid - interest aware augmented reality

Ricardo Brilhante
INESC-ID / Technical University of Lisbon
ricardo.brilhante@ist.utl.pt

ABSTRACT

Mobile augmented reality applications are increasingly more popular. The ability to merge a virtual world with the real one is a fascinating idea, and the possibility of having such a system available in a mobile device makes it even more interesting given its inherent ubiquity.

Ensuring the freshness of (augmented reality) information provided to the user on the mobile device is of utmost importance. At the same time, the user must not be overwhelmed by useless information, and the network usage should be kept to a minimum, so that scalability of the overall system can be ensured.

We present Idroid, a Location Based Augmented Reality system that is scalable and provides the information the user needs, according to his freshness and interest requirements (regarding the points of interest in his surroundings). It uses an interest-aware consistency model to limit the amount of information sent to clients, while taking into consideration the real world constraints (e.g. streets, buildings, obstacles, etc.).

1. INTRODUCTION

In the past years, the growth of mobile computing has been huge. One of the main reasons for this increasing interest is the evolution in the software and hardware of mobile devices. With time, computers are becoming smaller and more powerful, which makes them more wearable and more pervasive. This allows developers to create mobile applications that provide all kinds of services anytime and anywhere[1].

Location Based Services (LBS)[2] are one of the most interesting. A LBS can be defined as a service that provides the user with information regarding his surroundings. The evolution in location technology, the increasing development of LBS middleware and the appearance of 3G mobile networks, made the use of LBS in mobile computing to become more popular[3]. Nowadays, a user who wants to obtain information about a certain place, only needs to reach for his pocket, pick his mobile device and ask for the service.

Another type of technology that, in the past years, emerged in mobile computing is Augmented Reality (AR)[4]. This technology is starting to become a reality and mobile devices are one of the reasons for its growth. Mobile AR systems can be described as systems that combine real and virtual objects in a real environment, running in real-time and mobile mode[5]. AR systems, normally have a very simple, user friendly interface, only using the camera of the mobile device. This simplicity allows any inexperienced user to work with any AR application.

With all this development, combining Location Based Services with Augmented Reality has become possible. Nowadays, a user can grab his mobile device, point it to a street or a building and get the information regarding the surroundings through virtual objects. However, these applications have to overcome some challenges. One of them is the amount of data sent to the user. If too much information is sent, the mobile application will display too many objects, cluttering the screen with irrelevant information. Another challenge is the high number of users which makes the number of requests to increase. This means that the use of network bandwidth will be high, possibly leading to high latency.

The goal of this work is to create an AR system, called Idroid, which provides the user with information regarding his surroundings. The system must be efficient, provide high scalability, display only the relevant information to the user, and guarantee the freshness of the information provided.

In this context, freshness means that Idroid guarantees that the information presented in the mobile device is up to date regarding the information available (stored within a database). In other words, data displayed in the mobile device is consistent with the most recent in a database. Another type of freshness relates the real world and the corresponding database information; this issue is out of the scope of this work.

There are already some solutions that provide Location Based Augmented Reality. Two of the most popular are Layar[1] and Wikitude[2]. These provide the user with knowledge of the surrounding area. However, they do not provide mechanisms that guarantee the freshness of the information sent to the mobile devices according to the user interests.

[1]http://www.layar.com
[2]http://www.wikitude.com/en

Other solutions, like GeoPointer[6], make an effort to reduce the amount of data sent to clients through the network (limited to the information that the user is able to see in his mobile device). The disadvantage in this solution is that a slight turn of the camera implies a new request to the database server. The increase of requests may lead to high latency, and therefore the freshness of the information cannot be guaranteed.

Idroid combines augmented reality with an interest aware location based service, providing only the information relevant to user. To achieve high efficiency and still guarantee freshness, Idroid uses an interest aware consistency model [7]. Thus, Idroid allows the freshness of the information to decay gradually as the distance of the user to a point of interest increases. Therefore, data sent to clients is minimized, increasing the scalability of the system.

This document is organized as follows. Section 2 describes the related work and briefly presents some relevant systems. Section 3 describes the Idroid architecture. Section 4 shows the evaluation results and Section 5 summarizes this work.

2. RELATED WORK

Embedding LBS in AR systems is a growing research issue. These systems not only have to address to challenges behind any LBS[8], but also the issues regarding the AR environments[4]. Throughout the years the attempts of merging the real world with a virtual one have become more successful and more efficient. What began by using heavy hardware, such as head mounted displays, has been replaced by small devices that can be carried in a pocket. It is important to understand this evolution and evaluate the previous systems in order to achieve the next step towards a more efficient and scalable solution.

2.1 Head Mounted Display systems

In the early years of mobile AR, the use of head mounted display (HMD) to generate virtual objects was very common. Feiner et al.[9] were the first developers to produce a touring machine that assists the user who is interested in a university campus.

As a user looks around a campus, his see-through HMD overlays textual labels on the campus. The system can display information about the campus, the user's current location, a list of departments and a list of buildings. After selecting a building within the list, the application provides a small compass pointer that guides the user to the building location. Another feature presented by this system is the ability to display more specific information about the buildings. To do that, the user has to select the building and then an informative page will be displayed.

This was the first system that combined LBS with AR. It uses a small, static database and the issue of the freshness of the information is not addressed. With a large, modifiable database, this system does not guarantee the consistency of the results. Another disadvantage of this application is the fact that it uses a HMD. Users normally do no walk around carrying such devices.

2.2 Mobile Phone applications

The implementation of LBSs using AR in mobile phones enables the user to walk with such applications in their pockets. Takacs et al.[10] proposed such a system: it captures the image retrieved by the mobile phone camera and then matches it to a set of images stored in a database. Then, the information related to the image is sent to the client and a virtual tag is displayed on the corresponding object (e.g. a building). The images are searched using an algorithm that only queries nearby images. Along with this, the image recognition algorithm only uses a small part of the image to make the match. These two feature guarantee the efficiency of the system.

By using image recognition, this solution guarantees that the information display is accurate. However, a system like this, only gets information of one building at a time. If the user wants to know his surrounding he has to query for all the objects (e.g. buildings). Also, the information passed to the client is rather limited; only passing a text tag is not really giving the user much information.

MARA[11] is a system that provides points of interest in the range of the user's view without the use of any kind of external feature. It was developed by the Nokia Research Center and it uses the mobile phone sensors to get the current position and heading of the user. The user points the device's camera and then, if there are any available annotations, the information about the surrounding objects is displayed. It is also possible to "click" on a selected object and be redirected to the associated hyperlink.

One drawback of this system is that it does not filter any information passed through the network. All the annotations that are available for the image in the device's screen will be presented, possibly cluttering the display with non-relevant information for the user. In addition, the amount of data passed through the network can be very high, depending on the number of annotations available on the database.

2.3 Augmented Reality Browsers

The appearance of operating systems such as Android and Windows Mobile made the implementation of mobile applications more portable. Since then, a new set of LBS applications that use AR were created.

Layar[3] is one example of such systems. It is a mobile augmented reality application that allows the user to discover new information by looking around the world. With the aid of cameras, GPS, compasses and accelerometers, Layar provides digital information superimposed onto reality.

The concept behind Layar is the use of layers that anyone can access or create. All the information about the world is stored as a layer. Such layers can provide information of the whereabouts of restaurants, coffees, subway stations, etc. For a user to gain access to these layers, he has to subscribe to the one that has the information he desires. The user can also define the radius of his search and therefore limit or wide the search results.

The biggest drawback of Layar is the fact that it is impossible to overlap layers. A user is restricted to the subscribed layer and if he wants to subscribe to another, he has to un-

[3]http://www.layar.com

subscribe the previous one. Other disadvantage is the difficulty of creating layers. They are basically a PHP server that supports JSON with a MySQL database[12]. For the average user this is not trivial and restricts the creation of layers to experienced developers. Finally, Layar does not guarantee the freshness of the displayed results. Nothing assures that, if the layer is altered, the information will be passed to the user, guaranteeing the freshness of the results.

Wikitude[4] is an application, similar to Layar. It was developed by Wikitude GmbH (formerly Mobilizy GmbH) and is a mobile AR system that scans the user's surrounding for geo-references using the camera and the device's sensors. In Wikitude each point of interest belongs to a provider, a so called "world". The functionality of this "world" is similar to the layers in Layar, but it has a difference: they can aggregate different types of places and not just only one. Another difference is the possibility to access different "worlds" at the same time and get information about all of them.

One disadvantage of Wikitude is the number of points of interest displayed when there is no customization of the preferences. Wikitude gathers all the nearby information of all the "worlds" and presents the results to the user, which can make the display cluttered by non-relevant information. This means a large amount of network data and, with high latency, it can make the performance of the application decrease.

GeoPointer[6] is a solution that tries to minimize the information passed to the user. It was developed by Wolfgang Beer as a client-server application with a lightweight client and server component. The client device is only responsible for getting the coordinates of the user (latitude and longitude), as well as the orientation of the device. In the end, the client device is also responsible for representing the information on the mobile device screen. After receiving the direction and position of the user, the server defines his area of interest. First, it calculates a point, called waypoint, which represents the maximum coordinate to present results. This waypoint is calculated by using basic trigonometry relating the current user's position, his direction and a maximum distance. Then, a rectangular area is calculated by using the current position, the waypoint, the direction of the user and a threshold that represents the size of the rectangle. In the end, the user will have results from this rectangle with the waypoint in its center.

GeoPointer seems to be a really good way to present only the information that the user is interested in, but it has a disadvantage. By changing the direction of the mobile device a new request is made and therefore the server has to perform new calculations. A user who wants to know information about a city will turn his device to every direction; this results in many requests that are made very often and, if there is high latency, the results can be inconsistent with the image displayed in the screen, decreasing the performance of the application.

3. ARCHITECTURE

Idroid provides to users the search for points of interest in their surrounding area. To do this, a user only needs to pick up his mobile device, define the search criteria and point the

[4]http://www.wikitude.com/en

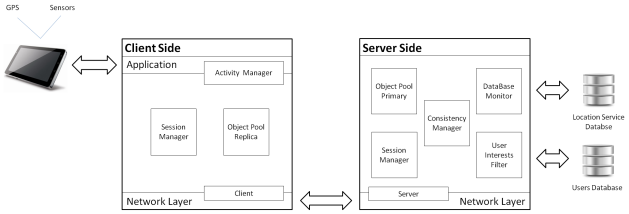

Figure 1: Idroid global architecture.

camera to any target. Then, the points of interest will be drawn on screen. Regarding data freshness, Idroid concistency model is based on Vector-Field Consistency (VFC), providing fresh results using the least resources possible. Idroid also takes into account the interests of the user, updating more rapidly the information closer to him and discarding non-relevant information. We named the Idroid consistency model as VFCdroid.

Idroid is based on a client-server architecture, see Figure 1. Clients submit their locations and interests to the server and wait for the response.

The points of interest presented by Idroid are stored within a database and the information associated to them can be altered by anyone who has access to it (and is authorized). For example, the rating of a restaurant can be modified by anyone who decides to rate it, or, the office hours of the restaurant can be changed by its owners, etc. When a change in the database occurs a **Database Update** is sent from the server to the concerned clients.

Users profiles are also stored within a database. Idroid stores only critical information about the user, such as his interests and his location. A user can change the information on this database by altering his search parameters or by simply moving around, updating only his location. When any of this changes occurs a **Client Update** is sent to the server.

The Client node is composed by several modules, as follows. The **Activity Manager** is responsible for the inputs and outputs of the application. It reads the GPS and Sensors of the mobile device, in order to understand the user's location and heading. It is also responsible to draw the results on the screen.

The **Session Manager** manages the results to be drawn, adding and removing objects. It is responsible for the processing of the messages sent between client and server. The client sends Client Updates that can be of two types: i) if the user changes his interests, then an **Interests Update** is sent; ii) if the user changes his location, a **GPS Update** is sent, with the new location of the user.

The **Object Pool Replica** stores all the objects that are in the user surroundings. The **Client** module is responsible to send and receive information from and to the server.

The Server node is mainly composed by several modules, as follows. The **Session Manager** manages the information at the server side; it processes the messages from the client and the messages to be sent to the user according to his location and interests.

The **Consistency Manager** is responsible to guarantee

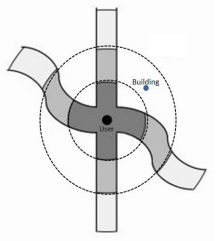

Figure 2: Idroid zones

that the users have consistent objects. It ensures critical updates to be immediately propagated to clients and less critical to be postponed.

The **Database Monitor** periodically monitors the changes made in the Location Service database. If any occurs, it stores the new data in the Object Pool Primary and informs the Consistency Manager about the alterations.

The **User Interests Filter** is responsible to store the information about the users that are requesting services from Idroid. It reads the information from the Users Database in order to understand their interests.

The **Object Pool Primary** stores all the objects that all the users are accessing. The **Server** module is responsible to send and receive information from and to the clients.

3.1 Idroid Consistency Model

Idroid consistency model, VFCdroid, is based on three main concepts: pivots, consistency zones and consistency degrees. A pivot represents a user location. A pivot is used to to calculate the distance between the location of the user and the points of interest.

Consistency zones are formed around pivots. In Idroid, we define three zones. The zones have variable size, that can be modified by the user. A point of interest is in a particular zone if the distance between the user and the point of interest is within the zone size.

Each consistency zone has a consistency degree associated. Consistency degrees vary with distance: zones closer to a pivot have higher consistency degrees. The degree of each zone is defined by a 3-dimensional vector that specify the consistency deviation limits for objects within a zone. The vector parameters are: time(θ), sequence(σ) and value(ν). Time specifies the maximum time an object can be without being refreshed with the most recent value. Sequence specifies the maximum updates an object can receive without being refreshed. Value specifies the maximum difference an object can be from the last refresh. The value of these parameters can also be customized by the user.

3.2 Idroid consistency zones

Regarding the points of interest in a user's reach, Idroid takes in consideration the obstacles of the real world. Therefore, the consistency zones of Idroid are shaped as tubes that cover the streets surrounding the user. This way, Idroid considers the existence of inaccessible buildings even if they are over within radius of the user. The size of each zone is defined by the distance a user can travel through a street. For example, if one zone is limited to 100 meters, then it

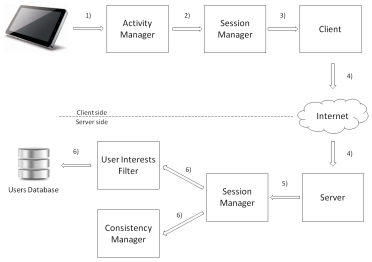

Figure 3: Client update process. The figure shows how the client update is propagated to the server.

is limited, to every possible route, by the same distance, irregardless of the amount of curves the street has. This is illustrated in Figure 2: the building shown is close to the user (birds eay view) but, in fact, it is far away from him because the streets available do not allow him to go directly. Idroid takes into account such scenarios, thus providing only usefull information to the user.

To provide a user friendly interaction, there are default values for the consistency zones limits. The first zone is the smaller one, the second zone is the second smaller and the third zone is the bigger one. To calculate the values we proceed like this: the user defines a maximum distance of search. For the first zone, the limit is calculated by multiplying this value for 1/6. The second zone is calculated by multiplying the distance for 1/2. And the third zone is limited by the maximum distance (more details in the next section).

3.3 Idroid consistency degrees

The values of the consistency degrees vectors can be customized by the user; however, to keep a user friendly interface, there are default values.

In order to understand the user's consistency needs it is relevant to consider if he is walking or driving. This defines the speed the user is moving, thus changing the parameters of the consistency vector. For example, the distance between the user and a point of interest can be traveled more rapidly if a user is driving his car, than if he is walking. This means that such point of interest has to be refreshed more often when the user is driving than when he is walking.

For commodity, the distance between the user and a point of interest can be given in minutes or meters, as shown in Table 1: consistency degrees of a both walking and driving users. In each case, any object in zone 1 will be refreshed every time it gets updated. The values calculated for the time limit correspond to the minutes a user takes to reach the current zone. We estimate that the average speed for a person walking and for a person driving is 4 km/h and 50 km/h, respectively.

As already mentioned, there are two ways a user can receive new information from the server: using Client Updates or Database Updates. We described both now.

3.4 Client Updates

Zone	Minutes	Meters
	Walking or Driving	Walking / Driving
Zone 1	[0, 1, ∞]	[0, 1, ∞]
Zone 2	[zone 1 limit, 5, 25%]	[zone 1 limit * 36/40, 5, 25%] / [zone 1 limit * 36/500, 5, 25%]
Zone 3	[zone 2 limit, 10, 50%]	[zone 2 limit * 36/40, 10, 50%] / [zone 2 limit * 36/500, 10, 50%]

Table 1: Consistency values for both walking and driving users.

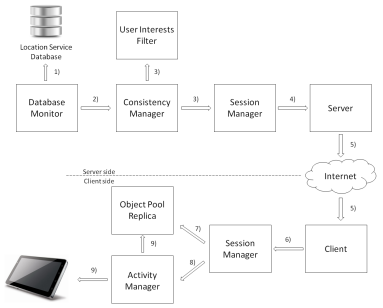

Figure 4: Database update process. The figure shows how the changes in the objects are propagated to the client.

The propagation of the Client Updates has six steps as shown in Figure 3: 1) Listening inputs: the Activity Manager listens to the changes in the the mobile device's application; 2) Informing the Session Manager: if a change occurs, the Activity Manager informs the Session Manager that a new request has to be sent to the server; 3) Creating a new request: the Session Manager constructs the new request with the necessary information. If only the location of the user changed, than the new coordinates will be sent. If the interests changed, the Session Manager constructs a message with new specifications; 4) Sending requests: the Session Manager gives the Client the new request, which is sent to the Server; 5) Processing new request: the Server receives the new request and sends it to the Session Manager to be processed; 6) Storing the new information: the Session Manager processes the request. It informs the User Interests Filter of the changes made on the user and requests the Consistency Manager to check if any object is outdated. The User Interests filter stores the changes of the user in the Users Database.

3.5 Database Updates

The propagation of the Database Updates has nine steps as shown in Figure 4: 1) Checking updates: Periodically, the Database Monitor checks the Location Service Database for new updates; 2) Informing the Consistency Manager: If there is a new update, the Database Monitor informs the Consistency Manager; 3) Verifying consistency: the Consistency Manager, will then check, for every user, if there is any object that needs to be refreshed. If there is, the Consistency Manager informs the Session Manager, in order to respond to the specified users; 4) Creating responses: For every user that needs to be updated, the Session Manager constructs a message with the objects that are outdated; 5) Sending updates: The updates are sent by the Server, through the network, to the Client. 6) Processing updates: The Client receives the updates and sends them to the Session Manager to be processed; 7) Updating information: The Session Manager stores the new updates in the Object Replica Pool; 8) Informing Activity Manager: The Session Manager informs the Activity Manager that the current display needs to be refreshed; 9) Drawing objects: The Activity Manager checks what are the objects on the Object Pool Replica and draws them on the mobile device's screen.

4. EVALUATION

In this section we present the evaluation results. These tests focus on the network savings of Idroid due to the fact that only the really needed information is sent to clients (according to their interests and freshness requirements).

4.1 Workload Description

To exercise the system, and to make the tests as real as possible, we decided to create a group of tourists that are visiting the city of Lisbon. This group start their tour in an hotel, near Avenida da Liberdade.

As tourists, the group has a set of interests in the main attractions of the city. We categorized them in: *restaurant*, *art gallery*, *museum*, *zoo*, *shopping mall* and *embassy*.

For the tests made we simulated a total of 4000 tourists. This group is separated in two: 2000 are walking and the other 2000 are driving.

4.2 Information Download Test

We start by discussing the results obtained for the basic test of downloading all the points of interest. In this situation, the users start the application for the first time and then, Idroid downloads all the information to the user, regarding his interests. The results of the number of points of interest sent in each system are illustrated in Figure 5.

The presented graphs show that the the number of points of interest passed to the user is slightly different. Idroid sends less points of interest than a system without a consistency model (Figure 5). This fact can be explained by the VFCdroid consistency tubes. They take into account the obstacles of the real world, removing the unaccessible places. Without using this consistency model the system will get all the points of interest that are in the users radius. Therefore, the excess sent by the system without VFCdroid is considered non-relevant information for the user.

Sending less points of interest means that less data will be passed to the user. Therefore, the bandwidth usage will also be lower in Idroid, saving network resources.

4.3 GPS Updates Test

In this test we intended to get the results for users that are on the move. We created a short route composed by five different places. In each place, the user will send a new request to the server, only giving his new coordinates. After completing the course, we measured the total bandwidth

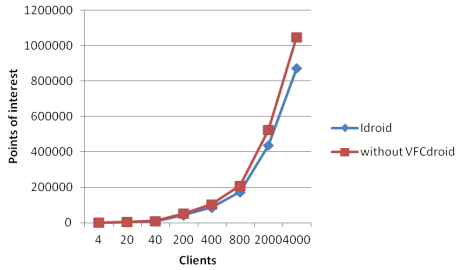

Figure 5: Total number of points of interest sent to every user of the tourists group.

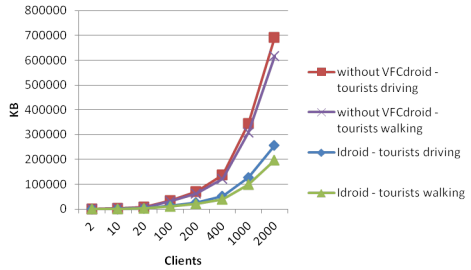

Figure 6: Bandwidth usage (KB) for sending the points of interest to the tourists group after traveling a small route.

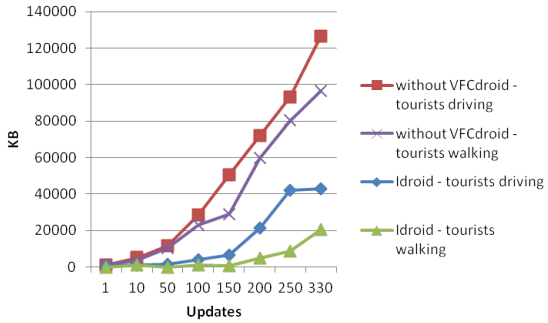

Figure 7: Bandwidth usage (KB) for updating the points of interest of the tourists group.

usage. Figure 6 shows the results of bandwidth usage by the test group.

The results show that, Idroid performance is always better than the system without VFCdroid. There are two reasons for this fact. Firstly, as seen on the previous test (Section 4.2), the amount of points of interest is different in both systems. Idroid has less points of interest than the system without VFCdroid. Therefore, if the user moves, the number of new points of interest to be sent is lower in Idroid. The second reason is due to the presence of a consistency model. Every time a GPS Update occurs, the distance between the user and his points of interest changes. Idroid only updates this information if it is critical, regarding the values of the consistency model. In a system where there is no consistency model, the new distances will always be propagated to the users.

4.4 Database Updates Test

The objective of this test is to understand the behavior of the system when several updates are made to the database. In the end of each test we measured the total bandwidth used when updating the points of interest.

To accomplish this, we made modifications to the information that was in the interest of the users. This means that we updated the points of interest categorized as being *restaurant*, *art gallery*, *museum*, *zoo*, *shopping mall* and *embassy*. The number of points of interest available were 330. It is important to clarify that not all the points of interest are part of the results of the users, however we did not exclude them. The only restriction made to the updates, was that the first one had to be done in a point of interest that was part of the results of every user. The results of the bandwidth usage are presented on Figure 7.

In comparison with the system without the consistency model, we can see that Idroid uses less bandwidth and, therefore, updating less points of interest. Analyzing the bandwidth usage (Figure 7), we can see that, with the growing number of updates, the lines of Idroid grow much slower than the system without Idroid. This means that, with the increasing number of updates, the difference of bandwidth between the two systems will grow. This differences can be explained by the presence of the Idroid consistency model. This model postpones non critical updates and, therefore, not all the points of interest will be immediately updated. Without Idroid, each update to the database will be propagated to the respective users.

We can also see that the gains of bandwidth are not linear. Sometimes, the distance between the lines of the two systems is bigger, and sometimes it gets smaller. This can be explained by the location of the points of interest in the different zones. When the consistency limits of each zone is exceeded the updates must be propagated, increasing the bandwidth usage and the number of points of interest updated. We can also see that, despite this variance, the gains throughout the updates, are still very high.

5. CONCLUSION

In this paper we presented Idroid, an interest aware Location Based Service combined with Augmented Reality. This system uses the mobile device's screen to present only the relevant points of interest in the surroundings of the user.

Idroid achieves high efficiency and guarantees the freshness of the information by using a consistency model, where distant objects request less freshness than closer ones. This consistency model, VFCdroid, selectively schedules the updates based on their importance. Therefore, multiple consistency degrees are applied to different points of interest, which gives the system the possibility to only send critical updates. By doing this, Idroid aims to reduce the network bandwidth on the server, increasing the scalability of the solution. In the end, Idroid consistency model provides an efficient support for any Augmented Reality application.

6. REFERENCES

[1] Höllerer, T.H., Feine, S.K.: Mobile augmented reality. In: Telegeoinformatics: Location-Based Computing and Services. (2004)

[2] Junglas, I.A., Watson, R.T.: Location-based services. Commun. ACM **51** (March 2008) 65–69

[3] Bellavista, P., Kupper, A., Helal, S.: Location-based

services: Back to the future. IEEE_M_ PVC **7** (2008) 85–89

[4] Azuma, R.T.: The challenge of making augmented reality work outdoors. In: In Mixed Reality: Merging Real and Virtual, Springer-Verlag (1999) 379–390

[5] Papagiannakis, G., Singh, G., Magnenat-Thalmann, N.: A survey of mobile and wireless technologies for augmented reality systems. Comput. Animat. Virtual Worlds **19** (February 2008) 3–22

[6] Beer, W.: Geopointer: approaching tangible augmentation of the real world. In: Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia. MoMM '10, New York, NY, USA, ACM (2010) 221–225

[7] Santos, N., Veiga, L., Ferreira, P.: Vector-field consistency for ad-hoc gaming. In: Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware. Middleware '07, New York, NY, USA, Springer-Verlag New York, Inc. (2007) 80–100

[8] Adusei, I., Kyamakya, K., Erbas, F.: Location-based services: advances and challenges. In: Electrical and Computer Engineering, 2004. Canadian Conference on. Volume 1. (may 2004) 1 – 7 Vol.1

[9] Feiner, S., MacIntyre, B., Höllerer, T., Webster, A.: A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. Personal Technologies **1** (1997) 208–217 10.1007/BF01682023.

[10] Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.C., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., Girod, B.: Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In: Proceedings of the 1st ACM international conference on Multimedia information retrieval. MIR '08, New York, NY, USA, ACM (2008) 427–434

[11] Kähäri, M., Murphy, D.: Mara - sensor based augmented reality system for mobile imaging. In: Proceedings of the Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR06). (October 2006)

[12] Arusoaie, A., Cristei, A., Chircu, C., Livadariu, M., Manea, V., Iftene, A.: Augmented reality. In: Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on. (sept. 2010) 502 –509