Enabling Tracks in Location-Based Smart Mobile Augmented Reality Applications

Rüdiger Pryssa, Marc Schickler, Johannes Schobel, Micha Weilbach, Philip Geiger, Manfred Reichert

Ulm University, Institute of Databases and Information Systems, James-Franck-Ring, Ulm, 89081, Germany

Abstract

To assist users through contemporary mobile technology is demanded in a multitude of scenarios. Interestingly, more and more users crave for mobile assistance in their leisure time. Consequently, the number of mobile applications that support leisure activities increases significantly. Mobile augmented reality applications constitute an example for user assistance that is welcome in these scenarios. In the AREA (Augmented Reality Engine Application) project, we developed a kernel that enables sophisticated location-based mobile augmented reality applications. On top of this kernel, various projects were realized. In many of these projects, a feature to enable tracks was demanded. Tracks, for example, may assist users in the context of mountaineering. The development of an AREA algorithm that enables track handling requires new concepts that are presented in this paper. To demonstrate the performance of the developed algorithm, also results of an experiment are presented. As a lesson learned, mobile augmented reality applications that want to make use of the new algorithm can be efficiently run on present mobile operating systems and be effectively realized by engineers using the AREA framework. Altogether, the new track feature is another valuable step for AREA towards a comprehensive location-based mobile augmented reality framework.

© 2017 The Authors. Published by Elsevier B.V.

Keywords: Mobile Augmented Reality; Location-based Algorithms; Mobile Application Engineering; Track Handling

1. Introduction

More and more scenarios shall be assisted by mobile technology\textsuperscript{1,2,3}. For example, when walking around in New York with its numerous tourist attractions, a smart mobile device shall detect these attractions to provide related information to them. In such a scenario, location-based mobile augmented reality is one appropriate concept to properly assist users. In the AREA project\textsuperscript{4}, we developed a kernel that enables sophisticated location-based mobile augmented reality applications. To be more precise, AREA detects predefined points of interest (POIs) within the camera view of a smart mobile device, positions them correctly, and provides additional information on the detected POIs. This additional information, in turn, is interactively provided to the user. Furthermore, an architecture was

\textsuperscript{1} Corresponding author. Tel.: +49-731-5024136 ; fax: +49-731-5024134.
E-mail address: ruediger.pryss@uni-ulm.de
realized, which enables the rapid development of location-based mobile augmented applications on top of the kernel. In general, AREA is used for various scenarios in everyday life and it has revealed practicality in these projects. Recently, a feature for AREA that enables so-called tracks was often demanded. A track, in turn, may be used as the cycle path a user wants to perform in a certain area. Therefore, we developed a track algorithm for the AREA kernel that is presented in this work.

Section 2 presents fundamental aspects of the AREA framework. In Section 3, the coordinate system and the track notion used by AREA are introduced. The algorithm developed for the track handling, in turn, is presented in Section 4. A conducted experiment that evaluates the performance of the new track algorithm is presented in Section 5, while Section 6 discusses related work and Section 7 concludes the paper.

2. AREA Project

The AREA framework (cf. Table 1) basically consists of a mobile augmented reality kernel that enables location-based mobile augmented reality applications. In addition, it provides a sophisticated architecture to create location-based mobile augmented reality applications on top of the kernel. Four technical issues were crucial when developing the kernel. First, POIs must be correctly displayed even if the device is held obliquely. Second, the approach to display POIs correctly must be provided efficiently to the user. To be more precise, even if multiple POIs are detected, the kernel shall display them without any delay. Third, the POI concept shall be integrated with common mobile operating systems (i.e., iOS, Android, and Windows Phone). Fourth, the kernel allows for the handling of points of interest clusters. So far, the AREA kernel mainly consists of three developed algorithms: The POI algorithm, the clustering algorithm, and the track algorithm. The first two algorithms were already presented (cf. Table 1).

Table 1. AREA Project

<table>
<thead>
<tr>
<th>AREA</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android App</td>
<td>√</td>
</tr>
<tr>
<td>iOS App</td>
<td>√</td>
</tr>
<tr>
<td>Windows Phone App</td>
<td>√</td>
</tr>
<tr>
<td>POI Algorithm (all mobile OS)</td>
<td>AREARenderingPipeline Algorithm</td>
</tr>
<tr>
<td>Clustering Algorithm (all mobile OS)</td>
<td>AREACLuster Algorithm</td>
</tr>
<tr>
<td>Track Algorithm (all mobile OS)</td>
<td>AREATrack Algorithm</td>
</tr>
<tr>
<td>Sensor Management (all mobile OS)</td>
<td>SensorFusion(Compass, Gyroscope, Accelerometer)</td>
</tr>
<tr>
<td>Architecture (all mobile OS)</td>
<td>Version 2</td>
</tr>
<tr>
<td>Overall Sensor Management Android</td>
<td>Own approach</td>
</tr>
<tr>
<td>Overall Sensor Management iOS</td>
<td>Built – in functions OS</td>
</tr>
<tr>
<td>Overall Sensor Management Windows Phone</td>
<td>Built – in functions OS</td>
</tr>
</tbody>
</table>

3. AREA Coordinate System and Track Notion

The concept of AREA to relate a user to the objects (i.e., POIs, tracks) detected in the camera view is based on five aspects. First, a virtual 3D world is used to relate the user’s position to the one of the objects. Second, the user is located at the origin of this world. Third, instead of the physical camera, a virtual 3D camera is used that operates with the created virtual 3D world. The virtual camera is therefore placed at the origin of this world. Fourth, the different sensor characteristics of the supported mobile operating systems are covered to enable the virtual 3D world. On iOS, sensor data of the gyroscope and the accelerometer are used, whereas on Android sensor data of the gyroscope, the accelerometer and the compass of the mobile device are used to position the virtual 3D camera correctly. Fifth, the physical camera of the mobile device is adjusted to the virtual 3D camera based on the assessment of sensor data.

1 See http://www.liveguide.de for mobile applications that use AREA.
In order to realize the presented concept, a coordinate system, consisting of three different sub-systems, is required. The first sub-system uses GPS, ECEF (Earth-Centered, Earth-Fixed), and ENU (East, North, Up) coordinates. The second sub-system, in turn, uses a virtual 3D space with the user located at the origin. The third sub-system uses a virtual 3D camera, with the camera being again located at the origin of the 3D world.

As illustrated in Fig. 1, the user is located at the ECEF origin (0, 0, 0). The objects, in turn, are located on the surface of the earth, again using ECEF coordinates. To use this metaphor for the virtual 3D world, two additional transformations became necessary. As a smart mobile device can only sense GPS coordinates, first of all, the GPS coordinates of the user and objects need to be transformed into ECEF coordinates. Second, as a user cannot be physically located at the origin of the earth, ECEF coordinates are transformed into ENU coordinates. ENU coordinates, in turn, enable the required metaphor for the virtual 3D world. Finally, the distance between a user and the objects based on ENU coordinates can be calculated. Finally, Listing 1 presents the notion of tracks based on the presented coordinate system.

Listing 1. Track Notion (Android Version)

```java
//Initialize list of vectors (x, y, z)
List<Float[]> pathStructure = new List<>();
//Set start point as origin
pathStructure.add([0.0, 0.0, 0.0]);

//Transform GPS-Coordinates into OpenGL-Coordinates
for each checkpoint{
    //Calculate angle at start point between this point and north axis (GPS)
    float alpha = calcNorthDegree(Location startPoint, Location currentCheckpoint);
    //Calculate vector
    float x = -OppositeLeg;
    float y = startPoint.altitude - currentCheckpoint.altitude;
    float z = AdjacentLeg;
}

//Generate rectangle between each consecutive point
List<Float[]> path = new List<>();
for each vector in pathStructure {
    if !pathStructure.hasNext() break;
    //Calculate angle between this point and next point on xz-plane
    float angle = calcDegree(vector, vector.next());
    //Depending on angle calculate edges of rectangle
    List<Float[]> edges = calcEdges(vector, vector.next(), angle);
    path.add(edges.get(0)); path.add(edges.get(1)); path.add(edges.get(2));
    path.add(edges.get(1)); path.add(edges.get(2)); path.add(edges.get(3));
}
```

---

a curve is generated as pitch circle consisting of several triangles. 
Next, determine degree between this rectangle and its next possible one */
angle = calcNextDegree(vector.next(), vector.next().next());
/* For each 5 degree interval, generate triangle and form the pitch circle.
Thereby, a pitch circle starts on the last edge of the current rectangle and
ends with the first edge of the next possible rectangle */
List<Float[]> pitchCircle = calcCircle(edges.get(2), edges.get(3), angle);
for each item in pitchCircle{
    path.add(item);
}

4. Track Algorithm

Listing 2 presents the track calculation. Due to space limitations, solely the Android version of the algorithm is
presented. The method onDrawFrame calculates the tracks and presents them to the user within the camera view. The
method onDrawFrame is called multiple times per second, depending on the performance features of the smartphone.
The core calculations are based on the following aspects: First, 3D models for all relevant tracks are calculated (cf.
Listing 2, Line 9). As this calculation requires a lot of resources, all relevant tracks are precalculated. Relevance is
given through several aspects. For example, only tracks are considered that run along a river. These aspects are not
presented in more detail. After getting all relevant tracks, their 3D models are created (cf. Listing 2, Lines 10-12).
Thereby, only those tracks are drawn that are inside a given radius (cf. Listing 2, Line 13). The latter can be specified
by the user. The concrete positioning of tracks is performed as follows: Each track to be displayed is placed at the
origin of the virtual world (cf. Listing 2, Lines 23-26). Following this, the current attitude of the smartphone is
gathered and related to the origin in order to obtain the actual position of a track to be shown (cf. Listing 2, Lines
28-59). Finally, the track is drawn onto the camera view of the user (cf. Listing 2, Line 61).

Listing 2. Track Algorithm (Android Version)

```
public void onDrawFrame(GL10 gl) {
    GLES20.glClearColor(1000.0f); //Specifies object buffer depth for drawing area
    GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT); //Empty drawing area

    //Set rotated camera projection matrix to mMVPMatrix
    GLES20.glMatrixMode(GLES20.GL_PROJECTION);
    GLES20.glLoadIdentity();
    //Set mMVPMatrix
    GLES20.glMatrixMode(GLES20.GL_MODELVIEW);
    GLES20.glLoadIdentity();

    AREATrack[] t = getNewTracks(); //AREATrack[] stores all tracks
    for(AREATrack aT : t){ //Create 3D model for each detected track
        //TOUR Class gets all track reference points and creates 3D track with fixed width
        TOUR nT = new TOUR(aT);
        drawTOUR.add(nT); //drawTour stores all tracks within specified radius
    }

    TOUR[] dT = drawTOUR.toArray(dT); //Renderer needs array instead of a list
    for(TOUR aTour : dT) { //Track positioning on drawing area
        float[] scra = new float[16]; //scra constitutes a 4x4 Matrix, which enables the correct positioning of a track
        Matrix.setIdentityM(scra, 0);
        Matrix.translateM(scra, 0, -aTour.middleX, -aTour.middleY, -aTour.middleZ);
        float[] rotateXMatrix = new float[16]; //4x4 matrix for rotation around the x-axis
        float[] rotateYMatrix = new float[16]; //4x4 matrix for rotation around the y-axis
        float[] rotateZMatrix = new float[16]; //4x4 matrix for rotation around the z-axis
        Matrix.translateM(aTour.mTranslationMatrix, 0, 0, 1, 0);
        float rotateXMatrix = new float[16]; //4x4 matrix for rotation around the x-axis
        float rotateYMatrix = new float[16]; //4x4 matrix for rotation around the y-axis
        float rotateZMatrix = new float[16]; //4x4 matrix for rotation around the z-axis
        Matrix.translateM(aTour.mTranslationMatrix, 0, 0, 1, 0);
        Matrix.translateM(aTour.mTranslationMatrix, 0, -aTour.middleX, -aTour.middleY, -aTour.middleZ);
        //Calculates rotation matrix (i.e., y-axis) based on smartphones rotation; calcRotations() determines the rotation
        Matrix.setRotateM(aTour.mTranslationMatrix, 0, 0, 1, 0);
        //newKnownDegree determines the viewing direction of the smartphone;
        Matrix.setRotateM(aTour.mTranslationMatrix, 0, newKnownDegreeX, 0, 1, 0);
```
Fig. 2 illustrates how track handling is presented to the user on Android and iOS. The first two screenshots are impressions of algorithm results on iOS, whereas the last two screenshots are impressions of the algorithm result on Android. Note that the area algorithm is not shown in this work.

![Fig. 2. Track Algorithm in Practice](image)

5. Experimental Results

In order to evaluate various performance indicators of the AREA track algorithm, we compare these performance issues with the ones of competitive location-based mobile augmented reality applications. Therefore, we conducted an experiment, obeying the following goals. First, we specified the following performance indicators for the Android and the iOS version of AREA: CPU usage, memory usage, and battery consumption. Note that we omit the experiment for Windows Phone. Second, we compare these indicators with the ones of well-known smart mobile applications providing location-based mobile augmented reality including track handling as well. Based on this, we defined an experiment setting for using the smart mobile devices in two different scenarios: (a), when holding the smart mobile device without performing any position change and (b), when continuously moving the smart mobile device.
For the concrete experiment, we use an Apple iPhone 5c for the AREA iOS version and a Google Nexus 5 for the AREA Android version. Concerning the mobile applications AREA is compared to, we determined 3D World Path on iOS and AR GPS Navigator on Android as the competitive applications. In addition, we determine performance indicators for the camera as well as the main menu of the two smart mobile devices. Thereby, camera means that solely the camera function of the smart mobile device was started without using a particular smart mobile application. Main menu, in turn, means that the main menu of AREA was opened without using the augmented function. These two measurements were accomplished to enable a better comparison of the three performance indicators.

Concerning the two scenarios (a) and (b), the following experimental setting was established: for the static Scenario (a), a vice was used (cf. Fig. 3; left pictures) to simulate a user holding the smart mobile device without any position change. For simulating a user continuously moving his smart mobile device (Scenario (b)), we used a ventilator (cf. Fig. 3; right pictures). Note that for Scenario (b) the user himself is not moving, but based on the movement of the ventilator, the number of detected POIs is continuously changing. Consequently, a moving user can be simulated.

For properly measuring the above mentioned three performance indicators, we used the SystemPanel App for Android and the Instruments framework for iOS. Based on this overall setting, each application was evaluated using the same experiment procedure:

- The smart mobile device was set to factory defaults and the battery was loaded to 100%.
- The smart mobile application and the monitoring app were downloaded.
- All other mobile user applications (i.e., except the background processes) were terminated.
- The smart mobile device was mounted to the vice or ventilator.
- The two mobile applications (i.e., test and monitoring application) were started.
- The experiment was conducted over a period of 30 minutes.

Table 2 shows the results of the experiment. For each tested application, the average value of a performance indicator during the 30-minutes experiment is shown. Note that the three applications AREA, 3D World Path and AR GPS Navigator provide alike location-based mobile augmented reality functions.

Experimental results indicate that AREA shows a better performance than the tested commercial location-based mobile augmented reality application AR GPS Navigator and competes well with 3D World Path. Regarding the battery performance indicator, AREA performs best in all scenarios, except for the static scenario on Android. This issue will be tackled in more detail in further developments and experimental results. Regarding the RAM indicator, AREA shows better results on Android than the commercial tool, whereas on iOS it shows slightly weaker results. Therefore, this is another issue that is currently dealt with. Regarding the CPU indicator, AREA is better on Android, while on iOS, it is comparable to the commercial application for the moving scenario and shows weaker results for the static scenario. This is the third issue that is tackled in current developments. Overall, 12 comparative results of AREA with the two commercial tools providing track features were shown. Thereby, seven times AREA showed
Table 2. Experimental Results

<table>
<thead>
<tr>
<th>Device</th>
<th>Scenario</th>
<th>Application</th>
<th>CPU</th>
<th>RAM</th>
<th>Battery</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone 5c</td>
<td>Static (a)</td>
<td>AREA</td>
<td>71,61%</td>
<td>86,00%</td>
<td>17,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Static (a)</td>
<td>3D World Path</td>
<td>62,00%</td>
<td>72,00%</td>
<td>21,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Static (a)</td>
<td>Camera</td>
<td>30,36%</td>
<td>81,00%</td>
<td>13,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Static (a)</td>
<td>Main Menu</td>
<td>14,34%</td>
<td>53,00%</td>
<td>0,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Moving (b)</td>
<td>AREA</td>
<td>72,10%</td>
<td>86,00%</td>
<td>26,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Moving (b)</td>
<td>3D World Path</td>
<td>70,25%</td>
<td>72,00%</td>
<td>43,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Moving (b)</td>
<td>Camera</td>
<td>30,36%</td>
<td>81,00%</td>
<td>13,00%</td>
</tr>
<tr>
<td>iPhone 5c</td>
<td>Moving (b)</td>
<td>Main Menu</td>
<td>14,34%</td>
<td>53,00%</td>
<td>0,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Static (a)</td>
<td>AREA</td>
<td>55,07%</td>
<td>28,00%</td>
<td>54,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Static (a)</td>
<td>AR GPS Navigator</td>
<td>68,74%</td>
<td>53,00%</td>
<td>24,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Static (a)</td>
<td>Camera</td>
<td>22,37%</td>
<td>49,00%</td>
<td>14,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Static (a)</td>
<td>Main Menu</td>
<td>8,91%</td>
<td>41,00%</td>
<td>6,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Moving (b)</td>
<td>AREA</td>
<td>57,21%</td>
<td>52,00%</td>
<td>28,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Moving (b)</td>
<td>AR GPS Navigator</td>
<td>62,58%</td>
<td>54,00%</td>
<td>33,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Moving (b)</td>
<td>Camera</td>
<td>22,37%</td>
<td>49,00%</td>
<td>14,00%</td>
</tr>
<tr>
<td>Nexus 5</td>
<td>Moving (b)</td>
<td>Main Menu</td>
<td>8,91%</td>
<td>41,00%</td>
<td>6,00%</td>
</tr>
</tbody>
</table>

better results, one time AREA showed slightly weaker results, and 4 times AREA showed weaker results. Altogether, the track algorithm developed for AREA revealed meaningful results.

6. Related Work

Previous research related to the development of a location-based augmented reality application in non-mobile environments is described in [12]. In turn, [13] uses smart mobile devices for developing an augmented reality system. The augmented reality application described in [14] allows sharing media data and other information in a real-world environment and enables users to interact with this data through augmented reality. However, none of these approaches share insights regarding the development of location-based augmented reality on smart mobile devices as AREA does. Regarding tracks in mobile augmented reality, only little work can be found. For example, the approaches [15,16,17] present tracks as key feature of (mobile) augmented reality applications. However, algorithms for implementing track handling are not presented. Moreover, only little work exists, which deals with the engineering of mobile augmented reality systems in general. As an exception, [18] validates existing augmented reality browsers. Moreover, [19] discusses various types of location-based augmented reality scenarios. More precisely, issues that have to be particularly considered for a specific scenario are discussed in more detail. However, engineering issues of mobile applications are not considered. In [20], an authoring tool for mobile augmented reality applications, which is based on marker detection, is proposed. In turn, [21] presents an approach for indoor location-based mobile augmented reality. Furthermore, [22] gives an overview of various aspects of mobile augmented reality for indoor scenarios.

Another scenario for mobile augmented reality is presented in [23]. The authors use mobile augmented reality for image retrieval. However, [20,21,22,23] do not address engineering aspects of location-based mobile applications. In [24], an approach supporting pedestrians with location-based mobile augmented reality is presented. Finally, [25] deals with a client and server framework enabling location-based applications. Altogether, neither software vendors nor research projects provide insights regarding the engineering of a location-based mobile augmented reality kernel, which includes track handling.
7. Summary and Outlook

This paper gave insights into the implementation details of the track algorithm for the AREA kernel. On top of the presented algorithm, the paper showed experimental results of the algorithm. To be more precise, an experiment with two commercial tools have been carried out and results be presented. The experimental results demonstrated that the track algorithm of the AREA kernel had shown a good performance compared to competitive location-based mobile augmented reality applications. In general, AREA shows that a mobile augmented reality kernel providing sophisticated features like track handling can be accomplished on different mobile operating systems. In future, new features will be addressed for the AREA kernel. For example, we currently integrate a trail navigation for mountainous regions. Altogether, mobile augmented reality shows that mobile applications become more and more mature. On the other, suitable concepts are required to deal with the complex peculiarities of contemporary mobile technology.

References