Increasing the Accuracy of Indoor Localization Applications by Using Predefined Markers and the Phones Camera

Szabolcs Orban
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
orbanszabolcs11@gmail.com

Teodor Stefanut
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
teodor.stefanut@cs.utcluj.ro

ABSTRACT

This article shows our approach on determining the global position of an Android device user, who wants to navigate in an interior of a building. The application is based on our previous application, which offers the possibility to navigate through a 3D model using the movement of a mobile phone, determined by its own sensors.

Previously the application used the navigations starting position given by the user himself. The method presented in this article adds to the accuracy of the existing application by using predefined points detected with the phone's camera. These points are placed in the building to determine the initial position or to correct the placement errors of the user. After the location is determined the application switches back to the navigation using the phone's sensors.

INTRODUCTION

In our everyday life we use our cellphones more and more as time passes. Therefore, it is a perfect platform to implement different applications to ease everyday activities or to get information faster.

In this article we propose a solution to a scenario where a mobile phone user is situated in the interior of a huge building with a complicated infrastructure for the first time and wants to reach a specific destination also located in the building. The ideal method would be getting the exact location of the user and navigating him to the desired destination using the mobile device without any other information input besides the destination from the user's part.

To implement an application of this kind it is advantageous to select one of the two main and wide-spread platforms, Android or IOS. Because Android is more accessible and the development and code signing is easier we choose this platform.

The most popular and easy localization method for a mobile application is the use of the GPS (Global Position System) functionality. It can specify the user's location from a wireless network or just from activating the GPS directly from the mobile device. The main problem with this method is that it's transmitted frequency is weakened by the building's walls when used inside and consumes a lot of battery [1].

Another possibility to determine the user's positon is by using the RSS (Received Signal Strength) of the wireless signals of routers placed in the building. This method needs routers to be placed to cover all the areas of the building and the resulted location is not very accurate. [1]

In our previous article about this subject we did not take into consideration the global or initial location of the application user. [2] The main goal was to navigate through the building using the predefined position introduced initially by the user. The navigation started after the given starting and end points and consisted in mapping the shortest path with the help of the predefined 3D model of the building. The navigation itself used the device's sensors to get the direction and movement of the user. The direction was given by the gyroscope and the magnetometer and the movement was given by a step counter implemented from the accelerometer's data.

The advantage of this approach was the constant mapping of the user's position and movement in the interior of the building, but was heavily based on the initial global position introduced by the user itself. It had no correction or misplacement detection implemented and this way if the user did not know his exact position, he could use the application wrong.

So, finding the current positon of the user automatically or resynchronizing it represents the main flaw of the previous implementation. In this article some possible solutions are highlighted and the chosen one is detailed and implemented.

The proposed solutions make use of the mobile device's own camera for detecting and identifying key points or objects that can be used to determine the current location. One approach is a marker based method, which consists in placing predefined images of markers in the building. The application detects these markers and determines the location on the map. One other approach is feature based. Using a local database on the device, the camera scans the surrounding area and maps it in this database. This approach needs a predefined database of the key points of the building.

The chosen method uses predefined markers in addition with an augmented reality framework, to automatically detect the markers and display different objects on the scanned area that guide the user to the desired destination.

Augmented reality in this application is used to present the direct view of the real-world environment with added augmented elements that are computer generated in real time.

RELATED WORK

Global localization methods

Global localization methods help define the user's location from where navigation methods can be used. These methods consist in detecting a predefined point in the environment and searching its correspondent location already mapped in a database. The methods differ on the way of defining these key points.

A localization method on a smaller surface is presented in the article [3]. The authors use a simultaneous localization and mapping algorithm to construct a real time discovering and tracking method for global localization. This real time tracking after an initial detection can easily adapt to different scaling transformations of the object.

For this method the initial object must be first scanned and mapped into a local database. The end user will use the mapping to re-localize the object as a starting point. This object can easily be detected from any angle or point of view.

As stated by the authors of the article [4] localization done only by one tracking method is not reliable every time. Consequently, they propose a solution that uses two localization methods and uses a switching algorithm to change the one in use depending on different factors.

The two algorithms used a feature based method, which uses already extracted knowledge points as prerequisite and a marker based method, which needs the device's camera to track for markers and to estimate de 3D position of the user.

The application uses a client-server model. All the computing, deciding and method switching is made on the server side. The client side must ensure to send continuous information about the camera's view, to extract potential key points or to detect markers.

The algorithm on the server side uses a fail-safe method to determine which algorithm to use. If the marker based method does not find a marker or the feature based method cannot extract key points it changes to the other one. In addition to the switching they also use sensor based localization and real-time mapping for improved stability of the system.

In our earlier article about localization and navigation [2] we presented a solution for navigating in the interior of a building using its 3D model and the mobile phones sensors. In order to start navigation, the user needed to enter the current location and desired destination. The application assumes that the user introduced his global position correctly and displays the 3D model of the interior of the building. On the 3D model a path is mapped dynamically formed by connection points. These connection points are predefined and only those composing the shortest path will be displayed.

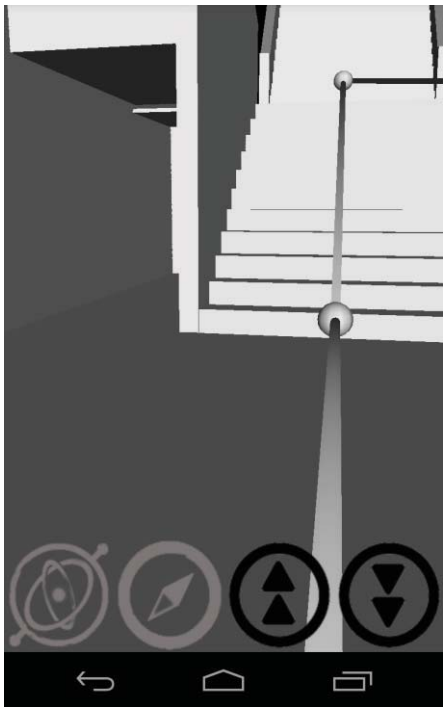

Figure 1. Sample screen of our existing navigation application.

The user can interact with the 3D model by moving the phone around himself and the model shows the corresponding view synchronized with the real world. This has been implemented based on the phone's gyroscope, which keeps track of the rotation movement of the device.

Using the phone's accelerometer, the user's traveled distance could be determined through the implementation of a step detector. Analyzing recorded data, the algorithm decides in real time if the user took a step and maps the movement on the 3D model.

In order to bind the distance travelled with the facing orientation, the device's geomagnetic field sensor was used. This sensor monitors the earth's magnetic field and thus behaves like a compass. Combined with the accelerometer, it allowed the mapping of the user's exact movement on the 3D model.

The main flaws of this method were the initial global localization method and the possible errors that may occur while detecting user's movements, which can cause the application to lose track. To adjust or correct these errors, the user had the possibility to manually synchronize by indicating a connection point forward or backward on the path. The approach presented in this article will aim at resolving these flaws.

Existing tools

There are several tools which can be used for detecting markers. One of them is Vrui VR toolkit [5], which was designed to implement a toolkit for scalable and portable

applications. Its most notable strength is the focus on interactivity and immersive display environments.

Another example of an augmented reality toolkit is Meta Developer Kit [6]. It is produced by a Silicon Valley company that provides a 3D object projection using an augmented reality headset. These objects are able to lock onto areas where specific markers are placed in the real world.

A final example of an existing tool is Vuforia [7]. It provides SDK for Android, IOS and Unity, which is capable of recognizing frame markers, images, text or 3D objects of different forms, such as cylinder or box. Vuforia uses a natural feature tracking algorithm, which makes augmented reality practicable on low performance devices. The exact algorithm used by Vuforia is not public for the community, but it can be used and improved as part of the samples they provide. Because of its good performance and range of samples we decided to use Vuforia for our application.

RESEARCH ON THE MARKERS

Markers

Vuforia uses a variety of marker types to detect objects. The main type is a plain image, which can be detected from a fair distance, with different luminosity and from a wide angle. In addition, there are cylinder type or cuboid type markers. The difference between these and the simple image marker is the mapping of the key points on the desired object.

The used algorithm scans the input image from the source code and determines important key points specific to the image. Then, while scanning the input image from the camera, it tries to find these key points. Once found it keeps track of them until they get out from the frame or the camera cannot get a clear image of them.

The Vuforia platform also provides for developers the possibility to deposit the marker images on their cloud storage. This way the client application functions only if it has internet connection and access to the database, but the input images can easily be added, or edited without the application being modified.

Marker definition

Vuforia has defined a number of predefined frame markers that are present in its source code. Every one of these markers contains a unique pattern on its frame with a series of black and white squares, as seen in Figure 2.

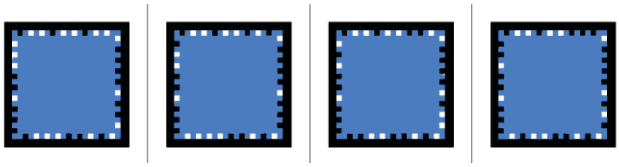

Figure 2. Example of predefined frame makers.

The interior of the marker is transparent and thus allows the integration with any desired image, without any influence on the detection of the marker. For these predefined markers the Vuforia framework uses a more efficient and precise algorithm for detection.

In addition, the application developer can define its own images for detection, which must be added to the online database by specifying the marker's type. After the upload the marker is given a rating, which shows the value of the detectable key points. If the rating is low the desired image has a poor detectability and needs to be changed. An image with a high rating must have a high amount of detail, to allow the extraction of a high number of key points.

We tried to define our own markers and to make different variations of them. After the creation we uploaded them to the online database for ranking and concluded the results. The simplest pattern to create was a chessboard like 8x8 pattern only with black and white colors shown in Figure 3. From this we varied the two colors in each square to generate different markers. For the tests we also used a same type of marker with different color variations.

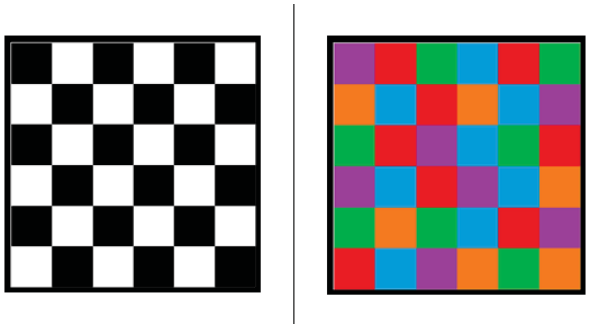

Figure 3. Proposed markers.

After the upload and gathering the rating, the results showed that the markers needed more detail, because the rating was average. Consequently, we concluded that if we want to keep the pattern and to generate more key points, we need to separate the marker's squares into squares of smaller sizes. The key point extraction algorithm used by Vuforia works only on greyscale images, so using colors as an approach to add more details to the image is not has not been successful.

Taking into consideration the presented results, the final design of the marker is very similar to the predefined markers of Vuforia. The only difference is the transparent interior, which is an advantage because it can contain any desired image and the marker can easily be added to any surface without standing out. Because of this and because the predefined markers have maximum rating, we use these in our application.

Marker tolerance

The marker's tolerance to modifications depends on the number of extracted key points and in case of a user defined marker it must have a high rating. In case of the predefined frame markers the tolerance is as optimized as possible.

We tested the tolerance of the user defined markers from Figure 2. The considerations were: distance of detection, angle of vision and the luminosity. For the tests we used a 10x10cm image and changed the mobile device's

				Normal Luminosity		Low Luminosity	
				Angle of Visibility		Angle of Visibility	
Distance (cm)	Distance Measured (units)	Distance Calculated	Error (cm)	90 - 0	0 - 90	90 - 0	0 - 90
20	100	20.40816327	0.4081632653	80	80	60	45
30	155	31.63265306	1.632653061	80	80	60	40
40	203	41.42857143	1.428571429	80	60	60	30
50	243	49.59183673	0.4081632653	80	60	60	30
60	290	59.18367347	0.8163265306	65	50	50	30
70	341	69.59183673	0.4081632653	60	45	45	30
80	385	78.57142857	1.428571429	55	45	45	30
90	434	88.57142857	1.428571429	50	30	45	30
100	474	96.73469388	3.265306122	45	30	40	30

Table 1. Distance and angle of visibility measurement.

distance to the image by 10cm every time. For every section of 10cm the angle in which the image was detected and lost was also observed. The measures were repeated until the image could not be detected by the device's camera. All the measurements were repeated in strong and poor luminosity.

At that moment we did not know the connection or the scale between the width of the original image in centimeters and the unit of distance measurement returned by the application. The virtual distance measurement was made with the translation and rotation components of the detected image.

According to the measured results, monitored properties and the actual distance in centimeters increase in a linear fashion, so we could determine a constant which helped to transform the result into centimeters. This constant was determined in such a way to have the smallest error in centimeters. In our case measured value is 4.9. This value varies for each marker that has a different width from 10cm. The resulted measurements can be seen in Table 1 and the comparison of the actual and measured distance can be seen in Figure 4.

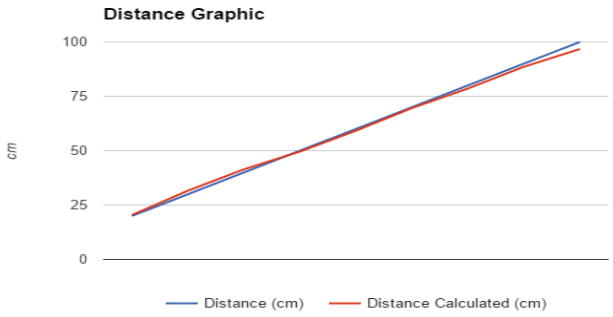

Figure 4. The Graphic of the actual distance and the calculated distance in cm.

Of course the camera of the device has a huge influence on the measurements. We used an 8MP camera, which is the most common on the mobile cellphones.

The tests show that the angle of visibility at smaller distances reaches about 80 degrees and at bigger distances about 40 degrees. While the angle of detection is smaller than the angle on which the applications loses track of the marker, the application provides a continuous tracking of the detected key points.

The maximum distance at which the 8MP camera could detect the marker was about ten times the width of the image. This of course scales with the size of the marker. The luminosity did not affect the measures on a very significant scale.

PROPOSED METHOD

Our proposed method uses the Vuforia framework to detect precisely placed markers inside the building to determine the current position of the user. After the location is computed the application automatically adjusts the 3D model of the building to match the viewer's sight. Synchronization between virtual and real worlds is further performed with the help of the device's sensors, while continuously scanning for further markers. Every time a new marker is identified, the position of the user is resynchronized on the device.

For a more precise location computing we use a distance measurement from the detected marker, based on the analysis of marker's distortion. Furthermore, making use of the Vuforia capabilities, we display a 3D object on the top of the marker. The arrow will point in the direction of the next connection point.

Proposed markers

Because of the difficulty of the definition of custom markers with a high rating and the difficulty of detection we do not use any user defined markers. For a better tracking performance, we decided to use the predefined frame markers which are detected more efficiently.

For the detection to go naturally without the user's need to specifically position the device towards a marker, we propose to place all the markers on the ground. The natural holding of a mobile phone is 45 degrees to the ground's normal vector, so this placement is easily justified.

Because the maximum detected distance is ten times the width of the marker we use 20x20cm markers. The markers are placed on the ground so it is no need for a maximum distance greater than 2m.

Due to the contrast difference of the colors of the black and white markers with transparent background, which can be any image, have high tolerance to luminosity change, they are a good choice for using in the interior of a building.

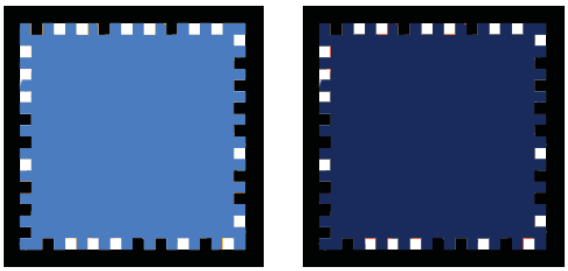

Figure 5. Frame markers with different luminosity.

Distance Measurement

When the marker is detected in the camera's frame we have only unverified information of the user's position (inferred using phone's sensors), but we have the known location of the marker. Because of this it is necessary to compute the distance between the user and the marker, which will give us the exact position of the former.

The distance is computed using the distortion of the detected marker to the sample image. The result is a transformation matrix, from which the translation and rotation values can be extracted. By knowing also the exact width of the marker, the distance can be determined.

The transformation matrix obtained from the detected image has the following format:

$$\begin{bmatrix} Transform\_XAxis.x & Transform\_YAxis.x & Transform\_ZAxis.x & Translation.x \\ Transform\_XAxis.y & Transform\_YAxis.y & Transform\_ZAxis.y & Translation.y \\ Transform\_XAxis.z & Transform\_YAxis.z & Transform\_ZAxis.z & Translation.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The extracted translation matrix is the following:

$$\begin{bmatrix} 1 & 0 & 0 & Translation.x \\ 0 & 1 & 0 & Translation.y \\ 0 & 0 & 1 & Translation.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finally, the resulted rotation matrices are as follows:

Around X axis:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Around Y axis:

$$\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Around Z axis:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The formula for computing the distance from the user to the marker is the following:

$$distance_{units} = (Translation.x)^2 + (Translation.y)^2 + (Translation.z)^2$$

$$distance_{cm} = distance_{units}/constant$$

Knowing the distance from the marker and the transformation matrix, the current position of the user can easily be determined. In this way the displayed 3D model can also be synchronized.

Mapping Plan

As presented in previous sections all the markers will be placed on the ground. Every one of the markers will be near or on a connection point of the building so the user has a very low chance to miss any of them.

In the case of a large building there is a problem with the number of the markers. The Vuforia framework has only 200 predefined frame markers. In the mapping process it is inevitable to use all the 200 markers or to place them more than once.

Our proposed solution for this problem is to leave a small number of markers as boundary markers that separate several marker sets. Each separated set contains the remaining markers, which can be repeated in every separated set only once. These boundary markers will always be placed between inevitable connection points that are detected every time the user passes next to them. In addition, the detection algorithm keeps track of the detected marker sets and after every boundary marker it determines in which wing or section of the building the user is located.

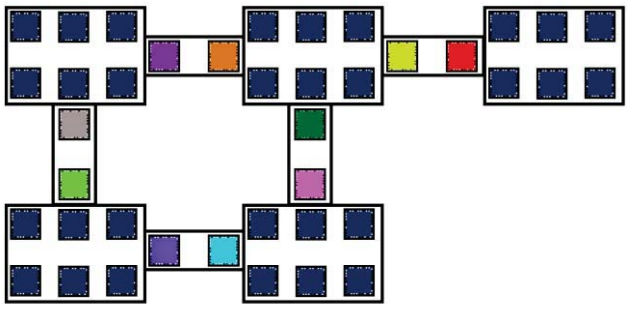

Figure 6. Boundary marker distribution example.

Limits

The used methods and algorithms all have their limits. The detection algorithm has a maximum distance and an angle in which the marker is still detected. The distance limit is solved by placing all the markers on the ground with a 20x20cm size, so the clarity is not a problem. The

number of markers used in mapping is also limited, but it is solved by using boundary markers.

What is more, the number of markers detected in one frame is also limited by Vuforia to 5. In our case, the placement and the ambiguity on which 5 markers to track is also not an issue because all the markers are placed on connection points of the building, so they are not present on the same frame very often.

MAIN FLOW

Compared to the previous flow, where the user had to specify his current location and his destination to navigate through a building, in this version the user needs to specify only the destination and to scan the first marker he sees on the floor to get the shortest path.

After the initialization, the 3D model of the building is displayed with the dynamically drawn path and connection points marked with spheres. As the user moves in the building his movement is also tracked on the 3D model.

Without the need to search and position the mobile device on a marker, it automatically starts the camera in the background and scans for markers. If a marker is found, the user can switch to camera mode and on top of the marker an arrow type 3D object is placed inside the scanned camera frame indicating the way to the destination.

If the sensor based navigation has an error and misplaces the user in the virtual environment, after every marker reached by the user the application automatically resynchronizes and corrects its misplacements.

CONCLUSION

This approach is an improvement of the previous version of the application, which used only the mobile device's sensors for navigation. The user had to know and introduce himself the starting point. In this method the application only needs to scan a marker on the ground to compute the user's position. This method has a more precise determination of global localization and does not need any additional input or correction by the user.

There is no need for the user to search for markers, because of their placements in key connection points of the halls. If there are any navigation errors, after a marker is detected, the errors are corrected, so the use of the application is more practical.

The next step of the research will be testing the performance of the application with the implemented marker detection on the field and to improve any downsides, lacks or failures that may occur and were not dealt with in the design phase.

As a further improvement we could remove all the 3D model mapping from the application and leave only the augmented reality representation. This consists in mapping the dynamically computed path directly on the scanned camera frame.

REFERENCES

1. Shala, U. and Rodriguez, A. Indoor Positioning using Sensor-fusion in Android Devices, *School of Health and Society, Department Computer Science*, Kristianstad University, Sweden, 2011.
2. Orban, S. and Stefanut, T. *Indoor Localization and Navigation Using Phone Sensors and a 3D Model of the Building*, 2015.
3. Martin, P., Marchand, E., Houlier, P., Marchal, I. *Decoupled Mapping and Localization for Augmented Reality on a Mobile Phone*, 2014.
4. Miyagi, A., Yoshihara, D., Kusui, K., Kimura, A. and Shibata, F. Mobile Augmentation Based on Switching Multiple Tracking Method, Graduate School of Information Science and Engineering, Ritsumeikan University, Japan, 2014.
5. Vrui VR Toolkit, http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/
6. Meta Developer Kit, https://www.metavision.com/
7. Vuforia, http://www.vuforia.com/